A GRAPHICS RASTERIZER IC

by

Martin John Izzard BSc. Eng.

This thesis is submitted in partial fulfillment of the requirements for the degree of Master of Science in Engineering, in the Department of Electronic Engineering, University of Natal, South Africa.

Durban March 1987.

Preface.

The experimental work described in this thesis was carried out in the Department of Electronic Engineering, University of Natal, Durban, from February 1985 to February 1987, under the supervision of Professor K. F. Poole, assisted by Mr. R. C. S. Peplow.

The author certifies that this thesis represents his own original and unaided work except where specifically indicated in the text and has not been submitted to any other university for degree purposes.

Acknowledgements

I would like to thank my supervisor, Professor Poole, for his guidance and encouragement during the course of this thesis. Thanks also to Mr Peplow for his practical help and advice.

Special thanks to my wife, Paula, for invaluable help in producing this document.

I thank the staff of the Department of Electronic Engineering for providing the resources and the environment necessary for study, and my fellow students for their friendship and help.

I gratefully acknowledge the financial support of the Council for Scientific and Industrial Research and the University of Natal. I also wish to thank the Foundation for Research Development for the financial assistance provided to fabricate the IC; and the National Electrical Engineering Research Institute, the University of Natal, and Plessey Semiconductors for the use of their design facilities.

Abstract.
A graphics rasterizer IC.

A single chip line-rasterizer that overcomes the major bottleneck in graphics display systems has been designed by the author on a 4408 element gate array marketed by Plessey Semiconductors limited. The rasterizer was fabricated by Plessey using their 2 micron, double-level metal ISO CMOS process, in the United Kingdom.

Two identifiable bottlenecks in the redraw speed on a general graphics display system are video memory bandwidth and rasterization speed (in dots produced per second). The rasterizer described here is capable of working in parallel with other rasterizers to overcome the rasterization bottleneck. Systems incorporating it are flexible and expandable.

The rasterizer requests a primitive from a host or master part of the system. Once it has a primitive to work on, it begins rasterization. The rasterizer queues requests to write dots to the video memory part of the system. The device accepts two ordered pairs of 16–bit numbers as start-of-line and end-of-line coordinates, on an 8–bit bus; the dot addresses are in the form of two 16–bit numbers on a 32–bit bus.

Simulation with CLASSIC showed that the device could be clocked at up to 8 MHz and would then produce dots at between 2 MHz and 4 MHz (dependent on the type of line) after the initial analysis overheads. This means that any video memory bandwidth may be fully used with this device and any improvements in memory bandwidth may be taken advantage of in a system using the parallel rasterization scheme.

The Plessey test engineers exercised the device to prove the success of the fabrication. Further tests were performed by the author. In these, the rasterizer was seen to gather data correctly.

The rasterization of a range of different types of lines, manhattan and general, short and long and lines of different direction, was tested. The various algorithm terminations were verified and all branches exercised. The flow control on the pixel bus was checked. The device used for all the tests, performed correctly at 10 MHz (design specification 8 MHz) which corresponds to a maximum rasterization speed of 5 MHz for 0° and 90° lines and between 2.5 MHz and 3.3 MHz for general lines.

The results show that the rasterizer performance will allow full use of the memory bandwidth of the system and hence overcome the major bottleneck in many graphics display systems.

Contents.
A graphics rasterizer IC.

List of Figures, Plates and Tables.
A graphics rasterizer IC.

Figures

Plates

Tables

Introduction.
A graphics rasterizer IC.

The purpose of this project was to design, fabricate and test an ASIC for use by an associate team involved in the development of a high performance Graphics Display System.

Chapter 1 provides an overall view of Graphics Display Systems. The Graphics Display Pipeline is defined as a sequence of logical operations on data representing a picture. Some examples are given. Speed bottlenecks in this pipeline are explored and two major bottlenecks are highlighted: rasterization and memory bandwidth. An objective in the Department of Electronic Engineering at the University of Natal, was to produce a high performance graphics workstation. Consideration of the objectives and the bottlenecks led to a specification for the system and the various subsystems.

In Chapter 2 the design environment is outlined and traditional rasterization techniques are critiqued. Parallel rasterization is proposed as a solution to the rasterization bottleneck and described in general terms.

Chapter 3 contains the details of the paper design of the rasterizer. This is essentially a digital system design. The introduction summarizes the chapter and the remaining sections deal with major topics: external operation, drawing algorithms, the arithmetic architecture, and the control logic.

Chapter 4 deals briefly with gate arrays in concept and then goes on to describe the Plessey gate array range and provides an introduction to their gate array CAD suite.

The Computer Aided Design of the rasterizer is the subject of Chapter 5. At this stage the scope of the work done is outlined and then the CAD procedure is detailed. Details of the simulation and evaluation of a logic circuit along with examples of selected types of analyses are provided. Some specific problems related to the gate array design are discussed. The generation and evaluation of test vectors, followed by an outline of placement and routing procedures is included.

Chapter 6 concludes with the testing procedure and discusses the test results and chip performance.

Chapter 1.
Graphics Displays

1.1. Definition of a Graphics Display System

Graphics displays are used on a wide range of computer equipment from Personal Computers to high power Workstations. The Graphics System may be defined as the set of hardware and software that is mainly dedicated to putting the picture that the user interacts with in front of him.

1.2. The Graphics Display Pipeline

In the context of the above definition, the user clearly requires that the hardware and software allow him, firstly quickly to construct his picture, secondly quickly to show him any changes he makes, and finally to keep his work safe. The first and last requirements are not demanding. However, the second is demanding since it ultimately means redrawing the display. Redrawing involves plotting a large number of vectors, especially for activities such as IC design, and hence has potential to be very slow: a circuit schematic involving several thousand gates can take more than 30 seconds to redraw.

The process of transforming a picture from the form in which it is stored to the form in which it is easy to display on a raster monitor may be referred to as the Graphics Display Pipeline.

In all Graphics Display Systems the problem is generally the same: the concept of the picture that the user wants to see has to be converted into the individual Pixels stored in the Graphics Memory or Frame Buffer.

The conversion process may be further broken down as the data originally describing the picture is made ready for display. The logical steps are as follows:

1. A model of the picture in the form of data or a programmed model must exist.

2. A display file compiler converts the model to a display file of a particular level of sophistication; for example, a sophisticated file may be a list of parts of a physical object related to one another in some way, or it may simply be a set of Graphics Primitives (vectors, polygons, circles, etcetera).

3. Display file post-processors repeatedly produce new display files until the appropriate level of sophistication is reached.

4. The Rasterizing processor converts the last display file into a Pixel-image in the Frame Buffer or Pixel Memory. The simplest form of display file is a set of vectors, each represented by two ordered pairs of screen coordinates. A vector must be converted into a number of individual pixels, the number depending on the length of the vector. Each pixel is represented by one or more bits, depending on whether the picture is binary, grey-shade or colour.

5. The Display Refresh processor utilizes the Pixel-image to produce the necessary video signal to drive the Display Monitor.

In the above the data is operated upon by a number of logical processors [Myers84]. Successive representations of the picture may or may not reside in the same memory or be dealt with by the same physical processor. Each logical processor may consist of different software on the same hardware or different software on different hardware or separate pieces of dedicated hardware. The greater the number of physical processors available the higher the

degree of parallelism and hence the higher the potential throughput. The further back up the pipeline that dedicated hardware moves, the faster the system is likely to become; however, at the same time, the further down the pipeline software based processes move, the more flexible the system is likely to be.

1.3. Typical Graphics Display Systems

An example of a graphics display system is Silicon Graphics' IRIS workstation [Nicke84]. In IRIS a MC68000 processor runs application tasks and handles display files. Their "Geometry Engine" [Clark82] performs coordinate adjustments, clipping and scaling. A 4×4 AM2903 bit-slice microprocessor rasterizes the resulting screen coordinate graphics primitives.

A Personal Computer based on the Intel 8086 and an applications package such as PCAD form another graphics display system. The graphics package, using the available general purpose microprocessor performs all the tasks in the pipeline.

It is clear that the hardware configuration will have a large impact on the performance of a Graphics System. On one hand, a single CPU will do the whole job of constructing the display file, scaling and clipping, and rasterization; on the other hand, there may be a dedicated piece of hardware for each task.

1.4. Graphics Display Pipeline speed bottlenecks

In order to improve the response of graphics display it is necessary to identify the speed bottlenecks. Each logical processor is a potential bottleneck, for instance clipping a full picture to the size of the display involves matrix operations which are time consuming on any hardware because of the number of simple arithmetic operations per matrix operation.

It may, however, be assumed that any interactive manipulation of the schematic will be done in screen coordinates, at a stage in the conversion process when much of the hard work has been done. This makes sense since the conversion to screen coordinates and back to world coordinates need only be done once. The most pressing problem is redraw of the screen–coordinate display file.

It is best to assume that the entire display file is in system memory. If not, part of the display file is on a disk drive and the bottleneck would clearly be disk access time. A prerequisite is thus a sufficiently big system memory to contain the entire display file.

Getting graphics elements out of system memory and to the engine that rasterizes them may be done at speeds of the order of 2 megabytes per second by using a direct-memory-access (DMA) subsystem (that is if the rasterizer is a separate piece of hardware). An element such as a vector may consist of, for example, four bytes per ordered pair and one or two bytes for attributes such as colour; hence, of the order of ten bytes are required per simple element. On a high resolution system the majority of vectors are over 100 pixels in length. It is clear, therefore, that elements representing at least 20 megapixels per second may be transferred to rasterizer hardware, using simple DMA transfer.

Rasterization is of major concern since it is at this stage that the volume of data required to represent the picture suddenly expands, and the process of expansion is both computation and input/output (IO) intensive: many coordinates have to be computed and then stored in the pixel memory.

Memory bandwidth is the final possible bottleneck [Whitt84]. The pixel memory is usually constructed from conventional dynamic random access memory (DRAM) because it is cheap and dense, and very large frame buffers are required for high resolution graphics displays. The frame buffer memory should be separate from the system memory for a high performance system or there

would simply be too many devices using the same data and address buses at once. The read/write bandwidth of DRAM is approximately 2 megahertz [Motor82]; this bandwidth must be shared between the subsystem writing pixels, the screen refresh hardware and the DRAM refresh subsystem. In many systems approximately half this bandwidth is available to the rasterizer [Gutta86]. In order to free some bandwidth some systems use a special dual ported DRAM called VRAM which has an on-chip shift register to serve the screen refresh hardware; however VRAM is expensive and hence is limited to high-end systems.

The video refresh hardware is not a direct bottleneck: it must simply be capable of reading the whole frame buffer in one frame-time (50/60 hertz) which is not visible to the eye. The video refresh hardware does, however, use memory bandwidth thus contributing to the previous bottleneck. The hardware must be designed to allow the transfer of large numbers of pixels into shift registers (on or off the memory chips) to cut down on bandwidth use [Whitt84].

1.5. A Graphics Display System for Natal University

One of the major fields in which Graphics Display is necessary is IC design where large layout schematics must be constructed and manipulated. In the course of manipulating any display interactively, frequent screen redraws are necessary to be sure of the current status of the display file. The schematics are often very complex, consisting of a large number of vectors, resulting in unacceptably long delays (tens of seconds).

Existing IC design environments use either graphics terminals on a powerful multi-user minicomputer such as a DEC VAX or a sophisticated workstation such as an Apollo for their graphics display systems.

The load of graphics manipulation on a multi-user system is high because of the large amount of computation and IO necessary to construct an image.

High power workstations are expensive and are generally comprehensive computer systems in their own right – they are usually capable of running a multitasking operating system like UNIX – and are thus under-utilized if used simply for graphics display and picture manipulation.

The University Engineering Department needed a simple high performance graphics workstation/terminal. The device was required to be intelligent enough to offload the host but at the same time would not be autonomous.

Software packages are available to make IBM PC XT/AT's operate as graphics workstations with the appropriate capabilities but the performance of these systems is, however, very poor due to the lack of dedicated hardware. Tests revealed that an IBM PC AT running a CAD package called PCAD redraws vectors at 14000 pixels per second using only the Intel 80286 microprocessor CPU. The system is clearly limited by its rasterization capabilities since the rate is nowhere near the memory bandwidth limitations. Further tests using a simple TurboPascal program showed coarse vector drawing capabilities of 40000 pixels per second on the PC AT. This confirmed the first conclusion: the CPU rasterization is too slow.

A solution to the above problem is to build some hardware to improve the response of the PC-type workstation by unburdening the PC CPU and hence in some way lengthening the dedicated-hardware section of the Graphics Display Pipeline.

The system envisaged consists of a VAX 11/750, a PC XT/AT with the dedicated hardware and a high resolution (1280×1204) monitor. The VAX acts as host running an IC design package or any general CAD package. It sends a display file to the PC XT/AT. The PC XT/AT is an intelligent terminal at which

interactive manipulation takes place. Dedicated hardware is part of the intelligent terminal and helps to improve its performance.

If the rasterization could be done as fast as the pixel memory could accept pixels, an improvement would be achieved. The standard memory (DRAM) bandwidth allows a drawing rate of the order of a million pixels per second [Whitt84]. This is a respectable improvement on the existing order of ten thousand pixels per second. Hardware rasterization would allow the memory bandwidth to be fully utilized, hence realizing the two orders of magnitude improvement in speed. The design of suitable hardware is described in the chapters that follow.

The philosophy adhered to in the design was to make a system that is simple, low cost while remaining flexible and capable of high performance both in terms of speed and resolution.

Chapter 2.
Rasterization

2.1. Introduction

The overall system (figure 2.1) is a simple graphics workstation and host pair. A VAX serves as host and runs the application package. The software on the VAX interacts with software on the PC. The PC handles the display file and runs local schematic manipulation software. The PC software also controls some specialized hardware to ease graphics display.

The structure of the specialized hardware (figure 2.2) allows for a pixel memory separate from the PC system memory. This leaves PC system memory free for the display file and the local software.

2.2. Traditional rasterization

The available graphics systems or system components use a number of different approaches to rasterization hardware. Some use a graphics controller IC such as the NEC 7220, Intel 82720 or the Intel 82786. Others use a high-power general purpose microprocessor such as the MC68000. The design of a bit-slice processor with a product such as the Advanced Micro Devices Am2903 is also popular.

Specialized graphics controllers like the Intel 82720 [DePal83] are very powerful graphics hardware tools. Graphics adapter boards for personal computers often make use of such chips to offload the main microprocessor. They are capable of handling both the rasterization of a variety of graphics primitives (lines, polylines, arcs, circles) as well as handling the screen refresh. Their comprehensive capabilities, however, mean that they are not flexible; for instance the 82720 can only handle 4 megapixels

Figure 2.1: The overall system.

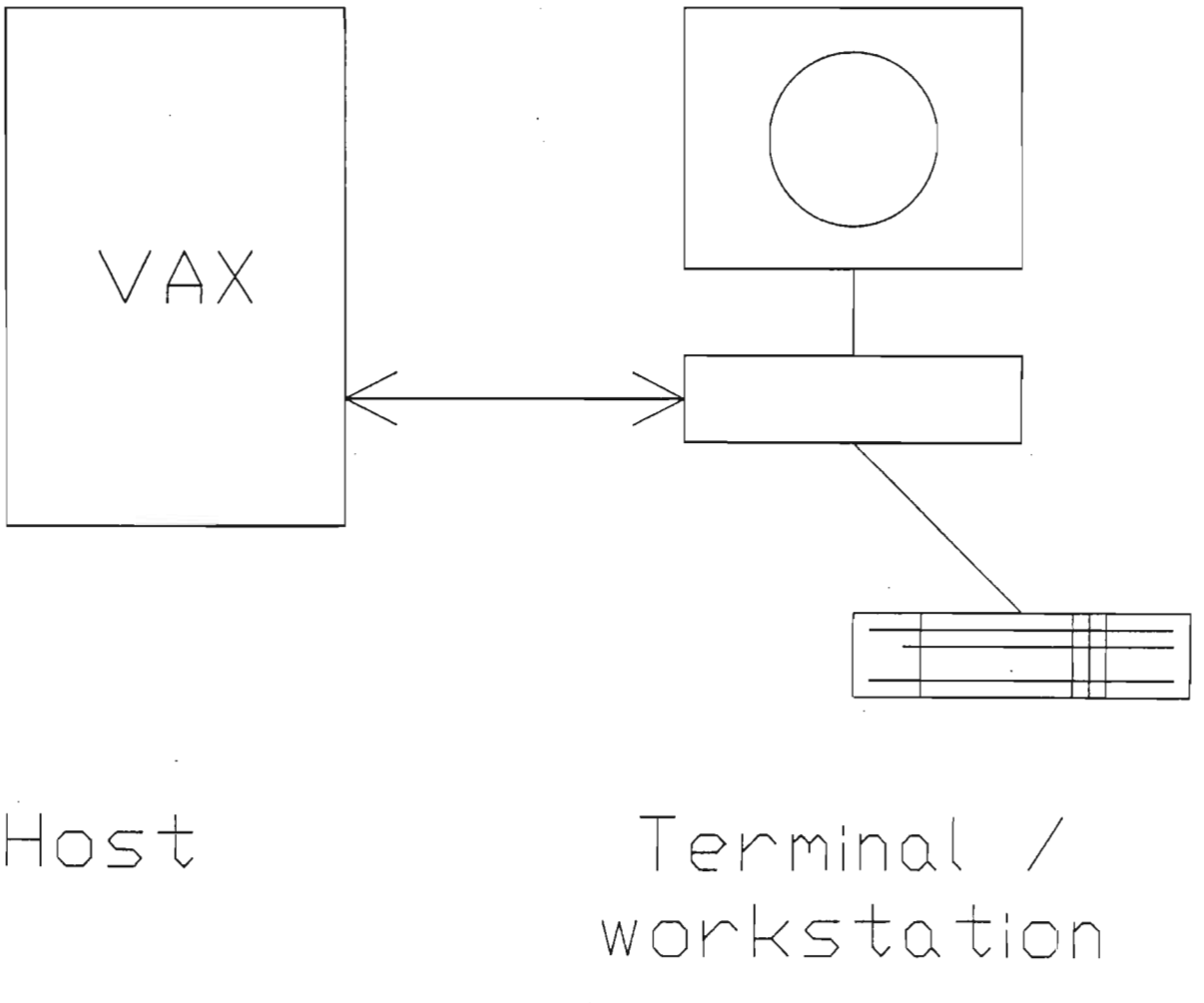
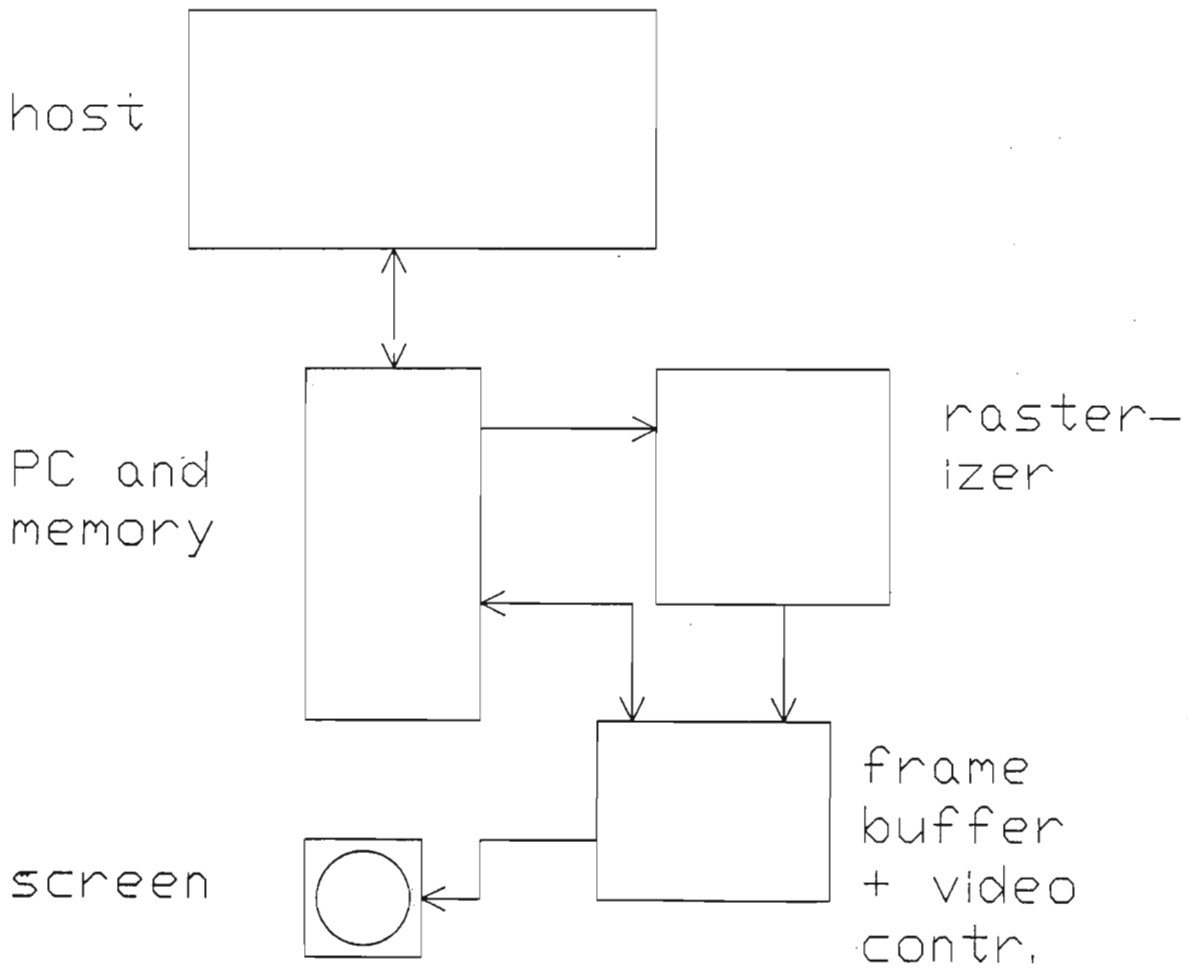


Figure 2.2: Structure of specialised hardware.



(2048×2048×1 or 512×512×16 pixels). Burst pixel rates for lines are approximately 2 megapixels per second. This means that, accounting for overheads, some pixel memory bandwidth would still be unused.

Powerful general purpose microprocessors such as the Motorola MC68000 or MC68020 are very fast (they support system clocks of up to 25 MHz) and are capable of addressing very big memory spaces (the MC68020 can directly address 4 giga-locations). They are certainly capable of nearly absorbing the memory bandwidth, however, due to IO overheads and computation overheads required to access a graphics primitive the pixel memory bus will still have unused cycles. The inner loop of a line drawing algorithm contains two or three additions and two tests; this is a relatively long loop on a general purpose microprocessor, hence only the very high clock rate relative to the rest of the system makes the high pixel drawing speed possible. The concept of using a microprocessor of much higher power than the system CPU seems inelegant to the author. High end workstations such as the Apollo DN3000 uses a MC68020 with MC68881 as its only computing engine — it runs System V Unix and handles rasterization and all other incidental tasks.

Workstations such as the Silicon Graphics inc IRIS and Lexidata's Lex 90/35 use 2900 family bit slice microprocessors to do the rasterization [Myers84]. These microprocessors are very fast and may be microcoded to do this special job more efficiently than the high power general purpose microprocessor. Taking into account the high development cost of bit slice systems rules this technology out for low cost systems.

As was indicated none of the above solutions were deemed ideal either because they would not provide enough flexibility, or an unnecessarily powerful machine was involved. In all cases there was no guarantee of full use of available memory bandwidth. It, however, remained a primary constraint of the design that a gate array solution be found and hence other avenues were persued.

2.3. Parallel rasterization

One way to ensure that all available memory bandwidth is used, is to allow multiple rasterizers to access the pixel memory.

This general idea was developed according to the schematic in figure 2.3. The system requires a DMA controller subsystem from which the bank of rasterizers may request work. The DMA subsystem must also incorporate a priority encoding scheme. The rasterizers in the bank must request permission use the pixel memory bus each time they have a pixel to write, hence, a priority encoded bus arbitration subsystem is also required on the pixel memory side of the rasterizer bank.

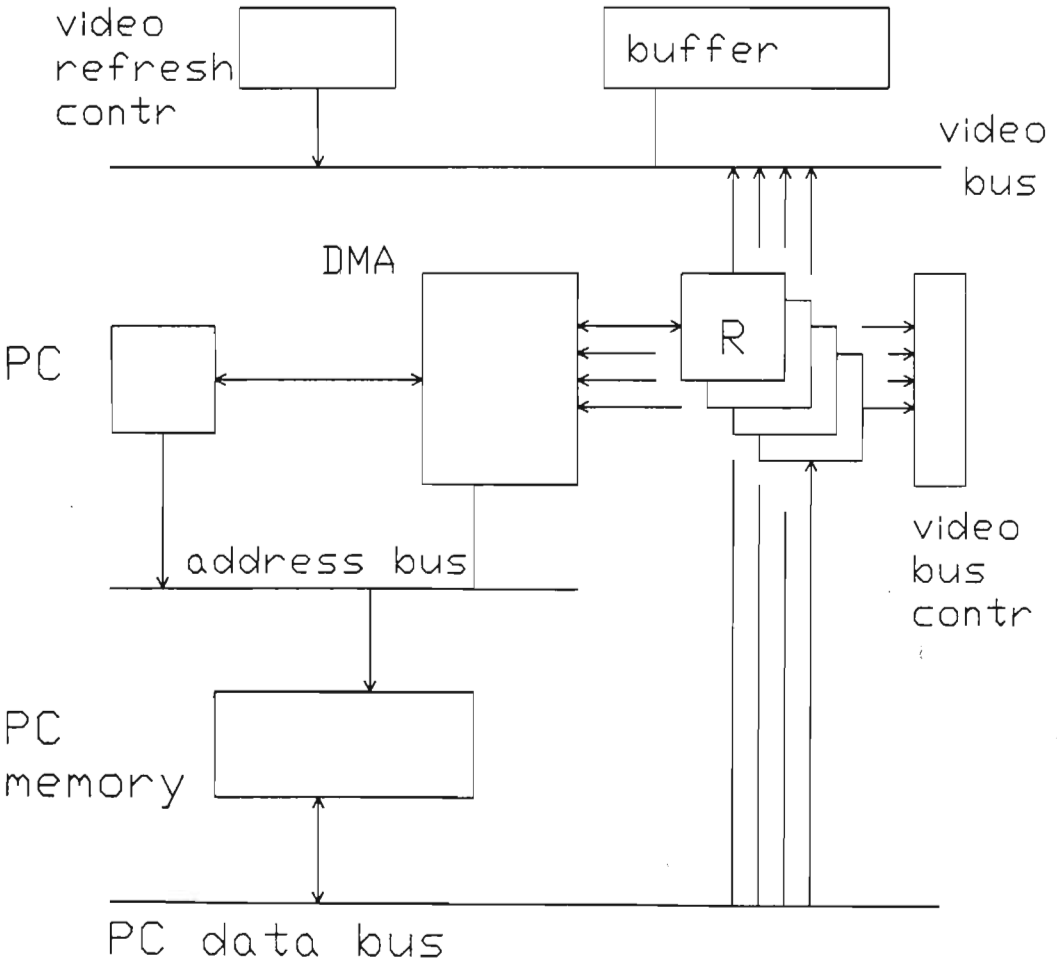
More specifically a parallel rasterizer operates as follows:

It puts out a DMA type request. As soon as it is able, the DMA subsystem transfers a fixed number of bytes to the rasterizer, strobing them in sequentially. The DMA subsystem is then free to service another request from a different rasterizer, while the first rasterizer starts work on the primitive that it has received. When any rasterizer has a pixel to write into the pixel memory it must issue a write request to the pixel bus controller and wait until the pixel is strobed into the appropriate location in the pixel memory.

The software on the PC manages the queues for the DMA controllers that supply the rasterizer bank with appropriate primitives. The screen refresh hardware and pixel bus controller unit operation is independent of the PC operation. The pixel memory bandwidth is shared between them with the aid of the arbitration unit.

The whole concept is a type of RISC environment for graphics rasterization. A rasterizer may be very specialized, for instance it may only be able to handle one type of primitive such as a line or a circle, but it would be able to do its specific task efficiently. Different types of rasterizers may then be put in the bank to

Figure 2.3: Parallel rasterization.



expand the complement of capabilities.

By its very nature the system is flexible. It is also expandable: the pixel memory has a 32 bit address bus which allows for over 4 billion (10^9) pixels.

Chapter 3.
Rasterizer implementation details

3.1. Introduction

The parallel rasterizer described above must fulfil a few basic requirements. It needs signals which will enable it to interact with a DMA subsystem and with the pixel memory subsystem. It must have a cycle time corresponding to a clock rate of 5 MHz minimum in order to receive data from the DMA subsystem in the way described in section 3.2. The device must be able to receive a vector in the form of two ordered pairs and by some algorithm literally "join the dots" in an optimally smooth way; that is, rasterize the vector.

The circuit designed for this project was implemented on a gate array since that was the only semicustom technology economically available to the University of Natal. The original intention was to use the SAMES processed MEDL array but the largest array in use at the time of manufacture was too small. The Plessey CLA5000 range was the next choice mainly because the tools in use in South Africa, to design the MEDL arrays, were Plessey products.

Designing on gate arrays is very limiting. Many components that are generally available in IC design, such as ROM or PLA's, are missing. Other constructs, for example tri-state buffers, have fixed parameters which make flexible use of them impossible as will become clear later.

The rasterizer's interaction signals consist of several sets of **request/acknowledge** pairs: one for vector request, one for DMA byte-transfer and one for pixel memory bus request. The data bus on the PC side is 8 bits wide, while the separate pixel memory bus is 32 bits wide.

There are two important line drawing algorithms: the simple digital differential analyzer (simple DDA) and Bresenham's algorithm. In order to visualize how discrete line drawing is done, one must see that a general line (not necessarily a flat, forty five degree or upright line) will have a faster moving variable and a slower moving variable. In order to achieve a smooth straight line, it makes heuristic sense to increment the faster moving variable for every dot drawn on the discrete grid of a raster screen; what is left, is to decide when to increment the slower moving variable, bearing in mind that one would like the line to resemble the ideal continuous line as closely as possible. In order to do this, the simple DDA keeps an as-accurate-as-possible representation of the actual value of the slow moving variable, and rounds this to the nearest integer to plot each dot. Bresenham's on the other hand keeps a record of the error in the discrete line with reference to the ideal line. By choosing a reference point on the grid well, Bresenham's is able to disregard the magnitude of this error and use the sign alone to tell when to increment the slow moving variable. Bresenham's algorithm has the advantage of not needing to do a division but the simple DDA, surprisingly, uses less hardware. Simple DDA was chosen for this design.

The arithmetic architecture (data structure) consists of a register set of six sixteen bit registers, a full carry-lookahead adder, tristate buffers to create the data paths and other ancillary hardware configured in a straightforward fashion, requiring one clock. See figure 3.6 and 3.7.

The control block is a finite state machine of the Moore configuration [Lewin85]. It has a state register of six bits and has 13 inputs, 37 outputs and 35 states (see figures 3.10 to 3.12 for the full state diagram).

3.2. External operation

The external signal schematic is shown in figure 3.1. The PC interaction is achieved with two pairs of signals. The request signal (**breq**) requests data (a number of bytes representing a graphics primitive) from a DMA subsystem which responds with an acknowledge signal (**back**); following the **back** the rasterizer will assert its read-ready signal (**rdr**); while **rdr** is true, bytes may be strobed in by the falling edge of the PC-side memory-read signal (**memr**). Appendix 2 has details of DMA controller timing considerations. The **reset** signal is an asynchronous signal that will reset the control machine, causing the rasterizer to return to requesting another primitive.

The pixel memory interface is in the form of a bus request (**BR**) which is issued when the pixel address is valid; the bus grant (**BG**) signal acknowledges the pixel and allows the rasterizer to continue.

The data format (figure 3.2) is a set of up to sixteen bytes beginning with two colour bytes followed, in the case of the line rasterizer, by eight bytes defining the start and end points of the line. The first implementation does not include the colour handling hardware.

3.3. Line drawing algorithms

Assume that a line is described by two ordered pairs (x_1,y_1), (x_2,y_2) and Δx is x_1-x_2 and Δy is y_1-y_2.

Newman and Sproull [Newma79] who give a good account of line drawing techniques, forward the following ideals for computer generated lines:

1. Lines should appear straight. This requires some attention on a screen with a finite number of addressable elements.

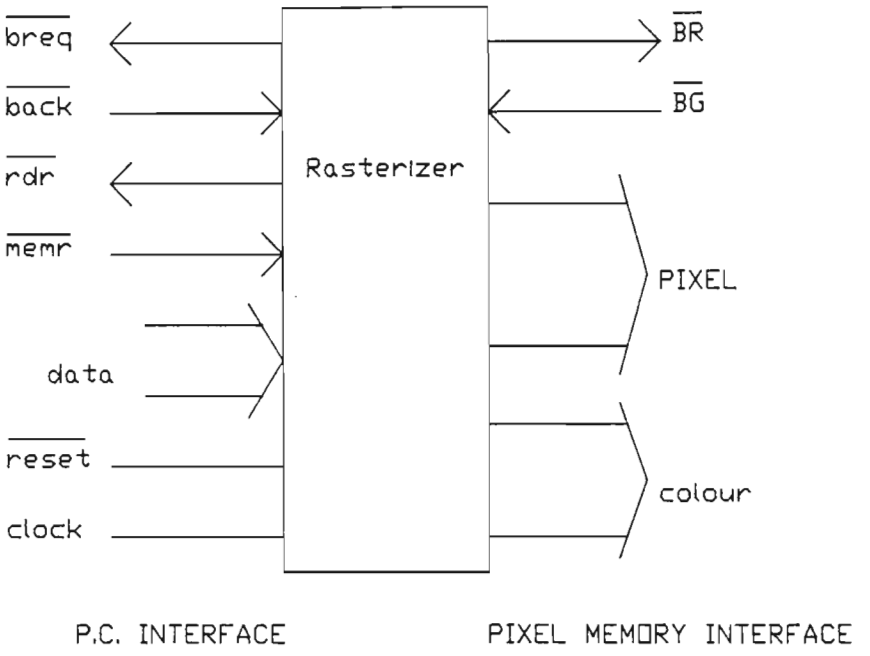


Figure 3.1: External signal schematic.

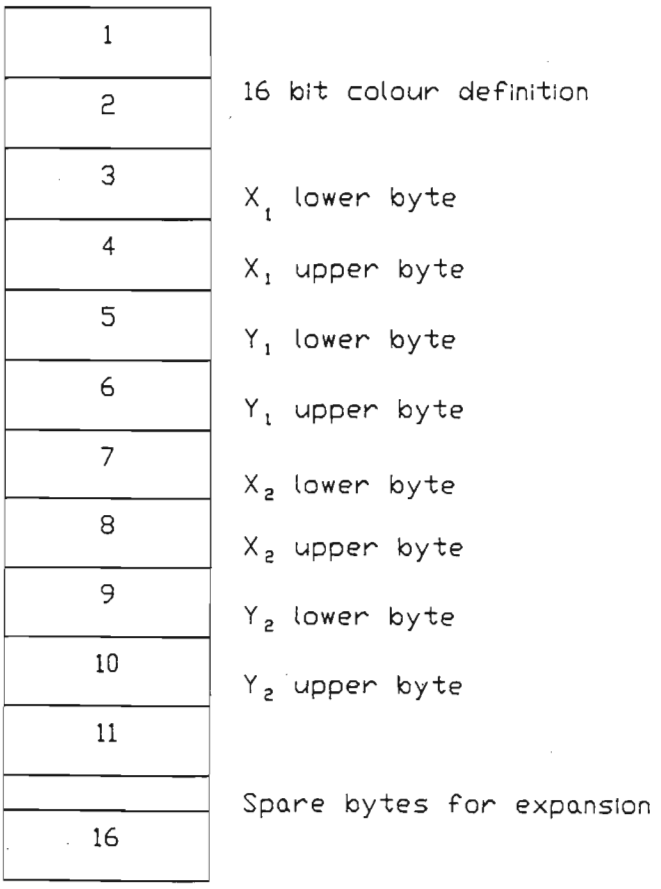


Figure 3.2: Data format.

2. Lines should terminate accurately. In a digital machine with limited precision, care must be taken to ensure that truncation errors do not catch one out especially near the end of a line.

3. Lines should have constant density. This is only really possible on lines of 0°, 45° or 90°. The nearest approximation to the ideal in other cases is possible, if the algorithm in use increments one variable (x or y) at every step, and the other variable at every i^{th} step where i depends on the slope of the line.

4. Line density should be independent of line length and angle. A length estimate is required here; commonly this estimate is the greater of Δx and Δy.

5. Lines should be drawn rapidly. The speed with which a line may be drawn with hardware is determined by a combination of the length of the algorithm used and the cycle time of the machine implementing the algorithm.

Line drawing algorithms are generally based on what Newman and Sproull [Newma79] call the digital differential analyzer (DDA) which generates lines from their differential equations; in the case of the straight line algorithm this is the slope. The slope of a line is simply the ratio of Δx and Δy.

Two algorithms give acceptably high quality results: the Simple DDA and Bresenham's Algorithm.

The simple DDA is based on simple numerical integration of the function describing the line. The process is represented in equations 3.1 to 3.6, where f(x) and g(y) are the said functions. Notice that δx and δy are chosen as Δx and Δy divided by the line length estimate (see equations 3.5 and 3.6), now the line length estimate in this case is the bigger of Δx and Δy, hence one of the summation terms in equation 3.2 or 3.4 is 1 and the other is the fractional slope. The result of the equation with the non-unity summation term is rounded to the resolution of the screen coordinates to give the actual pixel address for the slow moving

variable, while the pixel address for the fast moving variable is obtained by incrementing this variable at every step. Of course, there is no inaccuracy introduced by the numerical integrations since the differential equation of a straight line is a constant.

(3.1) $$y(x) = y(x_1) + \int_{x_1}^{x} f'(x)dx$$

(3.2) $$y(x) \approx y(x_1) + \sum_{i} f'(x_i)\,\delta x$$

(3.3) $$x(y) = x(y_1) + \int_{y_1}^{y} g'(y)dy$$

(3.4) $$x(y) \approx x(y_1) + \sum_{i} g'(x_i)\,\delta y$$

(3.5)
$$\delta x = \Delta x/\text{line-length}$$
$$x_i = x_1 + i\delta x$$

(3.6)
$$\delta y = \Delta y/\text{line-length}$$
$$y_i = y_1 + i\delta y$$

The algorithm is as follows:

```
SIMPLE DDA
begin
        length = |x2-x1|
        if |y2-y1| > length then length = |y2-y1|
        xincrement = (x2-x1) / length
        yincrement = (y2-y1) / length
        x = x1 + 0.5
        y = y1 + 0.5                          { add 0.5 to round }

        for i = 1 to length do
            begin
            output ( trunc(x), trunc(y) )
            x = x + xincrement
            y = y + yincrement
            end
end
```

Bresenham's algorithm may be viewed as a refinement of the previous method. Foley and Van Dam [Foley82] give a mathematical analysis of Bresenham's algorithm; heuristically it works as follows:

It presumes incrementation of the fast moving variable, and keeps track of the actual position of the ideal line with respect to the two possible positions of the next dot. Refer to figure 3.3 and assume the line to be drawn is in the first quadrant and Δx is bigger than Δy. The error vector **e** indicates position of the actual y-coordinate for the given x-coordinate with respect to the halfway mark between the two possible y-positions for the dot; if the vector points up, the y-variable is incremented, if it points down, the y-variable is not incremented. The error vector is calculated using much the same data as the simple DDA uses but the magnitude of the error is not of concern: one only requires the direction (or sign) of the vector. The algorithm runs as follows:

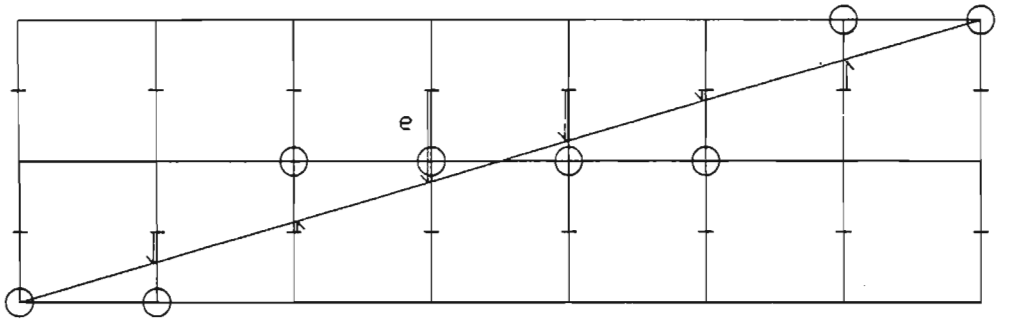


Figure 3.3: In Bresenhams algorithm the vector **e** points to the best dot location.

```
BRESENHAM'S ALGORITHM (Δx > Δy first quadrant)
begin
e = Δy/Δx - 0.5
for i = 1 to Δx do
        begin
           output ( x,y )
           if e > 0
           then
                    begin
                    y = y + 1
                    e = e + Δy/Δx - 1
                    end
           else e = e + Δy/Δx
           x = x + 1
        end
end
```

The advantage of the algorithm is that the error size is not important and hence may be scaled, giving the following result:

```
BRESENHAM'S ALGORITHM (Δx > Δy first quadrant)
begin
e = 2Δy - Δx
for i = 1 to Δx do
        begin
           output ( x,y )
           if e > 0
           then
                    begin
                    y = y + 1
                    e = e + 2Δy - 2Δx
                    end
           else e = e + 2Δy
           x = x + 1
        end
end
```

The division, $\Delta y / \Delta x$, is avoided.

Bresenham's Algorithm has from 2 to 3 additions per cycle in its fastest form which, as will be clearer later, is the same speed as the simple DDA after the sixteen cycles of the division required by the simple DDA. It also requires storage space for two constants and the variable e apart from the vector, whereas the simple DDA requires storage of one less constant. More important is that Bresenham requires the maintenance of 17 bit accuracy throughout while the simple DDA only requires this accuracy for the division (and even that may be hidden with a simple multiplexing trick). The importance of the smaller bus lies in the size of the gate array routing channel. The target array had a 16 track channel and the use of wider buses severely limited utilization and hence wasted real estate. It is evident that for a small loss of performance (significant only in small vectors), the simple DDA may be implemented on much less Silicon. This was important in the physical implementation due to a limitation on the size of semicustom IC available.

3.4. The global algorithm

A general algorithm for drawing a primitive (line, circle, polygon, etc) is described by figure 3.4. Firstly, a request for a primitive is tendered and when it is granted, a set of bytes is loaded. Next, the primitive is analyzed. Analysis in the case of a line, which is really presented in the form of a vector, involves determining direction and slope, and storing that information. The first pixel is often available at this point since it is often part of the description of the primitive, especially in the case of a line. The output state is thus the first part of the inner loop, followed by the end-of-primitive test and then the next-pixel computation.

The loading of the primitive (figure 3.5) is simply achieved by putting out a read ready (**rdr**) signal and waiting for the data to be strobed in by the memory read (**memr**) signal. The only assumption necessary is that the rasterizer can accept a byte and be ready for another before the DMA subsystem can write another. If this is not possible a slower two-state full handshake is required. See appendix 2 for DMA timing details.

The full details of the analysis and computation algorithms, and the hardware are very interrelated. Section 3.6. on the Control Machine contains the algorithm details. The general outline of the algorithms follow.

The analysis must identify the class and direction of a line. The two classes of line are, "manhattan", and all other lines. Manhattan lines are lines of 0° 45° and 90° or multiples thereof. It is worth filtering out these cases since, if treated specially, manhattan lines may be drawn with one or two additions in the inner loop as opposed to two to three for general lines. In the case of a line of general slope it is necessary to identify whether its slope is greater or less than 45°. All this data may be gathered as follows:

1. Compute and record delta Y. Record whether it is positive, negative or zero.

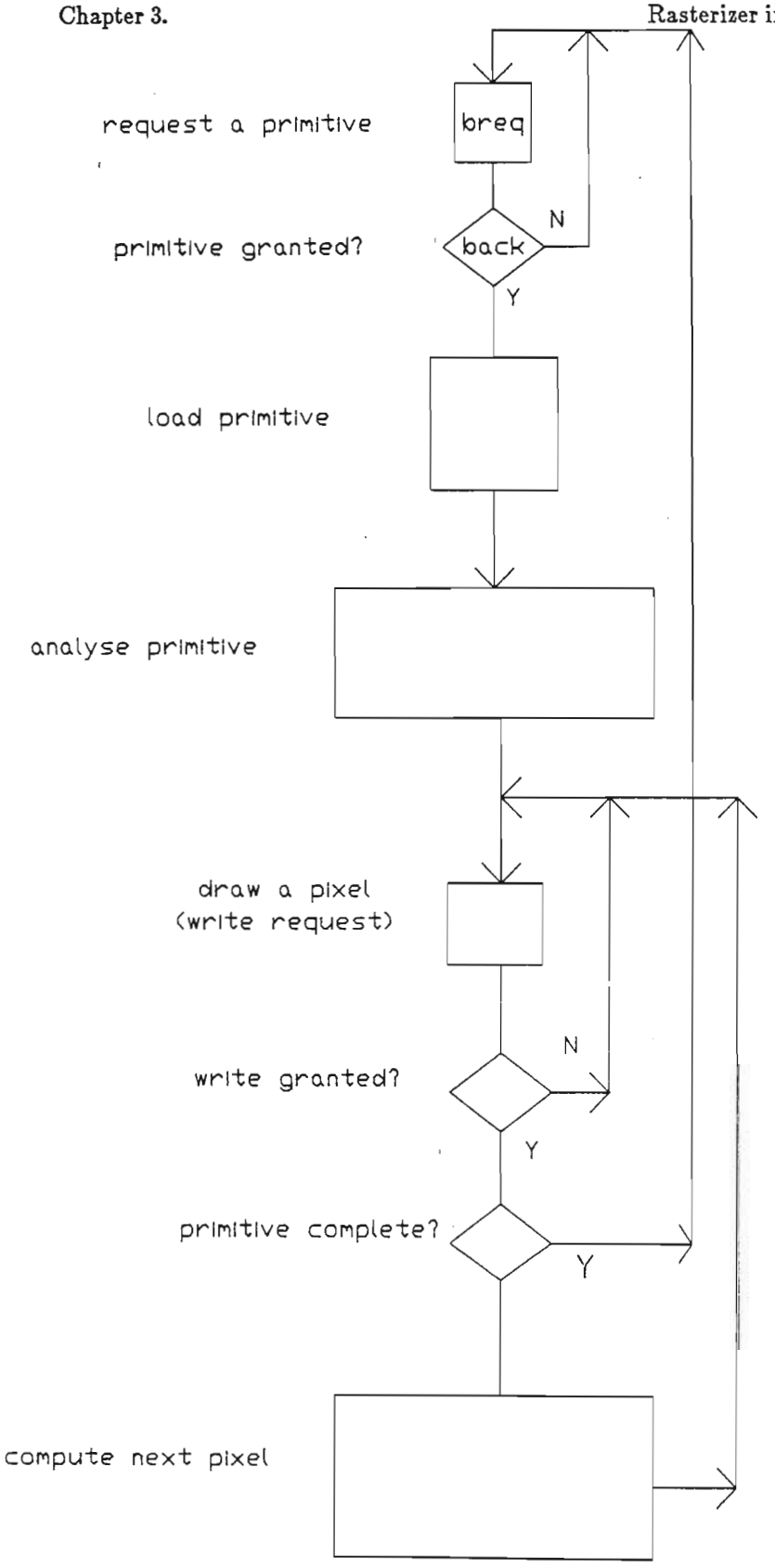


Figure 3.4: Primitive-Drawing algorithm.

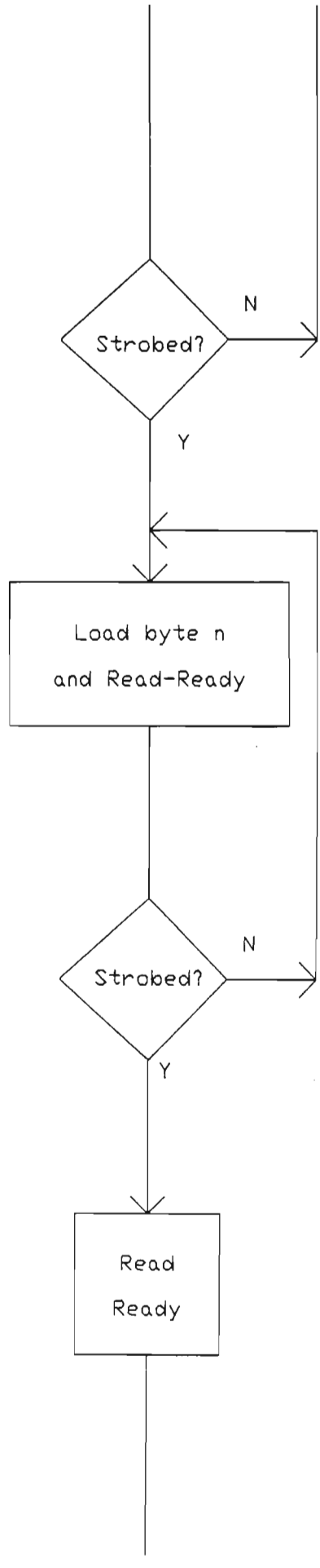


Figure 3.5: Loading algorithm.

2. Do the same with delta X.
3. Compute (delta X - delta Y) and record its sign or if it is zero.
4. If at this stage the line is not manhattan, divide the smaller of delta Y and delta X by the bigger and record the resulting fraction.

If the result of any of tests 1. 2. or 3. are zero, the line is manhattan and the divide may be skipped.

The section of the algorithm that computes the next pixel (the drawing section) must be divided in two to deal with the manhattan lines and the non-manhattan lines separately. Generally one of the ordered pairs may be incremented until it is equal to the other ordered pair. The way in which the components of the pair are incremented determines the shape of the line. The aim is to achieve the smoothest straight line between the two points.

The manhattan case simply calls for unconditional increment of the appropriate variable (either the X or the Y or both). In the non-manhattan case, one variable is incremented every cycle while the other is incremented with a frequency determined by the slope. The issue of when to increment this variable involves the resolution of the arithmetic machine. This is dealt with next.

A pixel address consists of two 16 bit integers, hence 16 bit registers and arithmetic hardware are required. In the simple DDA used, the slope fraction is accumulated and the infrequently incremented variable adjusted by the rounded value in the accumulator. Note that only the fractional value of the accumulator need be kept if the infrequently incremented variable is adjusted on overflow of the fraction. The slope fraction may be represented in 16 bits and hence has an uncertainty in 2^{-17}, thus even if a line the full length of the screen is drawn the error will accumulate to 2^{-1} which is less than 1; this means that no pixel will be misplaced if true rounding is performed. True rounding may be achieved by adding $1/2$ to the accumulator at the

beginning of the line drawing process.

The drawing section of the algorithm basically determines the drawing speed of the rasterizer. In the case of a 0° or 90° line the drawing algorithm only needs two states: an output state and a variable-increment state. The dot rate would hence be half the clock rate. A 45° line would need three states: one extra variable-increment state. Any general line would need between three and four states: an output state, an adjust/test state and one or two increment states. Clearly the worst dot rate is near a quarter of the clock rate.

3.5. The arithmetic architecture (data structure)

The arithmetic machine used to implement the simple DDA is shown in figure 3.6. (A more detailed circuit diagram is to be found in appendix 1.) Much of the functionality of the machine is useful for both of the two major tasks: division and incrementing/decrementing.

The heart of the machine is the full (two level) carry-lookahead adder. The adder has the ability to add and subtract (in both senses) two sixteen bit numbers (see appendix 5 for full details).

The machine has six sixteen bit registers, four to store the vector and two accumulators (a and b) which are ultimately used to store the fractional slope and the fractional part of the slow moving variable. The multiplexer and the shift register are used only for division. The data paths are controlled with the enable signals on the tristate buffers. The so called "extra flip-flops", or extra-bits, store information gleaned by the analysis algorithm. Figure 3.7(a)–(d) contains ancillary circuits, namely, a sixteen bit counter used by the division algorithm, a circuit to compare (x_1,y_1) and (x_2,y_2), the adder-result-zero testing hardware and signal synchronizing circuit. The data input latching circuit is to be found in figure 3.9.

The overall operation is as follows:

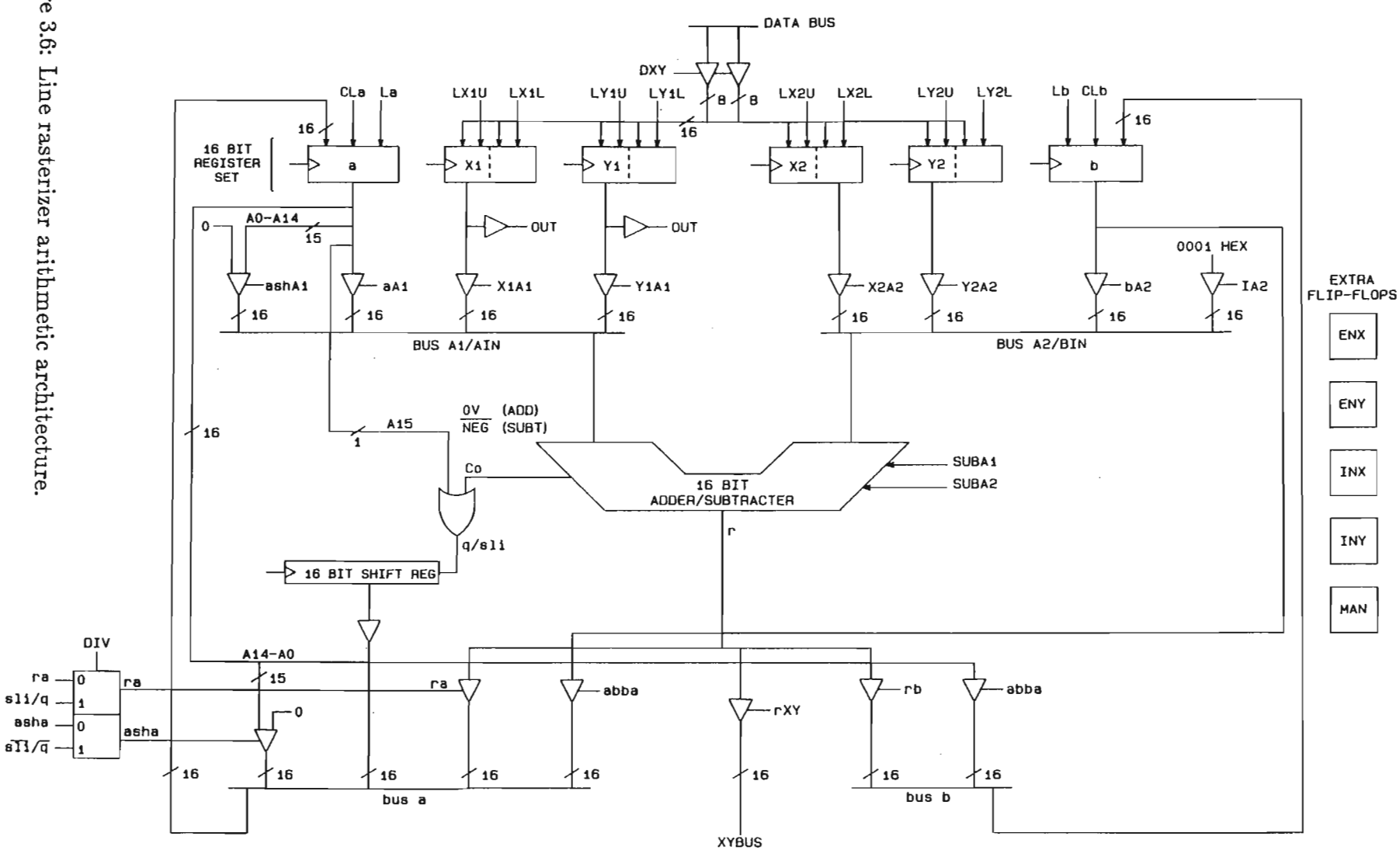


Figure 3.6: Line rasterizer arithmetic architecture.

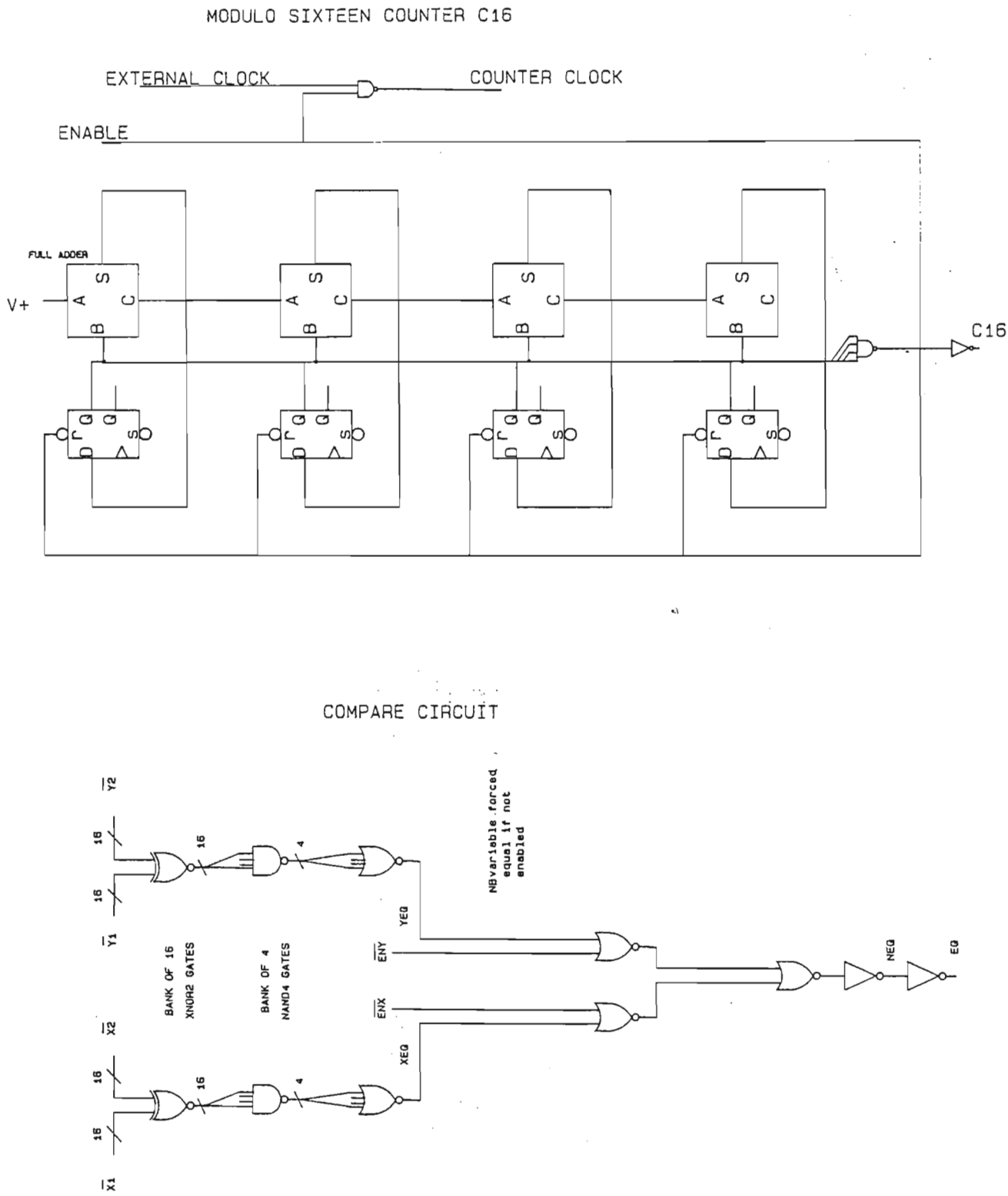


Figure 3.7(a): Modulo sixteen counter. (b): Circuit to compare (X1,Y1) and (X2,Y2).

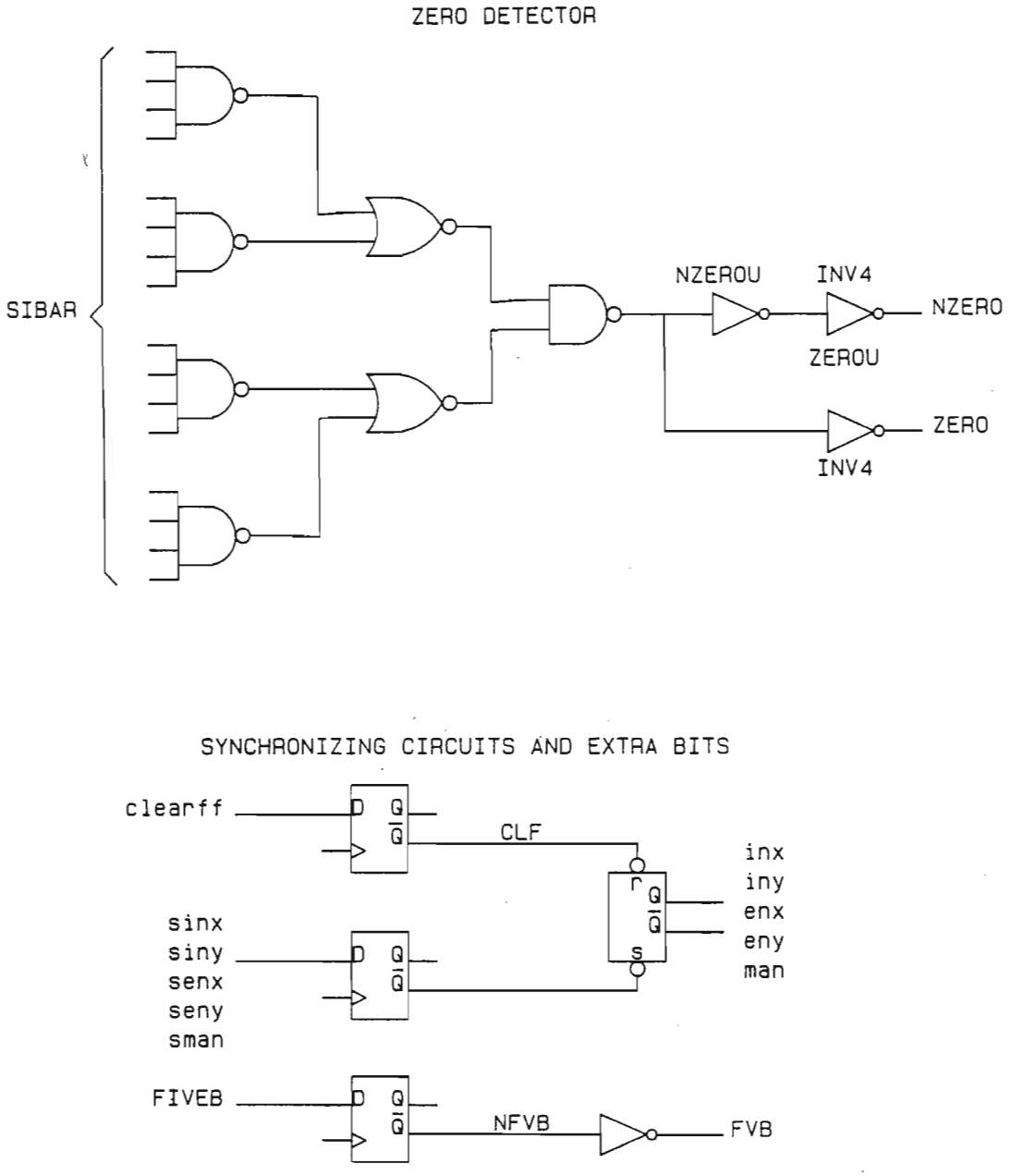


Figure 3.7(c): Adder-result-zero-test hardware. (d): Signal synchronizing circuit.

(Refer to figures 3.6 and 3.7 (above) and table 3.1 (below) for the meanings of signals refered to in the text.)
The vector, in the form of two ordered pairs (x_1,y_1) and (x_2,y_2), is loaded into registers X1, Y1, X2, Y2. Using that data and the scratch registers a and b, the line is analyzed. Next, the registers X1 and Y1 become variables x and y and are respectively incremented or decremented until they match X2 and Y2.

The registers all preserve their data unless they are loaded specifically with their load signal, for example **La** loads the a-register with the value on bus-a if **La** is asserted on a clock transition. The X and Y registers are byte loadable to deal with the byte-format input data. The b-register may have its most significant bit set, and all other bits cleared (using the set and reset lines on the internal flip-flops); this is to set the register value to 0.5 for the drawing algorithm.

The option to use an adder/subtractor that can perform a+b, a−b and b−a saves on the number of data paths and hence makes the buses smaller (in terms of number of bus users, and hence physical size); the machine consequently has a faster cycle. If this feature were not available, register contents would have to be swopable which not only adds data paths but adds cycles to any algorithm. Vectors may be drawn in any direction using the hardwired logic-1 on bus A2 to increment or decrement X1 and Y1. The adder is a full 16 bit carry lookahead adder based on an implementation in Hwang [Hwang79]; full internal circuit representations are available in appendix 5. Subtraction from either input is achieved by routing both inputs through banks of exclusive-or gates.

The analysis of the line extracts information and data necessary for the subsequent drawing of the line. It must determine the direction and slope of the line and set the appropriate extra-bits to store this information. Δx and Δy are byproducts of this process. The extra bits are: MAN, which indicates that the line is manhattan. MAN is used in conjunction with ENX and ENY which flag that a variable should be unconditionally incremented

or decremented. ENX and ENY are also used to flag the fast moving variable in the drawing of a general line. INX and INY, if set, show that their variable should be incremented, not decremented, and vice versa. At this point full use is made of the double ended subtractor to get the positive versions of Δx and Δy in the correct places, for instance if x_1-x_2 is negative then x_2-x_1 is calculated and stored in the a-register. Special hardware produces a ZERO signal. Instead of recalculating Δx and Δy when they are found to be in the wrong places for the divide phase, a data path is provided to swop the contents of the registers in a single cycle. This extra data path simplifies the control logic considerably.

The hardware compare circuit saves an addition in the inner loop of the drawing algorithm and also saves control logic complexity.

The division is accomplished with a restoring successive subtraction technique based on Hwang section 7.3 and 7.4 [Hwang79]. It produces a 16 bit fractional result from the division of one sixteen bit integer by a bigger sixteen bit integer; the radix point lies to the left of the most significant bit in the result. The multiplexer controlled by the signal DIV converts the machine into a form that allows the operation a/b to be performed in sixteen cycles. The divide is done without the intervention of the control logic apart from, for sixteen cycles; directing the a-register, shifted left by one, to the A1 adder input; directing the b-register to the A2 adder input; and loading the a-register. The effective configuration of the hardware during the divide is shown in figure 3.8. The sixteen cycles are timed by a special enablable counter: C16 (see figure 3.7).

Conventional restoring binary division is based on the following equation, after Hwang [Hwang79]

$$R^{j+1} = 2R^j - q_{j+1} \times D \tag{3.7}$$

in which

D is the divisor
$j = 0,1,...,n-1$
n is the word length of the quotient
q_{j+1} is the $(j+1)^{th}$ quotient
$2R^j$ is the partial dividend for determination of q_{j+1}
R^{j+1} is the partial remainder after determination of q_{j+1}
R^0 is the dividend
R^n is the remainder

now the q_{j+1} is chosen as

$$q_{j+1} = 0, \text{ if } 2R^j < D \tag{3.8}$$
$$= 1, \text{ if } 2R^j \geq D$$

Clearly the execution of equation 3.7 with q_{j+1}=1 performs the test in equation 3.8 and if the result is negative, ie $2R^j < D$, R^{j+1} must be reconstructed by adding D (hence the term restoring division).

Referring to figure 3.8, the above algorithm may be implemented as follows:
The positive 16 bit integer dividend, the smaller of Δx and Δy, is stored in the a-register and the positive 16 bit integer divisor, the bigger of Δx and Δy, is stored in the b-register. In implementing equation 3.7 with $q_{j+1} = 1$, a positive 16 bit number is subtracted from a positive 17 bit number, yielding a 16 bit positive result that must be saved or, a negative result that may be discarded. The positive result is certainly 16 bit since the divisor starts off smaller than the 16 bit dividend and is only ever multiplied by two. If the result is negative then $2R^j$ certainly has a zero most significant bit (bit a_{16} of a_0 to a_{16} is zero) and so $2R^j$ will actually fit into 16 bits. The last two points show that the a-register need only be 16 bits wide. The sign of the subtraction (result bit 17) is determined by the OR gate, after the following reasoning:
Presume that vector $A = (A_{17},A_{16},...,A_0)$ = the a-register arithmetic left shifted by one, with A_{17} as the phantom sign bit, and that the vector B is the sign extended two's complement of

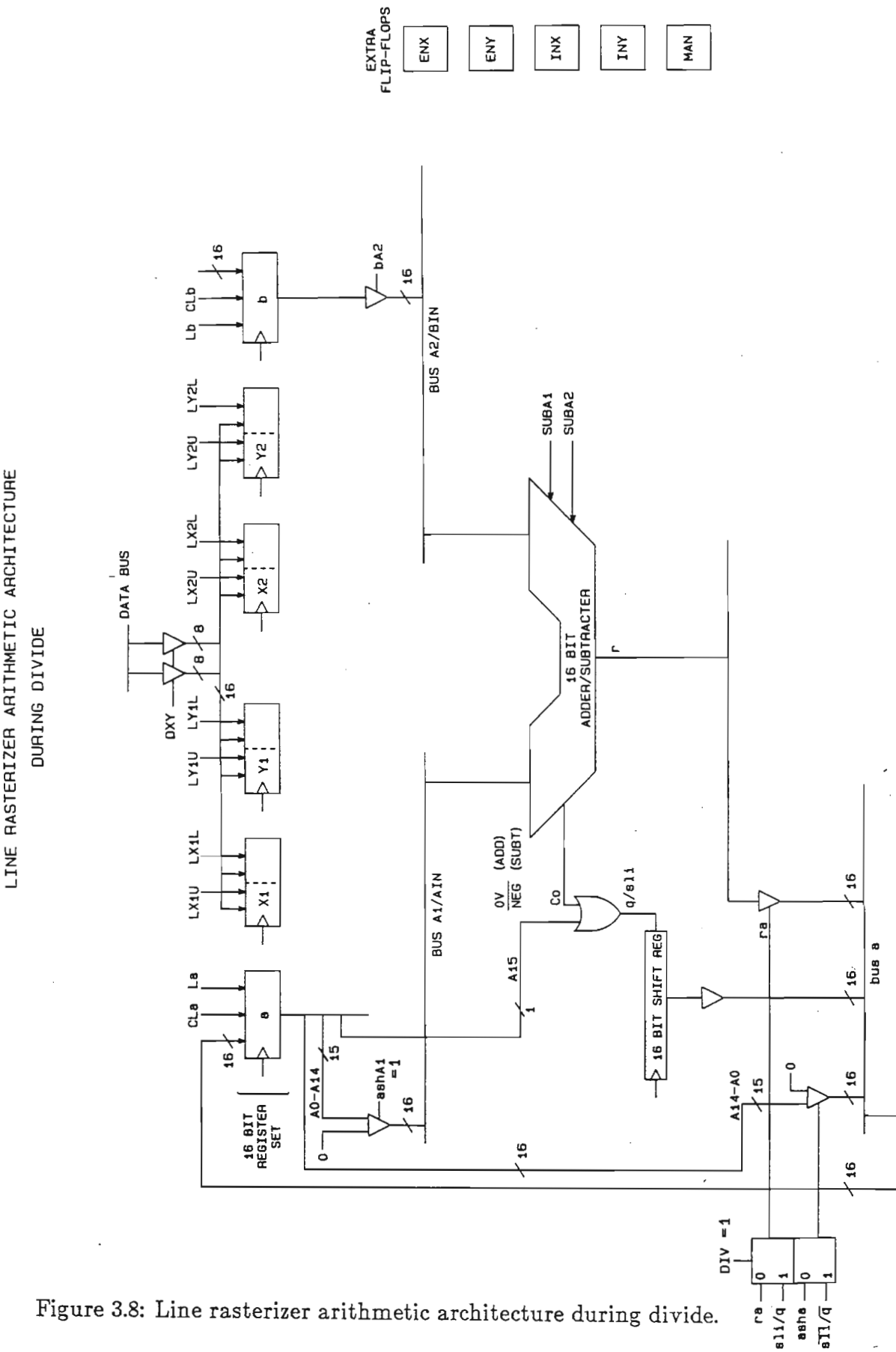


Figure 3.8: Line rasterizer arithmetic architecture during divide.

the b-register and S is the sum vector).

Now the sum sign bit S_{17} is required, and always, A_{17}=0 (positive 17 bit number) and B_{17}=1 (negative 16 bit number, sign extended) and:

(3.9) $$S_{17} = A_{17} \oplus B_{17} \oplus C_{17} = 0 \oplus 1 \oplus C_{17} = \overline{C}_{17}$$

where C_{17} is the carry in at this point and is equal to

(3.10) $$C_{17} = A_{16}.B_{16} + A_{16}.C_{16} + B_{16}.C_{16}$$

now $B_{16} = 1$ since B is a sign extended 16 bit negative value, so

(3.11) $$C_{17} = A_{16} + C_{16} = \overline{S}_{17} \qquad = q_{j+1}$$

for the j^{th} cycle.

The quotient bit is shifted left into the shift-register. The $2R^j$ is rescued automatically by selection of **asha** (a shifted to a) on bus-a when the quotient bit is a one. The result of the subtraction is put on bus-a by automatic selection of **ra** (result to a) when the quotient bit is a zero. Directly after the sixteen cycles of the divide, the a-register is loaded with the contents of the shift register, the quotient.

The input latching system is shown in figure 3.9. The register latches the data on the positive edge of **xmemr**. A positive edge-catch circuit synchronizes **xmemr**.

3.6. The control block and state diagram.

The control block consists of a finite state machine. A microprogrammed architecture was not used because of practical implementation difficulties; essentially, a standard-cell or full custom system is required for a microprogrammed machine because of the need for on-IC ROM.

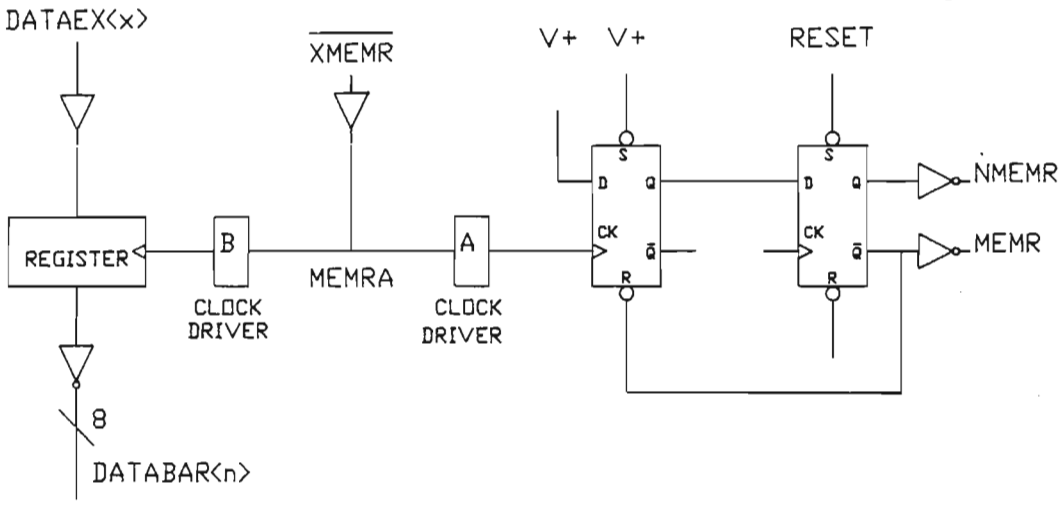


Figure 3.9: Input latching system.

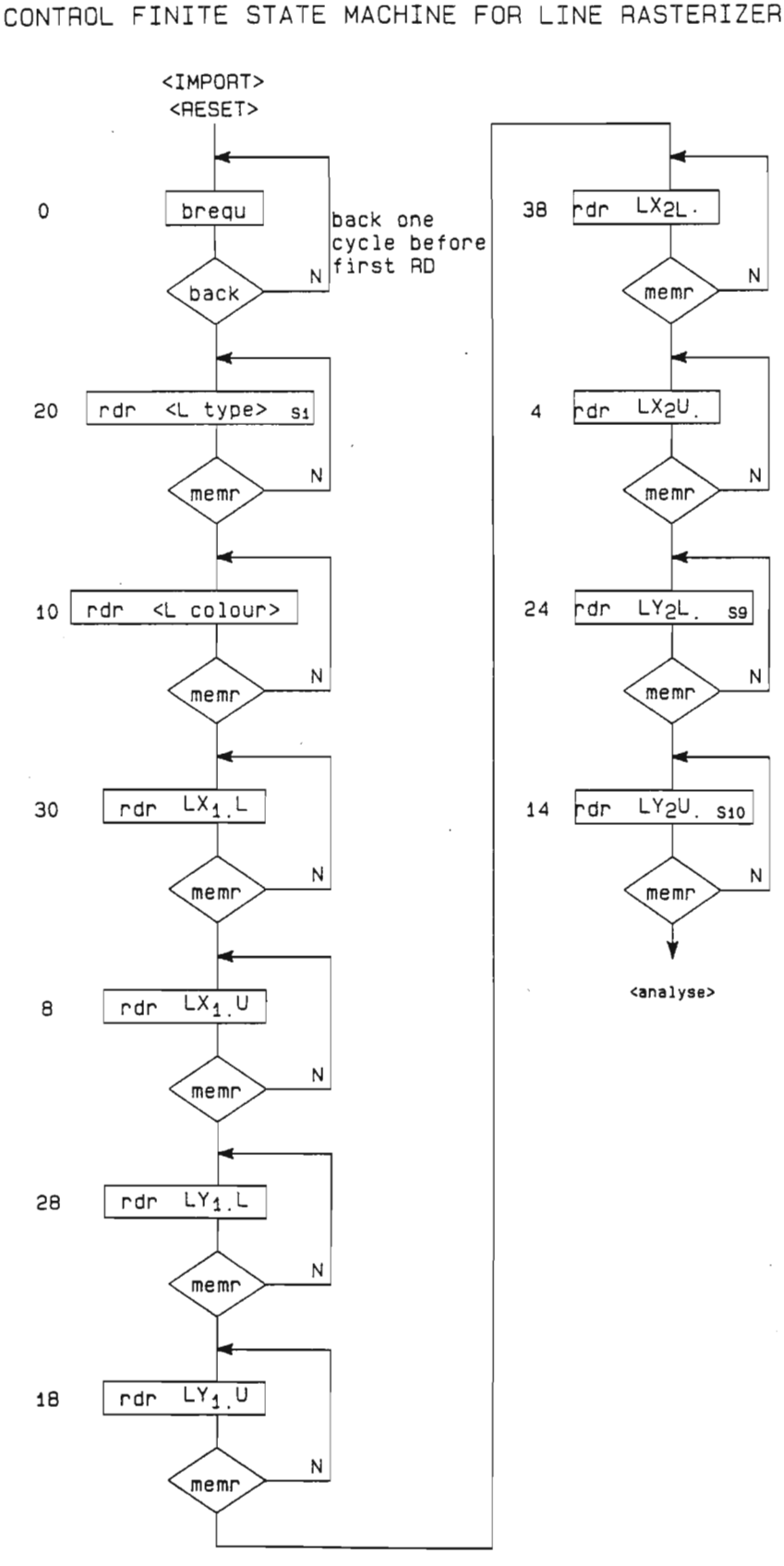


Figure 3.10: State diagram for line rasterizer (1/3).

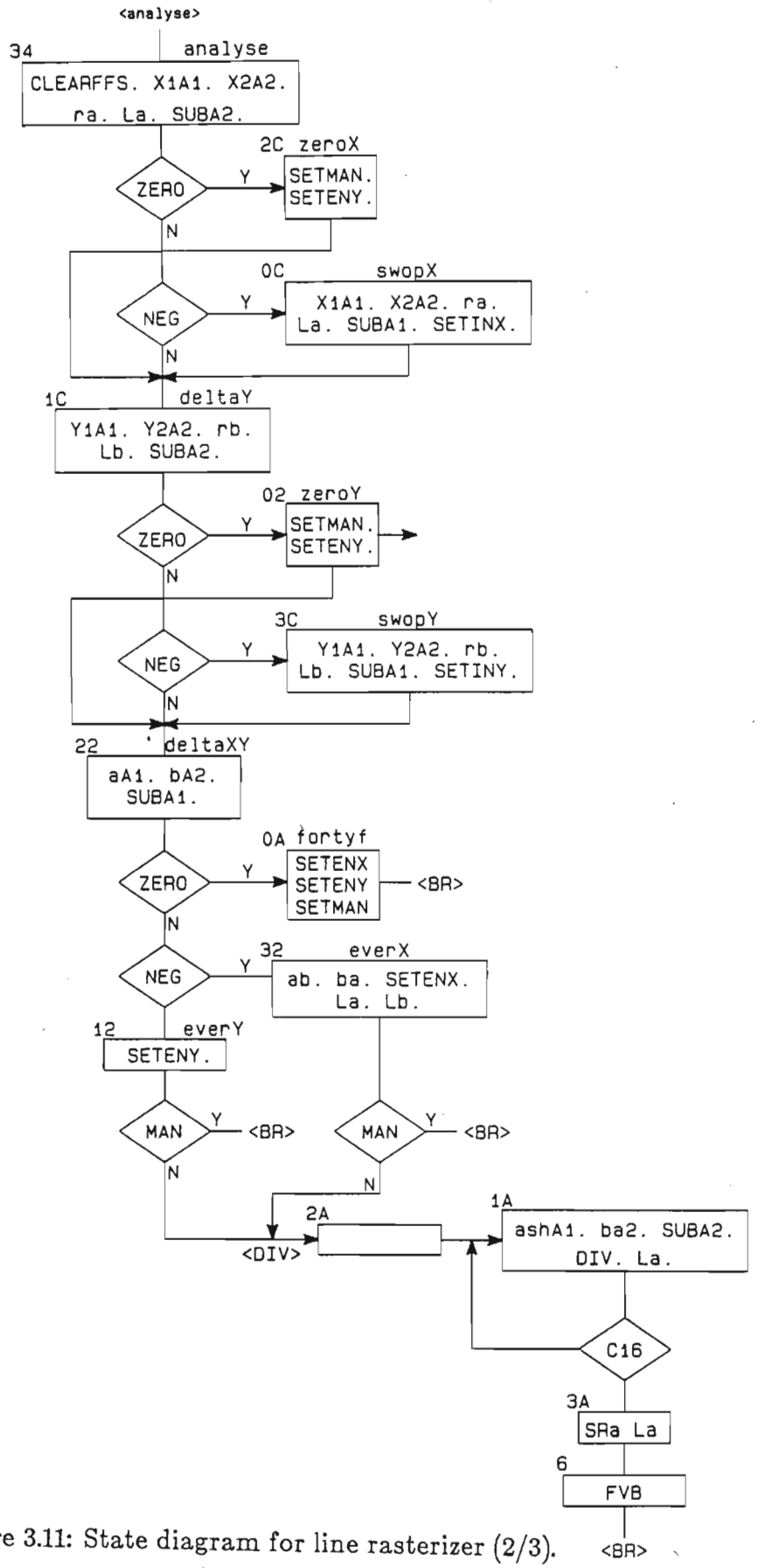


Figure 3.11: State diagram for line rasterizer (2/3).

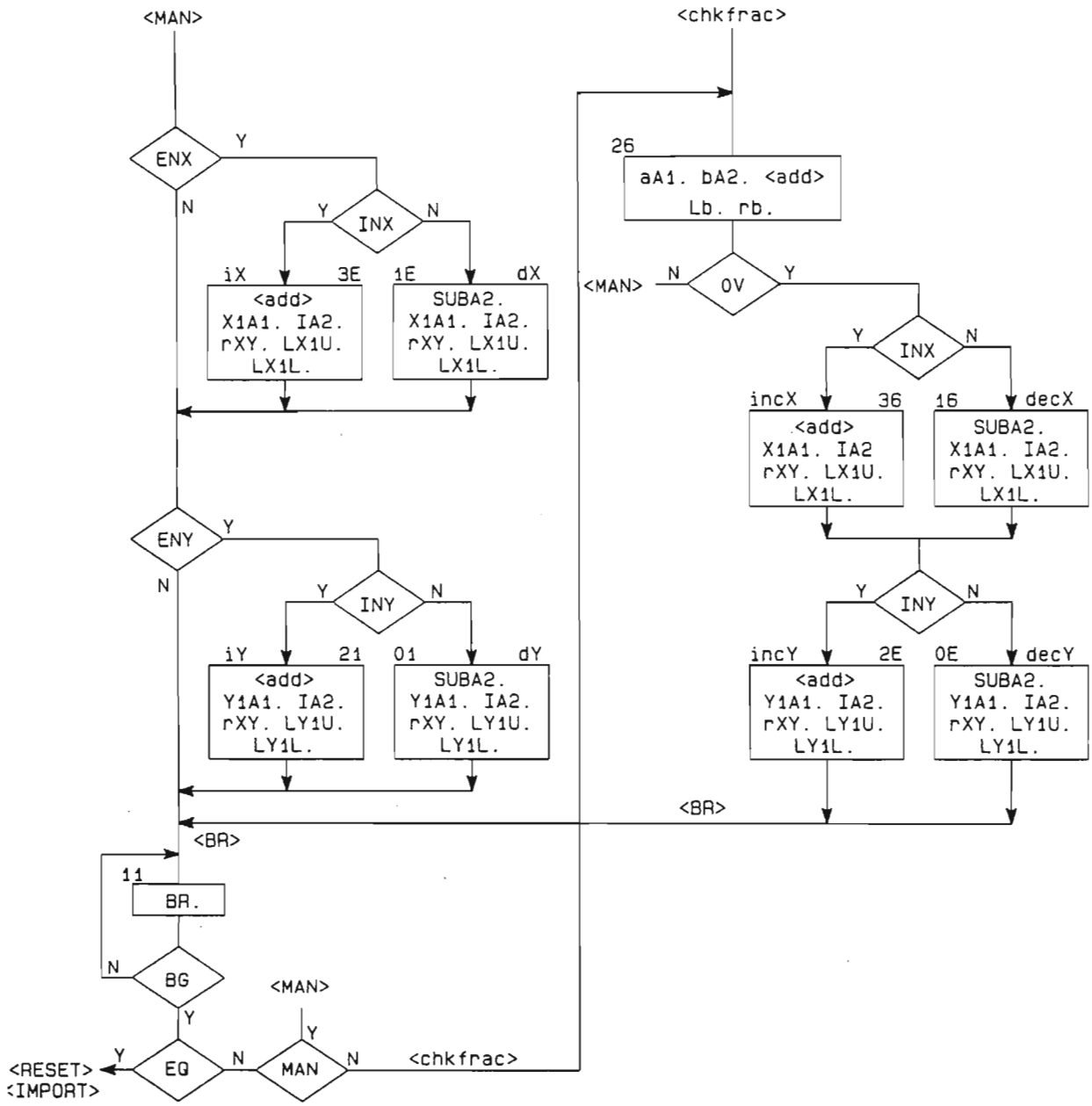


Figure 3.12: State diagram for line rasterizer (3/3).

The state diagram is shown in figures 3.10 through 3.12. The diagram conforms to the Moore model in which the outputs depend only on the present state of the machine [Lewin85]. The Moore machine is slower in general than the Mealy machine [Lewin85], which allows conditional outputs, that is, outputs that are dependent on the present state and the inputs. However, for a big state machine the Moore machine is simpler since timing considerations are less complex and there is software available that will help in the design of Moore machines. The general structure of the Moore machine and the Mealy machine are shown in figure 3.13.

The functionality of the state machine as defined in the state diagram has largely been described in connection with the hardware, only details are new; for instance consideration must be given as to when to load registers and what data to select for each bus. Definitions of all the input and output terms are listed in table 3.1; there are 13 inputs and 37 outputs and 35 states; the state vector has six bits.

Table 3.1 List of finite state machine inputs and outputs.

OUTPUTS

register load...

LX1L	load lower byte of X1
LX1U	load upper byte of X1
LX2L	
LX2U	
LY1L	
LY1U	
LY2L	
LY2U	
La	load a register
Lb	

data route...

ashA1	a register shifted left to bus A1
aA1	a register to bus A1
X1A1	register X1 to bus A1
Y1A1	register Y1 to bus A1
X2A2	register X2 to bus A2
Y2A2	register Y2 to bus A2
bA2	b register to bus A2
IA2	0001H to bus A2
ra	adder result to bus a
asha	a register left shifted to bus a *
SRa	shift register to bus a
abba	a register to bus b and b register to bus a
rb	adder result to bus b
rXY	adder result to XY bus

general...

breq	bus request to the DMA subsystem
BR	bus request to the pixel memory subsystem
rdr	primitive read ready
clearff	clear the extra bits
FVB	set the msb of b register and clear the rest
DIV	select on divider setup multiplexer
SUBA1	two's complement A1 adder input
SUBA2	two's complement A2 adder input
sinx	set INX bit
siny	set INY bit
senx	set ENX bit
seny	set ENY bit
sman	set MAN bit

INPUTS

back	DMA subsystem bus acknowledge
memr	input strobe
BG	pixel memory subsystem bus grant
ZERO	adder result zero
NEG	adder result negative
OV	adder overflow

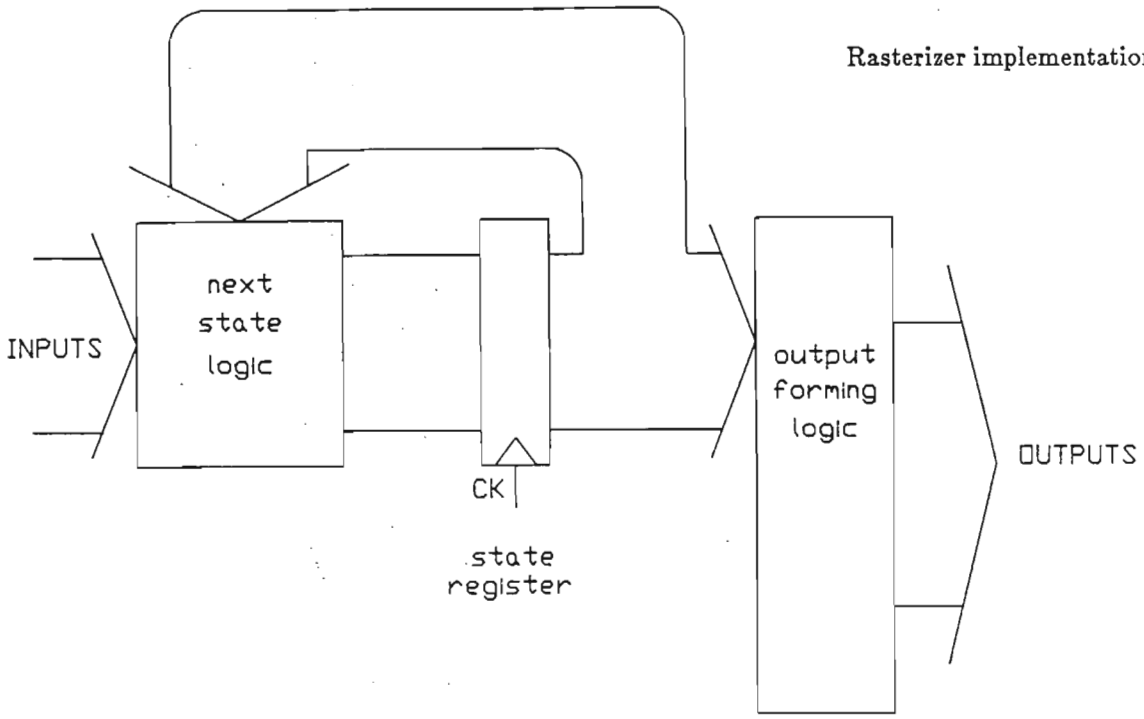
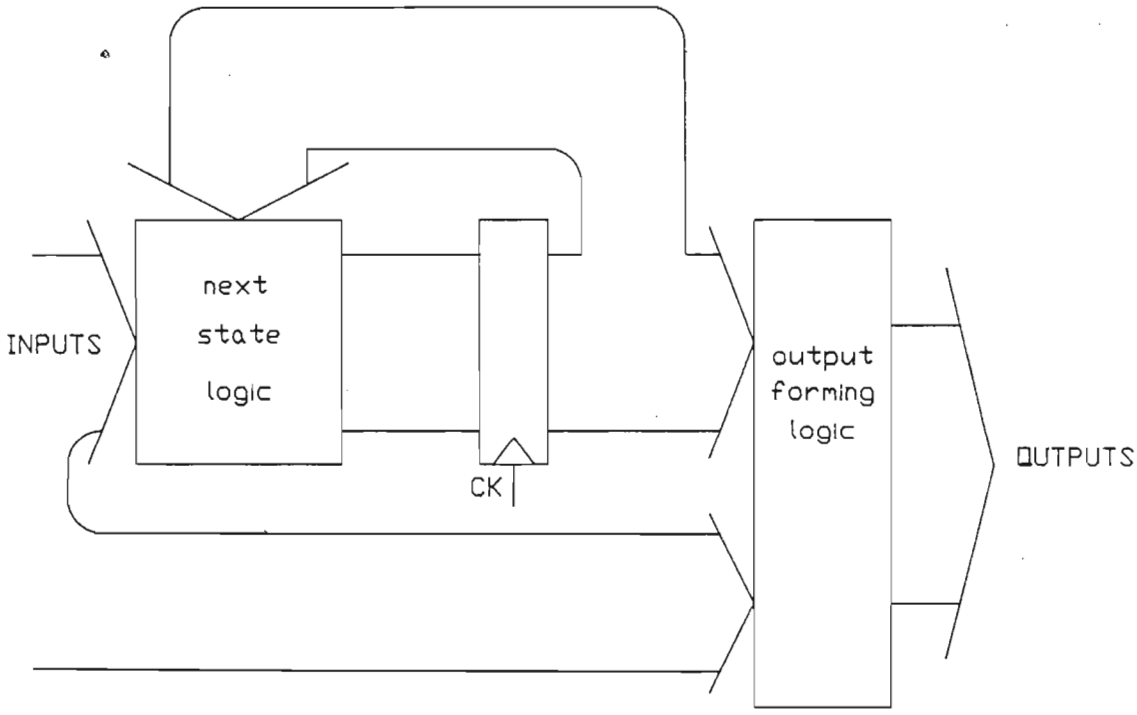


Figure 3.13: (a) General structure of: Moore machine.



(b) General structure of: Mealy machine.

EQ	X1=X2 and/or Y1=Y2
C16	sixteen cycles completed
ENX	adjust X unconditionally
ENY	adjust Y unconditionally
INX	increment not decrement X
INY	increment not decrement Y
MAN	current line is manhattan

The next state logic and output forming logic was formed and minimized by PEG, a Berkley PLA equation generator [Hamac83], running on a VAX 11/780 at Clemson University (SC, USA). The program is part of a PLA generation suite. The finite state machine was put through the process for the author by Mr D C Levy of the University of Natal, who was on sabbatical at Clemson University at the time. The PEG language representation of the state diagram, and the resulting logic is to be found in appendix 3. The process is reported to have run for 45 minutes on the VAX.

The PLA equations were ultimately implemented in random logic on the gate array since the PLA structure was not available on the array(see appendix 3 for full next state logic and output forming logic diagrams). In order to do this, the equations generated by PEG were manipulated and converted into logic circuits consisting of NAND and NOR gates. Any changes involved manual manipulation of the equation terms.

3.7. Timing

The system was designed to use a simple single phase clock. The control logic and the registers in the data structure are both sensitive to the positive edge of the external clock. The simple clocking procedure allows the arithmetic hardware to settle for the whole cycle minus the setup time of the registers (4ns). A complex multiphase clock would not be very useful in the limiting environment of the gate array.

Chapter 4.
Gate array design

4.1. Gate arrays [Beres83]

The gate array is the simplest route to realizing a semicustom IC implementation of a general digital circuit. It is an integrated circuit which is preprocessed in a standard format to the point before the metal interconnection layer is applied. Only the metalization layer is customized. It consists of an array of identical configurable cells. Cells and groups of cells are configurable as basic logic elements using cell-internal metal tracks. The user, a semicustom IC designer, builds his circuit using these basic gates and thereby defines their cell-external interconnection. These interconnection tracks and the cell-internal tracks are combined to produce the metal layer masks of which there are either one or three, one for single level metal and three for double level metal. The three in the double level case are metal-1, metal-2 and vias (the vias are metal-1, metal-2 interface). These are the only user-unique masks of at least six, which is a major cost reducing factor. Gate arrays in ECL, TTL, IIL and MOS technology are in production, each with different characteristics.

The most popular and common, because of low power consumption and high density, is the MOS technology gate array. The MOS gate array architecture (figure 4.1) consists of rows of cells separated by interconnection highways (blank areas reserved for wiring). Around the block of cells is a row of peripheral cells which act as interface buffers between the circuit on the array and the outside world. The basic MOS cell consists of two pMOS and two nMOS transistors configured as shown in figure 4.2. Power rails run above and below each row of cells. The basic cell may be configured as one of a number of simple logic elements such as the NAND gate in figure 4.3. More complex functions such as a full adder require more than one cell.

Reproduced from [PlesD86]

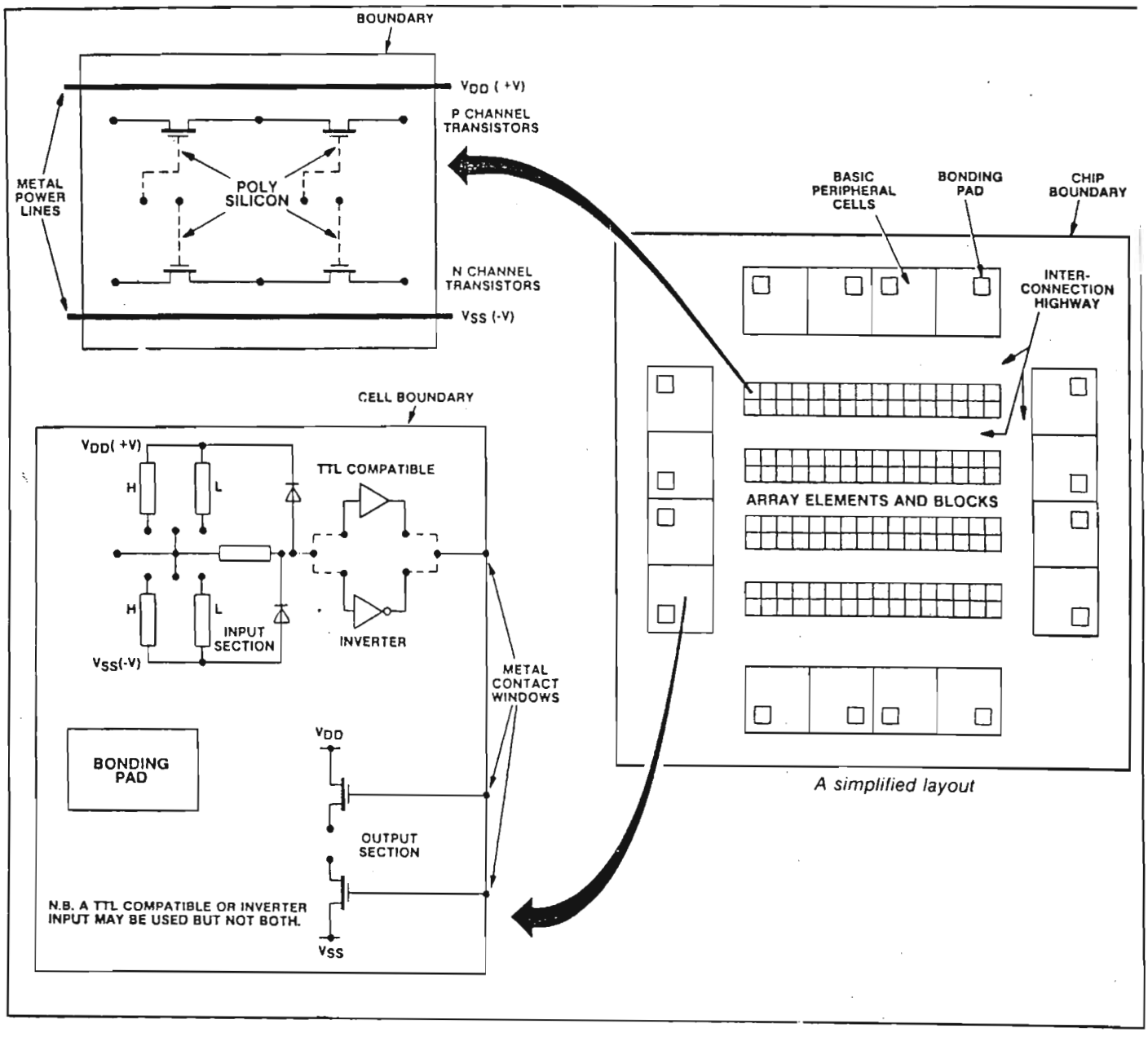


Figure 4.1: MOS gate array architecture.

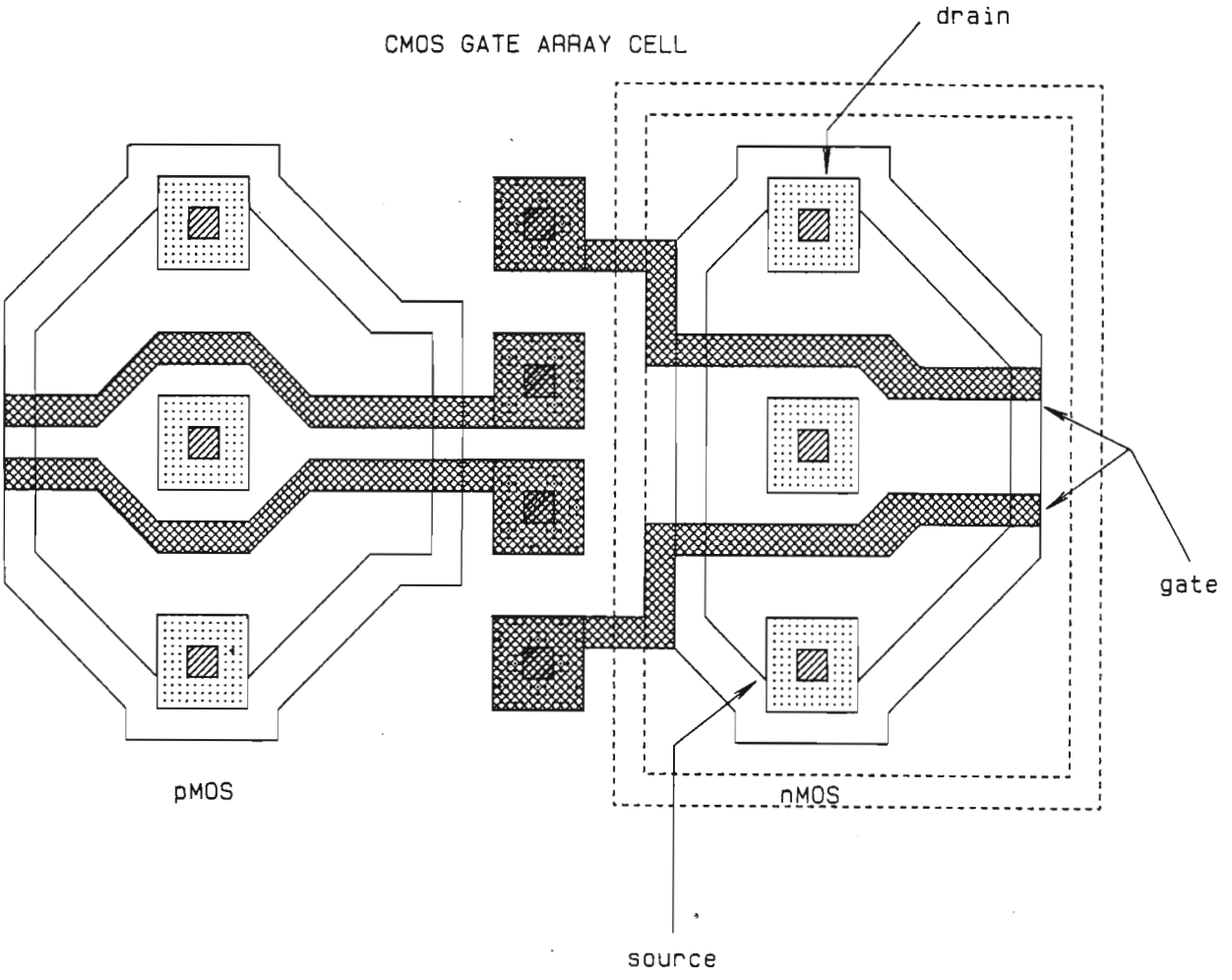


Figure 4.2:Basic MOS array cell.

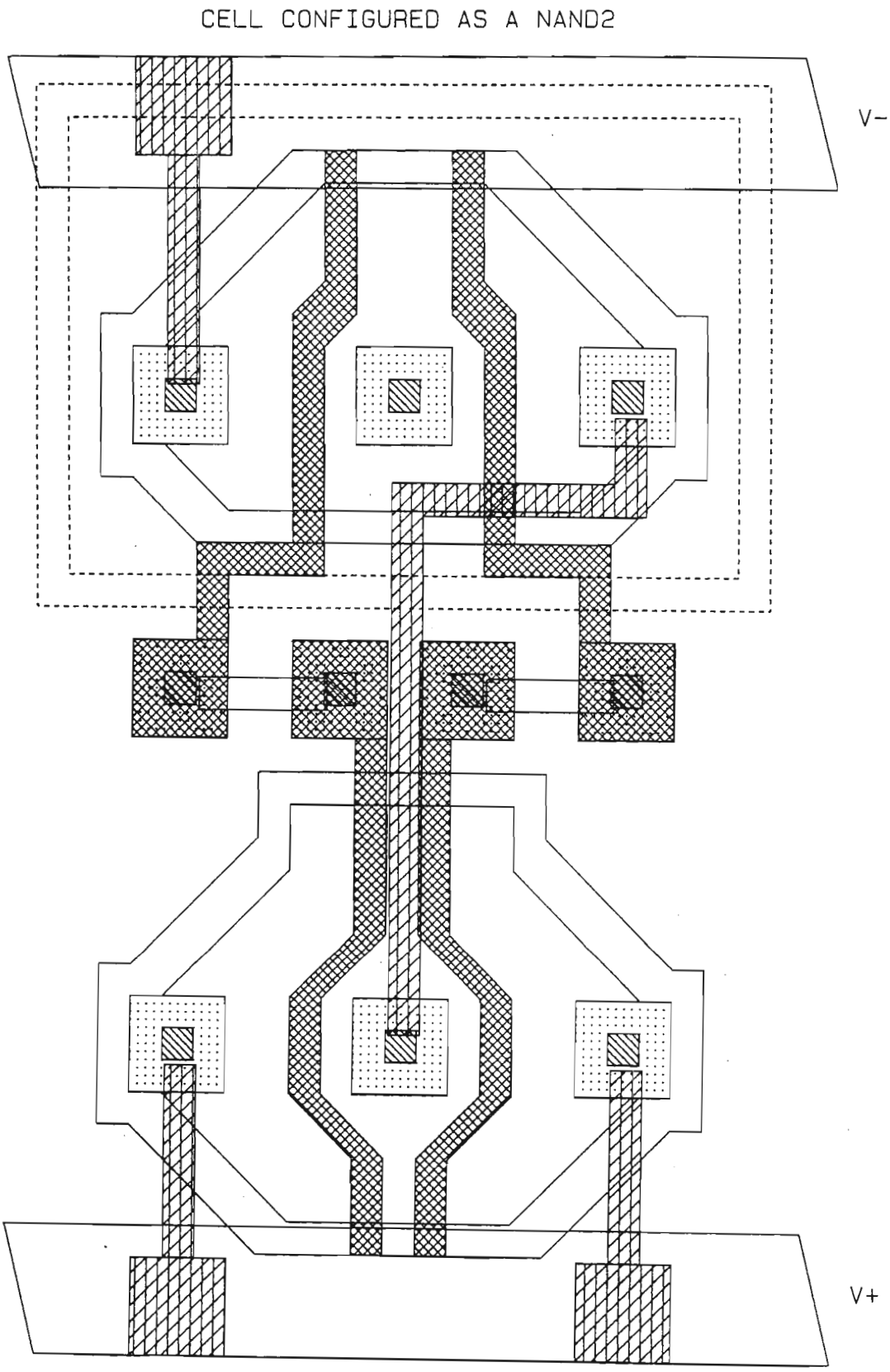


Figure 4.3: MOS cell configured as a NAND gate.

CAD tools are generally available to assist in the design process. The most basic CAD tool requirements are a logic simulator and an interconnection track router. Often, there are tools available to aid cell placement, circuit-schematic capture, etcetera.

Disadvantages of gate arrays as opposed to full custom or semicustom IC's are mainly due to the inflexibility of the cell structure (especially the size of the transistors in the cell) and the overall IC layout. The disadvantages amount to loss of performance and decreased density.

4.2. The Plessey system [PlesD86]

Plessey Semiconductors (Swindon UK) manufactured the rasterizer IC using their CLA5000 series of arrays. CLA5000 is a 2 micron CMOS process with double level metalization. There are a number of arrays in the family ranging from 640 to 10044 gates (see table 4.1). The array used for the rasterizer was a 4408 gate CLA55xx (the part number ascribed to the project was CLA5524).

Table 4.1 CLA5000 arrays

DEVICE CODE	GATE COUNT	I/O CELLS	HIGHWAY WIDTH
CLA51XX	640	36	12
CLA52XX	1232	48	12
CLA53XX	2016	64	12
CLA54XX	3060	80	12
CLA55XX	4408	96	16
CLA56XX	5984	112	16
CLA58XX	8064	144	20
CLA59XX	10044	160	20

As usual the array element consists of two p-channel and two n-channel devices. An array block is formed from four of these cells by reflecting the basic cell about central axes (figure 4.4). These blocks form the rows of cells and they are separated by

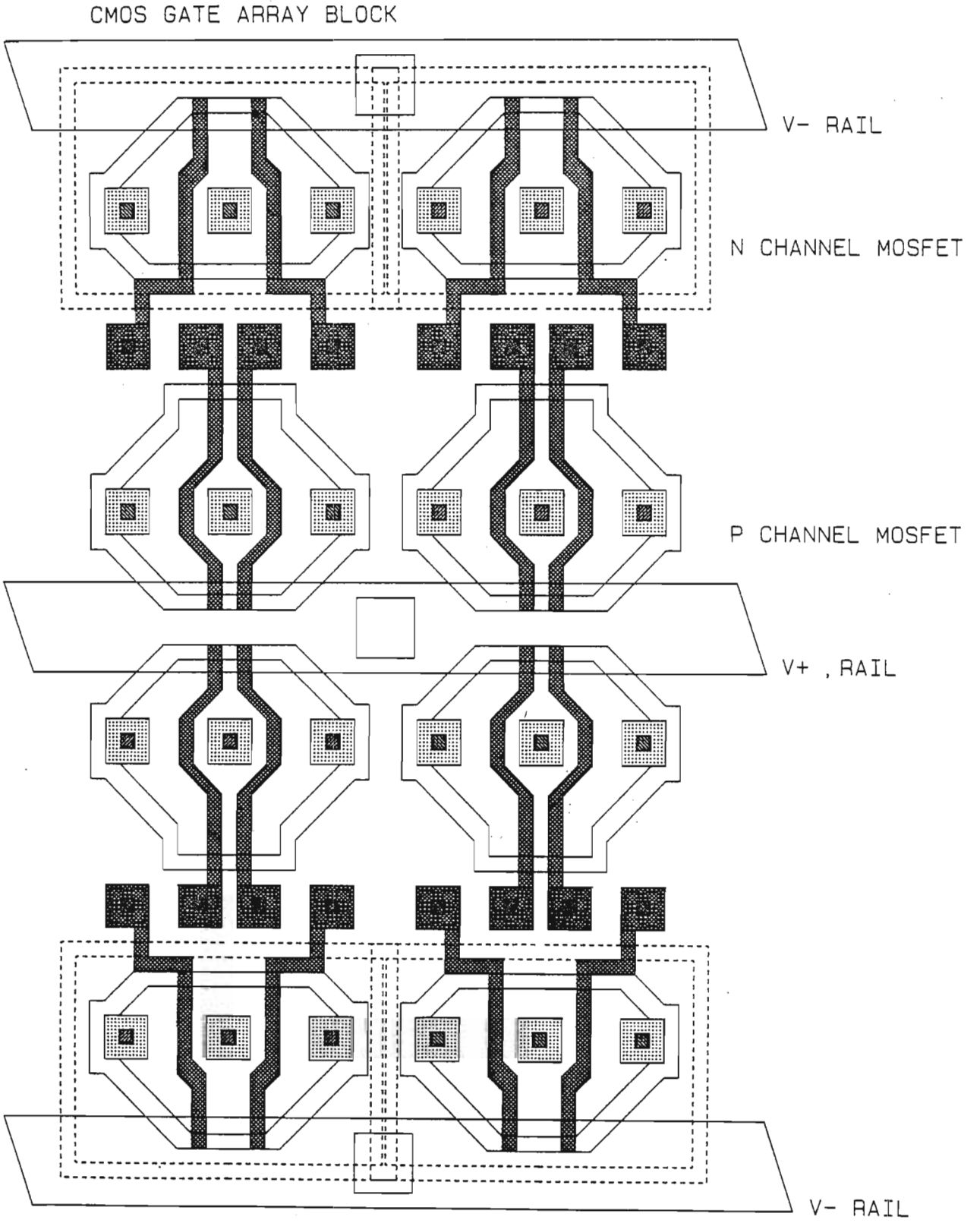


Figure 4.4: Array block.

interconnection highways wide enough to accommodate 12, 16, or 20 tracks depending on the array size. Peripheral cells are arranged around the cell array; these may all be configured as several different types of input, output, or bi-directional cells (figure 4.1). The most complex precharacterized cells available are the full adder and the edge triggered D-type flip-flop. The set of precharacterized cells form what is known as the technology-library. The designer or Plessey may use these cells to construct macros called subcircuits which may be used to build a so called user-library. The circuit under design is described in terms of components of the technology-library and the user-library. Only one level of user hierarchy is allowed by the Plessey CAD suite; that means that user-library components may not be made up of other user-library components.

The CAD suite consists of programs to perform simulation, check the testability of the circuit and autoroute.

The design flow is depicted in figure 4.5. The middle block of the diagram labeled PLESSEY/CUSTOMER (indicating that the responsibility for that part of the design may be taken by the customer or by Plessey staff) depicts an iterative process, done mainly with the aid of the CAD suite.

4.3. The Plessey components

4.3.1. Overview

Plessey provide a range of components in their technology library which have been designed at transistor level and hence characterized by testing in terms of timing and drive capability. These characteristics are built into the technology library. Any macros and user library components are built of and take their characteristics from the technology library components. The standard components consist of, a set of simple gates such as inverters, 2 to 4 input NAND and NOR gates, some complex gates including EXOR gates, a range of clock drivers, some arithmetic functions, transmission gates, and a set of bistables. The whole

Reproduced from [PlesD86]

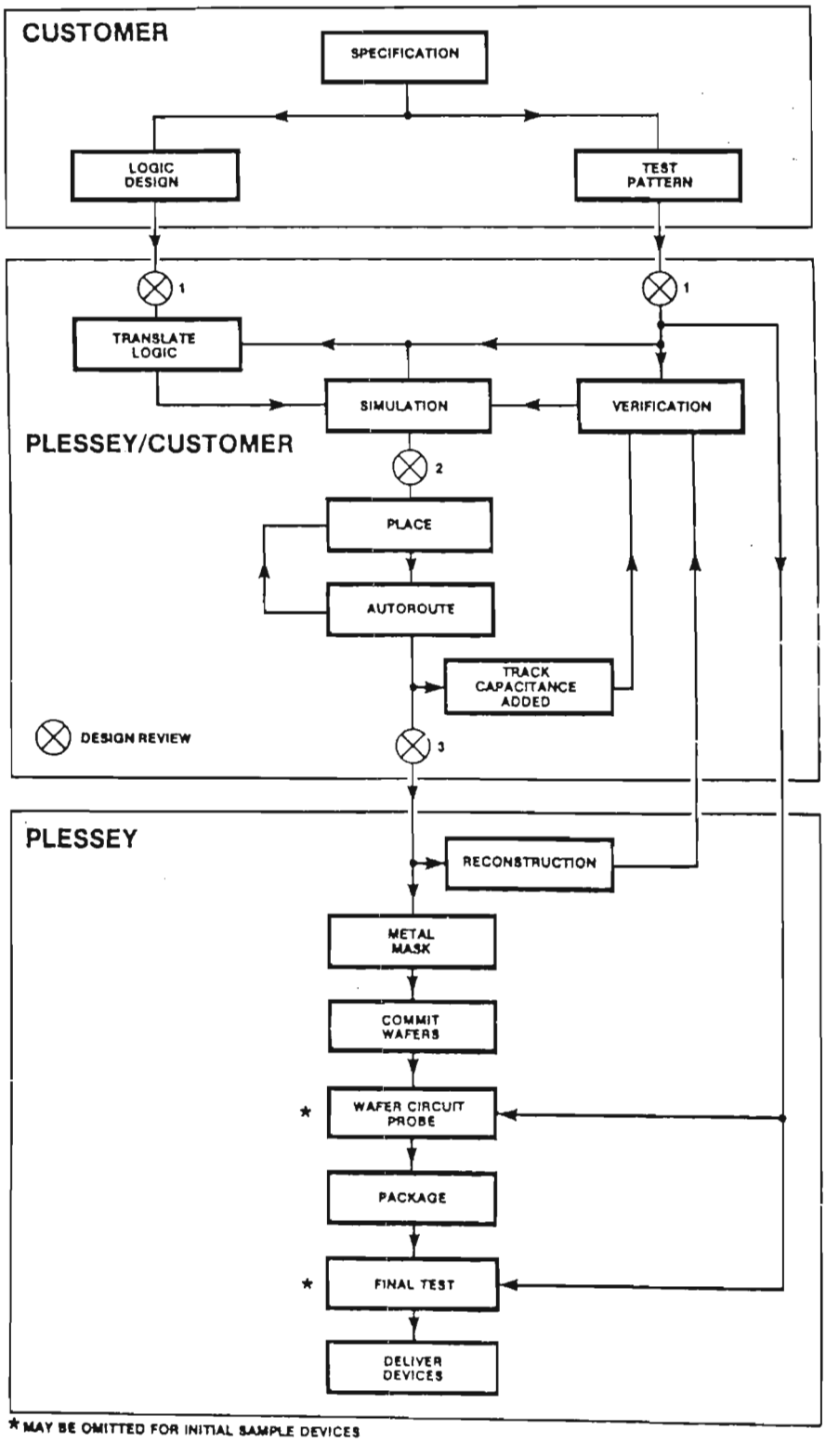


Figure 4.5: Design flow diagram.

range is detailed in table 4.2.

4.3.2. Drive capability and loading

In the CLA5000 series the shape factors of the p– and n-transistors is such that the p-transistor has approximately twice the resistance of the n-transistor and hence has half the drive capability. Since gates are made up of series and parallel connections of p– and n-transistors, those gates that feature more series p-transistors are slower than those that have more parallel p-transistors. For this reason NOR gates and especially the many-input NOR gates are to be avoided. The basic components are characterized in terms of drive capability. Drive is measured in terms of the load unit (LU) which is the load presented by one gate. The CAD software keeps track of loading in the circuit and issues appropriate warnings. After the routing phase, the tracks loads are calculated and made available for checking the ability of the circuit to drive its interconnection. Track loading will often double loads on a gate.

4.3.3. Tristate components

Transmission gates are the only mechanism available on the CLA5000 gate array for tristating a node. There are several problems associated with these components:

—they are bi-directional
—they have no voltage gain
—they increase capacitance on the node which they feed

Transmission gates may however be used as tristate buffers if they are driven by inverters. The buffers so formed still have very limited drive capability.

The internal configurations of some of the bistables use transmission gates, as is common in CMOS design.

Table 4.2 Cell List

Classic	Format	No of Elements	Description
2INV	[Z1BAR,Z2BAR,Z1,Z2]	1	dual inverter
INV4	[ZBAR,Z]	2	inverter,quad drive
INV8	[ZBAR,Z]	4	inverter, octal drive
NAND2	[F,A,B]	1	2–input NAND gate
NAND2A	[F,A,B]	1	2–input NAND gate (between clocked cells)
NAND3	[F,ZBAR,A,B,C,Z]	2	3–input NAND gate + inverter
NAND4	[F,A,B,C,D]	2	4–input NAND gate
NOR2	[F,A,B]	1	2–input NOR gate
NOR3	[F,ZBAR,A,B,Z]	2	3–input NOR gate + inverter
NOR4	[F,A,B,C,D]	2	4–input NOR gate
A2O2I	[F,ZBAR,A,B,C,Z]	2	2–input AND to 2–input NOR gate + inverter
O2A2I	[F,ZBAR,A,B,C,Z]	2	2–input OR to 2–input NAND gate + inverter
2A2O2I	[F1,F2,A1,B1,C1,A2,B2,C2]	3	dual 2–input AND to 2–input NOR gate
2O2A2I	[F1,F2,A1,B1,C1,A2,B2,C2]	3	dual 2–input OR to 2–input NAND gate
2ANOR	[F,A,B,C,D]	2	2–input ANDs to 2–input NOR gate
2ONAND	[F,A,B,C,D]	2	2–input ORs to 2–input NAND gate
A3O2I	[F,A,B,C,D]	2	3–input AND to 2–input NOR gate
O3A2I	[F,A,B,C,D]	2	3–input OR to 2–input NAND gate
A2O3I	[F,A,B,C,D]	2	2–input AND to 3–input NOR gate
O2A3I	[F,A,B,C,D]	2	2–input OR to 3–input NAND gate
EXOR	[H,G,F,ZBAR,A,B,D,E,Z]	4	XOR gate + NAND GATE + inverter
EXNOR	[H,G,F,ZBAR,A,B,D,E,Z]	4	XNOR gate + NOR gate + inverter
HADD	[F,FBAR,G,ZBAR,A,B,Z]	4	half adder + inverter
SUM	[SO,SOBAR,A,B,CI,D]	4	sum block
CARRY	[CO,COBAR,F,A,B,CI,D,E]	4	carry block + NOR gate
FADD*	[SO,SOBAR,CO,COBAR,F,A,B,CI,D,E]	8	full adder + NOR gate

CLKA	[A,ABAR,AI]	2	basic clock driver
CLKA–	[A,ABAR,ZBAR,AI,Z]	2	basic clock driver, + inverter
CLKA+	[A,ABAR,ZBAR,AI,Z]	2	basic clock driver, + inverter
CLKB	[A,ABAR,ZBAR,AI,Z]	4	large clock driver, + inverter
2TM	[C,CKT,CKI,A,B]	1	2–way transmission gate
TM	[C,CKT,CKI,A]	1	buffered transmission gate, inverting
RSNOR	[Q,QBAR,R,S]	2	NOR set-reset latch
RSNAND	[Q,QBAR,R,S]	2	NAND set-reset latch
DL	[Q,QBAR,EN,ENBAR,D]	2	bistable latch
DLRS*	[Q,QBAR,CKT,CKI,D,R,S]	3	set-reset bistable latch
DF	[Q,QBAR,CKT,CKI,D]	4	master-slave D type
DFRS*	[Q,QBAR,CKT,CKI,D,R,S]	6	set-reset master-slave D type
MDF*	[Q,QBAR,CKT,CKI,ST,SI,D1,D2]	6	2:1 mux master-slave D type
MDFRS*	[Q,QBAR,CKT,CKI,ST,SI,D1,D2,R,S]	8	2:1 mux set-reset master-slave D type
TRID	[P,N,D,T]	4	tristate driver

Peripheral cells

OPS	[OUT,HBAR,MBAR,A,H,M]	output inverter, single drive + inverter
OPD	[OUT,MBAR,A,M]	output inverter, dual drive + inverter
OPQ	[OUT,MBAR,A,M]	output inverter, quad drive + inverter
OPTRD	[OUT,P,N]	tristate output, dual drive
OPTRQ	[OUT,P,N]	tristate output, quad drive
OPDOD	[OUT,A]	open drain output, dual drive
OPODD	[OUT,MBAR,A,M]	open drain output, dual drive + inverter
OPQOD	[OUT,A]	open drain output, quad drive
OPODQ	[OUT,MBAR,A,M]	open drain output, quad drive + inverter
IPI	[F,IN]	input inverter
IPIL+	[F,IN]	input inverter, lower resistance pullup

IPIH+	[F,IN]	input inverter, higher resistance pullup
IPIL–	[F,IN]	input inverter, lower resistance pulldown
IPIH–	[F,IN]	input inverter, higher resistance pulldown
IPS	[F,IN]	low threshold input buffer
IPSL+	[F,IN]	low threshold input buffer, lower resistance
IPSH+	[F,IN]	low threshold input buffer, higher resistance
IPSL–	[F,IN]	low threshold input buffer, lower resistance
IPSH–	[F,IN]	low threshold input buffer, higher resistance
2BD	[H1BR,H2BR,H1,H2]	dual inverter, type H

4.3.4. Bistables

There are a number of bistables in the family from simple NAND set-reset latches to edge triggered D-types with asynchronous set and reset. The structure of the D-type flip-flop is shown in figure 4.6.

The flip-flops require two non-overlapping clock signals. These are generated by the clock drivers described below. The CAD software handles checking setup and hold time limits (both 4ns) and also checks for excessive clock skew and load.

4.3.5. Clock drivers

Both the Transmission gates and the flip-flops require a non-overlapping clock signal pair for correct operation. There is a group of clock driver cells designed specifically for this task. The array structure is designed such that these clock drivers may be positioned next to a row of clocked elements in such a manner that the wiring of the clock signals will not use up routing channel space. The clock driver cells may be rotated or inverted to provide either positive or negative edge triggering to clocked cells.

4.3.6. Input and output (IO) blocks

The IO blocks may be configured as two types of input cell: non-inverting TTL-compatible buffer or inverting CMOS buffer. The input configurations include resistor-diode protection against electrostatic discharge. Pull-up and pull-down resistors are also available.

The output cells are capable of driving one, two or four standard TTL loads respectively and are of three basic types:

—open drain
—push pull
—tristate

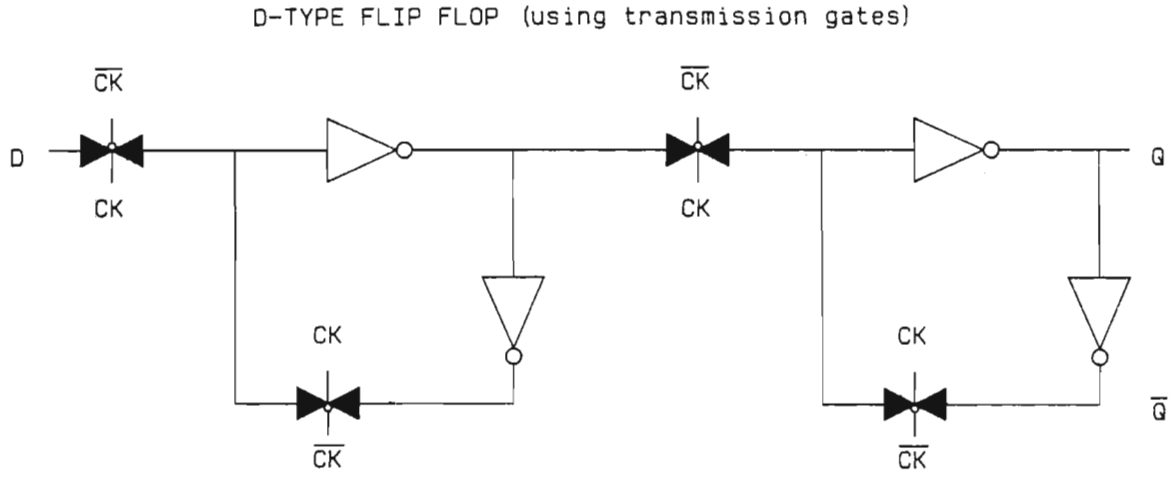


Figure 4.6: D-type flip-flop structure.

4.3.7. Power rails

There are a number of power supply pads on each array, more on the bigger than the smaller arrays. In addition IO cells may be configured as V– connections to increase the current carrying capability of the negative rail. This is especially necessary if many TTL compatible output cells are in use as they are required to sink large currents.

4.4. The Plessey gate array CAD suite [PlesR86]

4.4.1. Introduction

The CAD suite for the CLA5000 series consists of: CLASSIC, a logic simulation system; CLASP, for measuring circuit testability; and SCARP, an autorouting system. The CAD suite has been written to run on DEC VAX computers running the VMS operating system. All the programs use VMS-like command structures and make full use of the operating system features, especially the capability to run programs in batch and the 132 column CRT screen drivers.

4.4.2. CLASSIC "custom logic analysis and simulation system for ICs"

4.4.2.1. Overview

CLASSIC is a general logic simulation system, the core of which is an event driven simulator.

CLASSIC has four distinct parts:
The compiler (CCLassic) which converts the circuit description file into data files for the other parts of CLASSIC and does initial structure checks.

The simulator (SCLassic) which does the actual simulation.

The display editor (ECLassic) which is used to extract the simulation results from data files created by SCL.

The test program (TCLassic) used by Plessey engineers to make a test routine from simulator input test vectors and simulator output data files.

Refer to figure 4.7 for a functional flow diagram of CLASSIC.

All the user files pertaining to a CLASSIC simulation are named with one main part and different extents, such as RASTER.CCL for the circuit file and RASTER.TRK for a router input file. The extents identify the file and so will often be used alone in the text below as .CCL etcetera.

4.4.2.2. CLASSIC libraries.

The circuit description file is in the form of a list of components with all the connections to each component, inputs and outputs, named. The components are all extracted from one of three distinct libraries: the technology library, the system library and the user library.

The technology library is the basic library that defines the environment for CLASSIC. It describes the characteristics of the physical components that will be used to implement the simulated circuit; for example, a technology library may contain the LSTTL components. There is a set format in which any logic technology may be described. This convention allows the component to be modelled in terms of logical function and timing. The timing information is broken up into intrinsic delay and loading delays due to finite rise and fall times. Plessey controls the technology library for the CLA5000 family of gate arrays. As the name suggests, CLASSIC is designed for use with IC logic and hence all timing delays for a particular component are equal in a circuit; this is a limitation when using the system for simulation of discrete logic circuits.

Reproduced from [PlesD86]

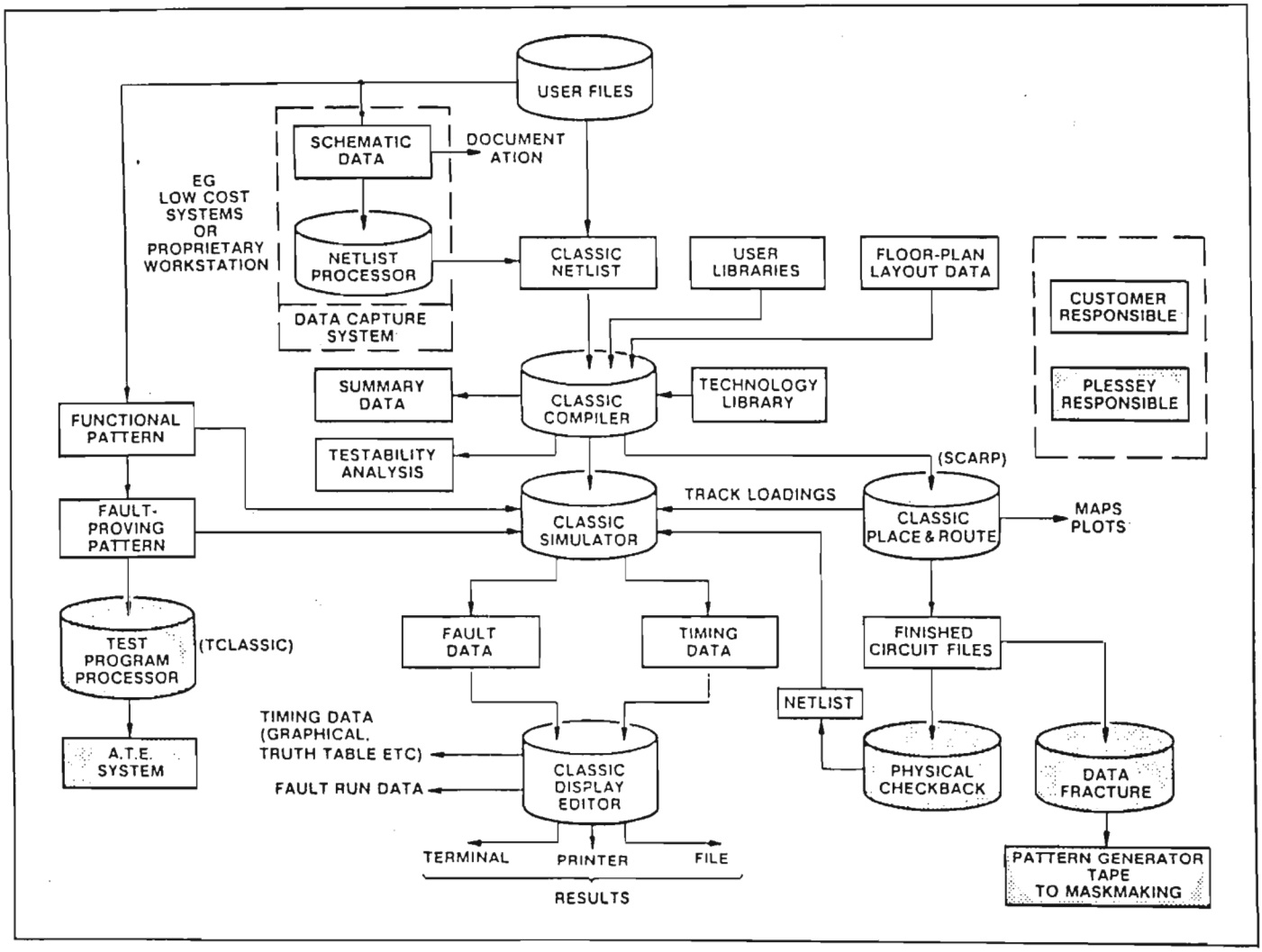


Figure 4.7: Functional flow diagram of CLASSIC.

The system and user libraries are essentially the same except for their ownership. A project manager or in some cases Plessey will control a system library while any user may create a personal user library. These libraries contain components additional to the technology library. There are two distinct types of component namely, a cell and a subcircuit. A cell is built in the same way as a technology library component, with a functional descriptor and a timing characteristic, while the subcircuit is a macro built of technology library components. Unfortunately only one level of subcircuit hierarchy is allowed.

4.4.2.3. The Compiler (CCLassic).

The compiler requires the circuit description file as input (.CCL file). This file contains the compiler options, which include the list of libraries to be used, the size of the simulation mesh and the types of output files required. The compiler "flattens" the circuit for the simulator by expanding all subcircuit references, and creates data files for other stages in the design process, such as a list of cells and their locations (.CEL file) and a list of required connections (.TRK file) for the autorouter.

A component is placed in the circuit with one line of the compiler source file (.CCL). Schematic capture is at present not available but the capability is under development. Plessey intend to support schematic capture libraries on a range of popular commercial workstations, including Daisy, Valid, Mentor and Futurenet.

A typical compiler source line follows:

D1: (23,45) DFRS(Q,QBAR,CLK,NCLK,DATA,+,+)

where
D1: is a unique block name;

(23,45) is the X and Y co-ordinate of the block origin;

DFRS is the library name of the model required;

Q,QBAR,... are the names of the nets connected to the model in the order defined by the library entry.

A boolean equation format is also available to describe circuit components.

The compilation turns up a limited number of errors, mainly concerned with syntax. It also finds non-unique block names, unconnected inputs to blocks and warns of undeclared inputs and outputs to the circuit. The compiler calculates the total load on every gate output and issues a warning if the load exceeds the drive capability. There is facility for adding extra load to a node. This should be used to add track loads near the end of the design process to check the drive capability of the circuit.

4.4.2.4. The simulator (SCLassic).

The simulator performs the actual simulation of the operation of the circuit. In addition to some data files describing the circuit provided by the compiler, the simulator expects an input file defining the circuit excitation pattern (.SCL file). The excitation pattern is described in the form of a set of named waves that have a starting value of 0 or 1 and then change state at certain times on a time scale which starts at zero. The simulator has a set of options which must be included in the file containing the excitation pattern.

There are two important timing notions: the mesh and the period. The mesh, as defined at compile time, dictates the interval between calculations by the simulator, while the period is the unit of time which is used to define the input patterns. The state change time for an input wave is expressed as a multiple of the period.

The simulator tracks changes of state, in time, on all the circuit nodes. A node may be in one of four states, or four transition states. The states are as follows:–

0 — positive logic false
1 — positive logic true
Z — high impedance
X — undefined

The transition or intermediate states are:–

u — changing from 0 to 1
d — changing from 1 to 0
z — changing to or from Z
x — changing to or from X

The range of gate delays normally found in IC's (MAX, MIN, TYP) are catered for by the CLASSIC simulator so that a designer may test circuit operation to ensure that timing is not too critical. The delay value (MAX, MIN, or TYP) required for a particular simulation must be specified in the options section of the simulator input file.

The simulator has a fault simulation option which does a "stuck at one" and "stuck at zero" test on every node in the circuit. This means that the simulator does a good simulation and then re-simulates with one node stuck at one or zero until it detects a difference between the current simulation and the good simulation. It only checks for this difference at the declared output nodes. Detection of a fault is defined as the detection of this difference at the output nodes. The simulator records the number of clock cycles that it took to detect the fault, moves on to the next node and starts the process again. Clearly this is a time consuming task. The fault simulation shows the effectiveness of the excitation pattern as a test pattern and also its efficiency. The efficiency is a measure of the length of time taken to detect the bulk of the faults. A time versus faults-detected profile is a helpful summary of the results.

The simulator is capable of running 16 simulations in parallel, each with a different set of stimuli. This feature is useful for running MAX, MIN, and TYP simulations together as well as for fault simulations. Fault simulation automatically invokes parallel simulation to try and cut down the time required.

The simulator handles the case of an X state appearing on the input to a gate in two ways, depending on whether XPROP is on or off. If XPROP is on, an X state on a gate input will cause X to appear on the output. This results in a pessimistic simulation, while if XPROP is off an optimistic simulation results. If XPROP is off, two X states must be present at gate inputs for a change of state to X on the output to begin.

The simulator reports syntax and consistency errors in an output file (.SSL) and stores the results of the simulation in a binary data file (.CBR). The display editor described next interrogates the data file.

4.4.2.5. The display editor (ECLassic).

The purpose of the display editor is to extract different classes of data from the simulation data (or history) file and display them in a useful form. The display editor is an interactive program with a number of different modes of operation, each designed to allow analysis of the circuit operation and performance.

Window analysis mode is the best way to study the relative timing of the signals on all nodes in the circuit . The presentation of the data is as pseudo-waveforms on a text screen. A set of appropriate characters is mapped onto the set of states as follows:

^ : 1
_ : 0
/ : transition 0 to 1
\ : transition 1 to 0
X : X
x : x
Z : Z
z : z

Also, some special characters are required for problem conditions:

! : The time period represented here contains a glitch.
: The time period represented here contains a pulse.

An example waveform is shown below:

```
____///^^^\\_____//!\_________xXXXX/^^^^^^zZZZ\___
```

The window analysis mode is good for finding timing problems but only a small number of nodes may be studied at once.

The table analysis mode provides the bulk data summary required to check the logical operation of a digital system. The most common table format is a row for each time step with the state of a node in each column. A 132 column CRT screen can hold a large number of nodes. Options are available to group a set of nodes, such as a data bus, and display the set as a value to some base, such as a hexadecimal number.

The display editor will also scan the data file for marginal conditions such as race conditions, glitches and spikes. Glitches and spikes are defined in CLASSIC, respectively, as pulses below a user-defined width and pulses that never reach a logic one or zero at their peak.

An error analysis mode lists clock-skew, and setup and hold time errors as defined in the technology library for clocked cells such as flip-flops.

If a fault analysis is performed, the display editor is also used to summarize the results. It lists the faults detected and not detected and provides a histogram which shows the clock-cycles versus faults detected profile mentioned above.

4.4.3. CLASP [PlesC86]

This program may be run after compilation. It calculates a series of numbers called controllability and observability figures for every node in the circuit. These are intended to give a designer an idea of how testable his circuit will be. They are measures, dependent on circuit structure and independent of any test vectors, of the ability to control a node from the inputs and to observe the value of a node at an output, respectively. There are two sub-classes within the series namely, sequential controllability and observability figures, and combinational controllability and observability figures. The sequential class is basically a measure of the number of clock cycles required to control or observe a node, while the combinational class is basically a measure of how deep a node is buried in a combinational block. The figures are not a great deal of help until a designer has done a number of different types of designs himself, because Plessey do not commit themselves to specifying what figures are good and what are bad. An important characteristic of the figures, however, is that they should all be roughly the same; any figure much bigger than the rest indicates a node that will cause the whole circuit to be difficult to test. Generally also, lower figures are clearly good.

4.4.4. CLAMP

CLAMP is intended as a placement aid. In the CLASSIC system, placement is done by assigning to the bottom left hand corner of any cell a cell-origin co-ordinate; the choice of this co-ordinate effects the orientation of the cell. These co-ordinates are entered in the circuit description file, which is an ordinary text file of circuit source code for the compiler. The co-ordinates may therefore be entered by using a text editor instead of CLAMP. CLAMP presents each block in the circuit in sequence for the user to interactively enter a co-ordinate — this is not ideal since if a co-ordinate is mistyped the whole sequence must be rerun to correct the errors. The VMS text editor EDT is easier to use.

4.4.5. SCARP [PlesS86]

SCARP does four distinct tasks: cell layout, layout-plot, track route, and track-plot.

The first phase converts the .CEL file which contains the cell origin co-ordinate information (a result of the compilation) into a file for use by the routing algorithm. The conversion involves the construction of a complete two-dimensional floor plan and hence any overlaps or conflicts are reported. The accompanying phase, layout-plot, produces a diagram of this floor plan for use by the designer in ensuring that all the cells are in the correct places.

The third phase, routing, is the core of SCARP and is the most time consuming. The router uses four passes in an attempt to complete the circuit interconnection on two levels of metal tracks. Plessey insist that the designer perform a 100% successful autoroute: no manual intervention is allowed. The algorithms are briefly described as follows:–

The process begins with two passes of the so called HEURISTIC algorithm, with a weaker set of restrictions for the second pass. HEURISTIC attempts to connect points on a node with single vector lines, "L" lines and two parallel lines with one bridge.

Two passes of the maze runner LEE algorithm follow the HEURISTIC passes. The first pass is restricted to tracking horizontally in metal-1 (in the routing channels) and vertically in metal-2, while the second pass is unrestricted.

The track-plot may be used by the designer to try and solve any routing problems by identifying the unrouted points and areas of congestion. There is a facility to priority route certain nets, but forcing the router to do a net first in preference to others generally causes more problems than it solves and the practice is not encouraged. Prioritising a long and troublesome net will usually cause a number of shorter net routes to fail. The best method of influencing routing is to shift blocks around and change the orientation of certain cells.

SCARP also provides a file of the capacitive track loading for each net in the circuit. This information must be put back at the compilation stage so that the simulation may be redone to check the drive capability of the circuit.

4.5. The customer interface.

Plessey will design and fabricate a circuit according to a customer specification, but as with many gate array vendors they prefer the customer to do the complete design. Plessey will lease a customer or prospective customer the CAD suite if they have the hardware to run it on, or they will provide for remote use of one of their own machines. They also provide a number of design rooms at their semi-custom design centres, such as the one in Swindon (UK), in which a designer may use a terminal on a machine running the appropriate software.

Chapter 5.
Computer Aided Design (CAD) of the rasterizer

5.1. Scope of the work

The work done by the author included: system design; circuit design and translation into a CLASSIC input file; circuit simulation and refinement; test vector generation and verification; array layout and routing. The routed layout was handed over to Plessey for manufacture. The completed IC was tested on Plessey Teredyne test equipment according to the test vectors and simulation results specified by the author. Working prototypes were sent to the author for subsequent system testing.

5.2. Overview of the procedure (methodology)

Refer to figure 5.1 for a flow diagram of the design procedure.

The system was clearly defined and specified according to the desired operation requirements. The circuit was then designed on a functional block level to fulfil the specifications.

Each block was then dealt with in detail. The function was reduced to gate level, described in a CLASSIC source file, compiled and simulated, while applying suitable test vectors. An iterative process was used to solve problems in the operation of the functional blocks.

The funtional block was then added to a user library if it was totally primary (i.e. made up only of basic cells). No functional blocks that called other user library blocks could be included in the user library, because CLASSIC only allows one level of subcircuitry. If it was not primary, the block was added to the main circuit description file. The adder, for instance, was dealt with as a functional block, but not added to the subcircuit library; some relevant data describing the structure and testing of the

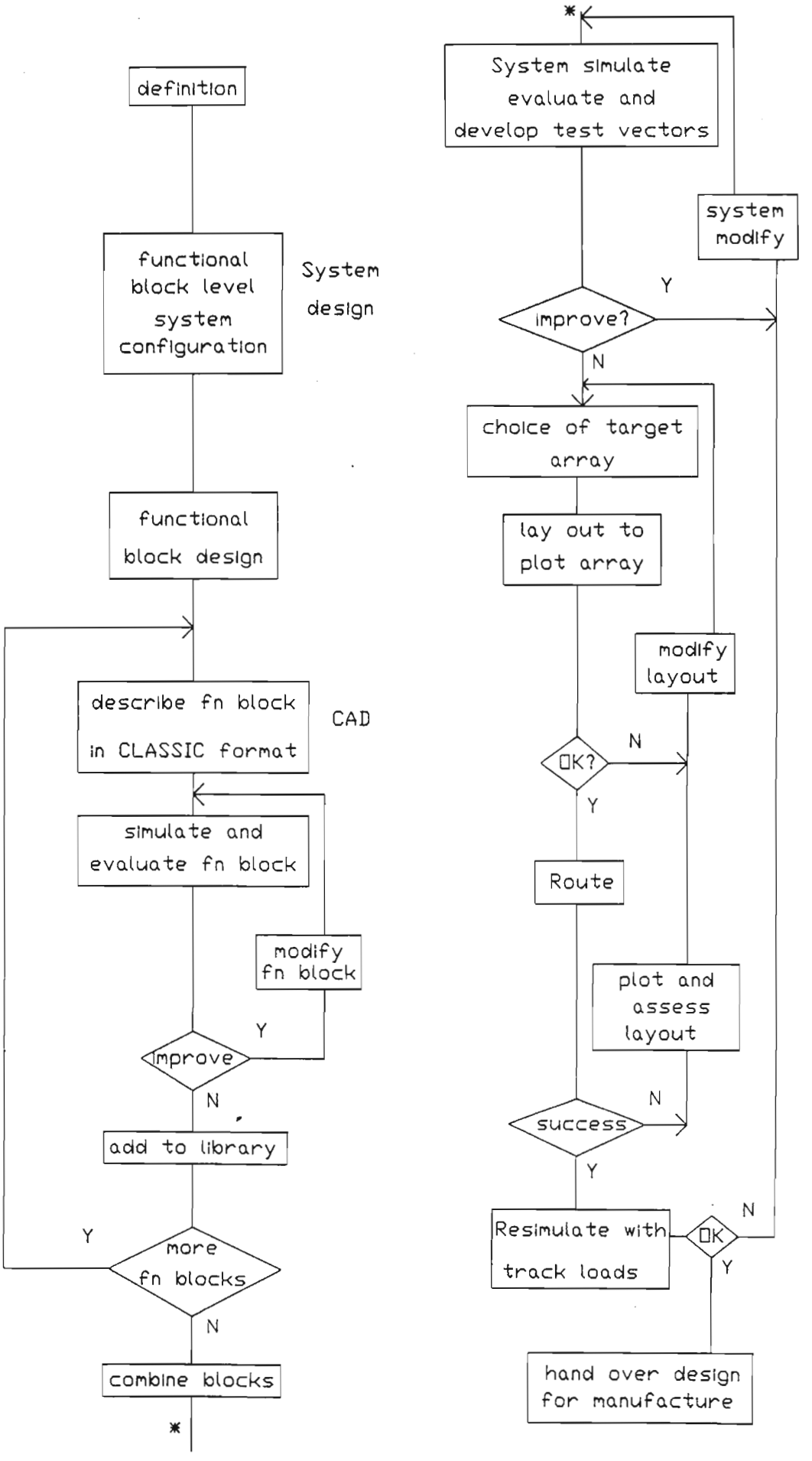


Figure 5.1: Flow diagram of design procedure.

adder is included in appendix 4. An example of circuit blocks that could be added to the user library is the register set (see library file in appendix 5).

Once all the functional blocks were completed, they were combined to form the complete system. The system was tested and refined until it was operating correctly.

At that stage is was possible to judge the size of the circuit accurately and hence choose a target gate array.

The layout was compiled next, followed by autorouting.

Finally the router produced a set of track loads. The system was then resimulated using the track load data to check that all nodes could still be driven by the circuit and that no timing problems resulted.

Once the system was finally problem free, the design could be handed over to Plessey for manufacture.

5.3. General simulation and evaluation

5.3.1. General

Once a circuit or a functional block of a circuit has been coded in CLASSIC format and simulated using a set of test vectors, a number of points must be checked to ensure that the circuit or block is satisfactory. Firstly, the code should be checked by hand to ensure that it corresponds to the circuit that the designer intended. Next, the test vectors should be checked, using the display editor, to ensure that they represent actual test signals. Then, the operation of the circuit should be examined, again using the display editor, to ensure correct logical function and to examine the timing of important signals. The timing analysis should show that there are no marginal timing conditions that may cause the circuit to malfunction, given manufacturing tolerances. Also, relative timing of signals should be noted in

connection with the destination of the signals; the user of signals generated by a block may require the relative timing of different signals to be within certain limits.

5.3.2. Logic verification

In the case of simple circuit blocks or small circuits, the window analysis mode is the easiest way to check logical operation, especially if the circuit has few inputs and outputs. The adder was such a block. The waveforms in figure 5.2(a) show a basic logic test on the adder with AI and BI inputs and SI their sum.

5.3.3. Timing analysis

The most important timing consideration in the rasterizer design was always the worst case delay. This may be easily assessed for the adder in figure 5.2(b). AI and BI change at 1000 (in units of 100ps) and SI has settled by 1138; the adder delay is hence approximately 13.8ns. Note that this example is taken from a design on a slower technology than was finally used; the final adder had a simulated delay lower than 10ns.

5.4. System simulation and evaluation

5.4.1. General

The last stage of the computer aided design of a system with CLASSIC is the assembly of the system circuit using all the subsystems and blocks that have been compiled and tested individually. The procedure is basically the same as for general simulation (section 5.3 paragraph one), except that care must be taken to ensure that all signals applied to the circuit as test vectors are available as IO signals at pins. IO signals should be routed through special IO cells, to and from the pins. One must ensure that the input signal timing is realistic.

```
                    C L A S S I C   D I S P L A Y   E D I T O R

Circuit compiled using CLASSIC  compiler  version    E.2   on  3-DEC-86 at 14:58:35
Circuit file spec DUA0:[IZZARD.DIRECTORY.CLASSIC.RA3000.THESIS]ADD.CCL;7
Circuit name ADD
Compilation time units   100PS

Circuit simulated using CLASSIC  simulator  version    E.2  on  3-DEC-86 at 14:59:34
Simulation file spec DUA0:[IZZARD.DIRECTORY.CLASSIC.RA3000.THESIS]ADD.SCL;2
Simulation name ADD
Simulation period   100NS
Min_mask(01) Typ_mask(00) Max_mask(10)
Simulation to spec : Supply =   5V  10%  Temp =    0->  70
Circuit simulated with X propagation ON

ZZARD                                  16-JAN-87 14:55:56   CLASSIC display editor version  E.2              Page    1
erminal                                CLA3000   V1R2       DUA0:[IZZARD.DIRECTORY.CLASSIC.RA3000.THESIS]ADD.DAT;1
```

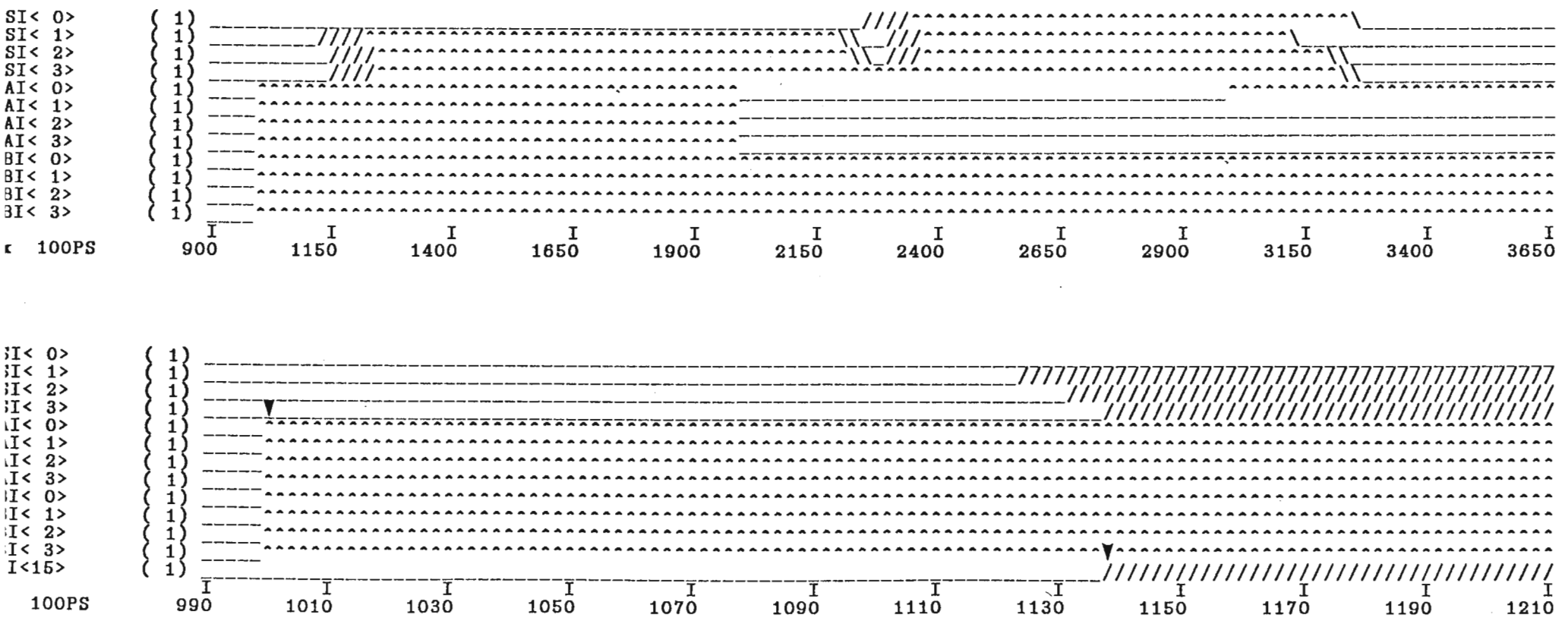


Figure 5.2(a): Adder block logic verification. (b): Adder block timing analysis.

5.4.2. Logic verification

A functional test (functional as opposed to a special test mode such as signature analysis) was used to verify the logical operation of the circuit, that is, a set of bytes representing a line was strobed in and the rasterizer was allowed to draw the line. A line from hex (000C,0003) to (0000,0000) is drawn in the example test vector set. The data in figure 5.3 is generated by the display editor in TABLE mode. The nodes selected for analysis are named at the top of the table as follows:

INST — the control machine state vector
PIXEL — the X and Y co-ordinates of the pixel location
QnBAR — the complements of various registers, n
BUSA1 and BUSA2 — the two input buses to the adder
SI — the adder output
BUSA and BUSB — buses feeding the a– and b-registers

The last four are control signals not of immediate interest.

INST is the state vector; it shows where the state machine is in the algorithm (see the state diagram, figure 3.10–12).

PIXEL is the external 32 bit pixel address; it is displayed as an X and a Y value here.

The multi-digit numbers are hexadecimal. The important data for each time step is marked. Firstly, from time 3609 to time 3653, the X1 Y1 and X2 Y2 registers are loaded; next, from time 3657 to time 3665, the line is analyzed; after that, while the machine is in state 1A, the a-register is divided by the b-register and the result stored in the a-register in state 3A (the value is 0.25 in decimal); the b-register is set to 0.5 in state 06. From time 3703, the line is drawn. The output state is 11. The algorithm terminates at the end of the line and returns to state 00 which is the start of the algorithm.

NODE	INST	PIXEL	PIXEL	QX2BAR	QY2BAR	BUSA1	BUSA2	SI	SLI	BUSA	BUSB	QABAR	QBBAR	NEG	OV	SLI	NXRA
INDX	HEX	HEX	HEX	HEX	HEX	HEX	HEX	HEX	HEX	HEX	HEX	HEX	HEX	HEX	HEX	HEX	HEX
SIM	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
I/O																	
33270000	00	0000	0006	FFFF	FFF9	0006	0006	000C	0	0000	000C	FFFF	FFF9	1	0	0	1
36090000	20	0000	0006	FFFF	FFF9	0006	0006	000C	0	0000	000C	FFFF	FFF9	1	0	0	1
36210000	10	0000	0006	FFFF	FFF9	0006	0006	000C	0	0000	000C	FFFF	FFF9	1	0	0	1
36250000	30	0000	0006	FFFF	FFF9	0006	0006	000C	0	0000	000C	FFFF	FFF9	1	0	0	1
36270000	30	000C•	0006	FFFF	FFF9	0006	0006	000C	0	0000	000C	FFFF	FFF9	1	0	0	1
36290000	08	000C•	0006	FFFF	FFF9	0006	0006	000C	0	0000	000C	FFFF	FFF9	1	0	0	1
36330000	28	000C	0006	FFFF	FFF9	0006	0006	000C	0	0000	000C	FFFF	FFF9	1	0	0	1
36350000	28	000C	0003•	FFFF	FFF9	0003	0006	0009	0	0000	0009	FFFF	FFF9	1	0	0	1
36370000	18	000C	0003•	FFFF	FFF9	0003	0006	0009	0	0000	0009	FFFF	FFF9	1	0	0	1
36410000	38	000C	0003	FFFF•	FFF9	0003	0006	0009	0	0000	0009	FFFF	FFF9	1	0	0	1
36450000	04	000C	0003	FFFF•	FFF9	0003	0006	0009	0	0000	0009	FFFF	FFF9	1	0	0	1
36490000	24	000C	0003	FFFF	FFF9	0003	0006	0009	0	0000	0009	FFFF	FFF9	1	0	0	1
36510000	24	000C	0003	FFFF	FFFF•	0003	0006	0009	0	0000	0009	FFFF	FFF9	1	0	0	1
36530000	14	000C	0003	FFFF	FFFF•	0003	0006	0009	0	0000	0009	FFFF	FFF9	1	0	0	1
36570000	34	000C	0003	FFFF	FFFF	000C•	0000•	000C•	1	000C•	000C	FFFF	FFF9	0•	1	1	0
36590000	1C	000C	0003	FFFF	FFFF	0003•	0000•	0003•	1	0001	0003•	FFF3	FFF9	0•	1	1	1
36610000	22	000C	0003	FFFF	FFFF	000C•	0003•	FFF7•	0	0003	FFF7	FFF3	FFFC	1•	0	0	1
36630000	32	000C	0003	FFFF	FFFF	0003	0003	0006	0	0003•	000C•	FFF3	FFFC	1	0	0	1
36650000	2A	000C	0003	FFFF	FFFF	0003	000C	000F	0	000C	000F	FFFC•	FFF3•	1	0	0	1
36670000	1A	000C	0003	FFFF	FFFF	0006	000C	FFFA	0•	0006	FFFA	FFFC	FFF3	1	0	0	1
36690000	1A	000C	0003	FFFF	FFFF	000C	000C	0000	1•	0000	0000	FFF9	FFF3	0	1	1	1
36710000	1A	000C	0003	FFFF	FFFF	0000	000C	FFF4	0•	0000	FFF4	FFFF	FFF3	1	0	0	1
36990000	3A	000C	0003	FFFF	FFFF	0003	000C	000F	0	4000•	000F	FFFF	FFF3	1	0	0	1
37010000	06	000C	0003	FFFF	FFFF	0003	000C	000F	0	8000	000F	BFFF	FFF3	1	0	0	1
37030000	11	000C•	0003•	FFFF	FFFF	0003	8000	8003	0	0000	8003	BFFF	7FFF	1	0	0	1
37050000	26	000C	0003	FFFF	FFFF	4000	8000	C000	0	0000	C000	BFFF	7FFF	1	0	0	1
37070000	1E	000C	0003	FFFF	FFFF	000C	0001	000B	1	0000	000B	BFFF	3FFF	0	1	1	1
37090000	11	000B•	0003•	FFFF	FFFF	0003	C000	C003	0	0001	C003	BFFF	3FFF	1	0	0	1
37110000	26	000B	0003	FFFF	FFFF	4000	C000	0000	1	0002	0000	BFFF	3FFF	0	1	1	1
37130000	16	000B	0003	FFFF	FFFF	000B	0001	000A	1	0005	000A	BFFF	FFFF	0	1	1	1

Figure 5.3: System logical operation verification using TABLE mode analysis.... continued

NODE	INST	PIXEL	PIXEL	QX2BAR	QY2BAR	BUSA1	BUSA2	SI	SLI	BUSA	BUSB	QABAR	QBBAR	NEG	OV	SLI	NXRA
INDX	HEX	HEX	HEX	HEX	HEX	HEX	HEX	HEX	HEX	HEX	HEX	HEX	HEX	HEX	HEX	HEX	HEX
SIM	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
I/O																	
37150000	0E	000A	0003	FFFF	FFFF	0003	0001	0002	1	000B	0002	BFFF	FFFF	0	1	1	1
37170000	11	000A*	0002*	FFFF	FFFF	0002	0000	0002	0	0017	0002	BFFF	FFFF	1	0	0	1
37190000	26	000A	0002	FFFF	FFFF	4000	0000	4000	0	002E	4000	BFFF	FFFF	1	0	0	1
37210000	1E	000A	0002	FFFF	FFFF	000A	0001	0009	1	005C	0009	BFFF	BFFF	0	1	1	1
37230000	11	0009*	0002*	FFFF	FFFF	0002	4000	4002	0	00B9	4002	BFFF	BFFF	1	0	0	1
37250000	26	0009	0002	FFFF	FFFF	4000	4000	8000	0	0172	8000	BFFF	BFFF	1	0	0	1
37270000	1E	0009	0002	FFFF	FFFF	0009	0001	0008	1	02E4	0008	BFFF	7FFF	0	1	1	1
37290000	11	0008*	0002*	FFFF	FFFF	0002	8000	8002	0	05C9	8002	BFFF	7FFF	1	0	0	1
37310000	26	0008	0002	FFFF	FFFF	4000	8000	C000	0	0B92	C000	BFFF	7FFF	1	0	0	1
37330000	1E	0008	0002	FFFF	FFFF	0008	0001	0007	1	1724	0007	BFFF	3FFF	0	1	1	1
37350000	11	0007*	0002*	FFFF	FFFF	0002	C000	C002	0	2E49	C002	BFFF	3FFF	1	0	0	1
37370000	26	0007	0002	FFFF	FFFF	4000	C000	0000	1	5C92	0000	BFFF	3FFF	0	1	1	1
37390000	16	0007	0002	FFFF	FFFF	0007	0001	0006	1	B925	0006	BFFF	FFFF	0	1	1	1
37410000	0E	0006	0002	FFFF	FFFF	0002	0001	0001	1	724B	0001	BFFF	FFFF	0	1	1	1
37430000	11	0006*	0001*	FFFF	FFFF	0001	0000	0001	0	E497	0001	BFFF	FFFF	1	0	0	1
37450000	26	0006	0001	FFFF	FFFF	4000	0000	4000	0	C92E	4000	BFFF	FFFF	1	0	0	1
37470000	1E	0006	0001	FFFF	FFFF	0006	0001	0005	1	925C	0005	BFFF	BFFF	0	1	1	1
37490000	11	0005*	0001*	FFFF	FFFF	0001	4000	4001	0	24B9	4001	BFFF	BFFF	1	0	0	1
37510000	26	0005	0001	FFFF	FFFF	4000	4000	8000	0	4972	8000	BFFF	BFFF	1	0	0	1
37530000	1E	0005	0001	FFFF	FFFF	0005	0001	0004	1	92E4	0004	BFFF	7FFF	0	1	1	1
37550000	11	0004*	0001*	FFFF	FFFF	0001	8000	8001	0	25C9	8001	BFFF	7FFF	1	0	0	1
37570000	26	0004	0001	FFFF	FFFF	4000	8000	C000	0	4B92	C000	BFFF	7FFF	1	0	0	1
37590000	1E	0004	0001	FFFF	FFFF	0004	0001	0003	1	9724	0003	BFFF	3FFF	0	1	1	1
37610000	11	0003*	0001*	FFFF	FFFF	0001	C000	C001	0	2E49	C001	BFFF	3FFF	1	0	0	1
37630000	26	0003	0001	FFFF	FFFF	4000	C000	0000	1	5C92	0000	BFFF	3FFF	0	1	1	1
37650000	16	0003	0001	FFFF	FFFF	0003	0001	0002	1	B925	0002	BFFF	FFFF	0	1	1	1
37670000	0E	0002	0001	FFFF	FFFF	0001	0001	0000	1	724B	0000	BFFF	FFFF	0	1	1	1
37690000	11	0002*	0000*	FFFF	FFFF	0000	0000	0000	0	E497	0000	BFFF	FFFF	1	0	0	1
37710000	26	0002	0000	FFFF	FFFF	4000	0000	4000	0	C92E	4000	BFFF	FFFF	1	0	0	1
37730000	1E	0002	0000	FFFF	FFFF	0002	0001	0001	1	925C	0001	BFFF	BFFF	0	1	1	1
37750000	11	0001*	0000*	FFFF	FFFF	0000	4000	4000	0	24B9	4000	BFFF	BFFF	1	0	0	1
37770000	26	0001	0000	FFFF	FFFF	4000	4000	8000	0	4972	8000	BFFF	BFFF	1	0	0	1
37790000	1E	0001	0000	FFFF	FFFF	0001	0001	0000	1	92E4	0000	BFFF	7FFF	0	1	1	1
37810000	11	0000*	0000*	FFFF	FFFF	0000	8000	8000	0	25C9	8000	BFFF	7FFF	1	0	0	1
37830000	00*	0000	0000	FFFF	FFFF	0000	8000	8000	0	4B92	8000	BFFF	7FFF	1	0	0	1
37850000	00	0000	0000	FFFF	FFFF	0000	8000	8000	0	9724	8000	BFFF	7FFF	1	0	0	1
37870000	00	0000	0000	FFFF	FFFF	0000	8000	8000	0	2E48	8000	BFFF	7FFF	1	0	0	1

Figure 5.3: System logical operation verification using TABLE mode analysis.

The time units are ten's of picoseconds. Note that not all clock cycles have a line on the table since repeated lines are omitted. The cycle time is 200ns.

A more complete set of results for this test, showing almost all signals of interest in the entire machine are shown in appendix 5.

5.4.3. Timing analysis

The window mode was used to verify timing of critical signals and make predictions regarding the operating speed of the circuit. Figure 5.4(a), below, shows one of the longest delay paths through the circuit, from the clock edge, through the state machine to the formation of the appropriate state machine outputs (such as bus driver enable signals), through the adder, adder result through the bus structure to settle on the register inputs. INST<n> is the present state. OUTSTn is the next state. SI<n> is the adder result. COUT is carry out of the adder and SLI is the quotient bit. BUSB<n> is the input bus for the b-register where the result of the current calculation is to be stored. ZERO and EQ are the result-zero and X1=X2 Y1=Y2 signals. The last signal to settle is the bus at 82ns after the external clock signal, XCLOCK. This simulation is done with worst case delays presumed for the gates. The second set of waveforms in figure 5.4(a) is the best-case-delay simulation. The system settles in 20ns, best case; a four fold improvement.

Another critical path is from the adder through the zero detection circuit to the next state logic and to the state register inputs (see figure 5.4(b)). The change on ZERO effects the next state OUTST2. The system settle time here is 118ns worst case and 32ns best case.

The autorouter provides a file of the capacitive track loads for each node. In many instances the load is greatly increased as may be seen in the example of node SLI which goes from 2 to 10 gate loads; INST1 goes from 52 to 86. See figure 5.5 for more examples.

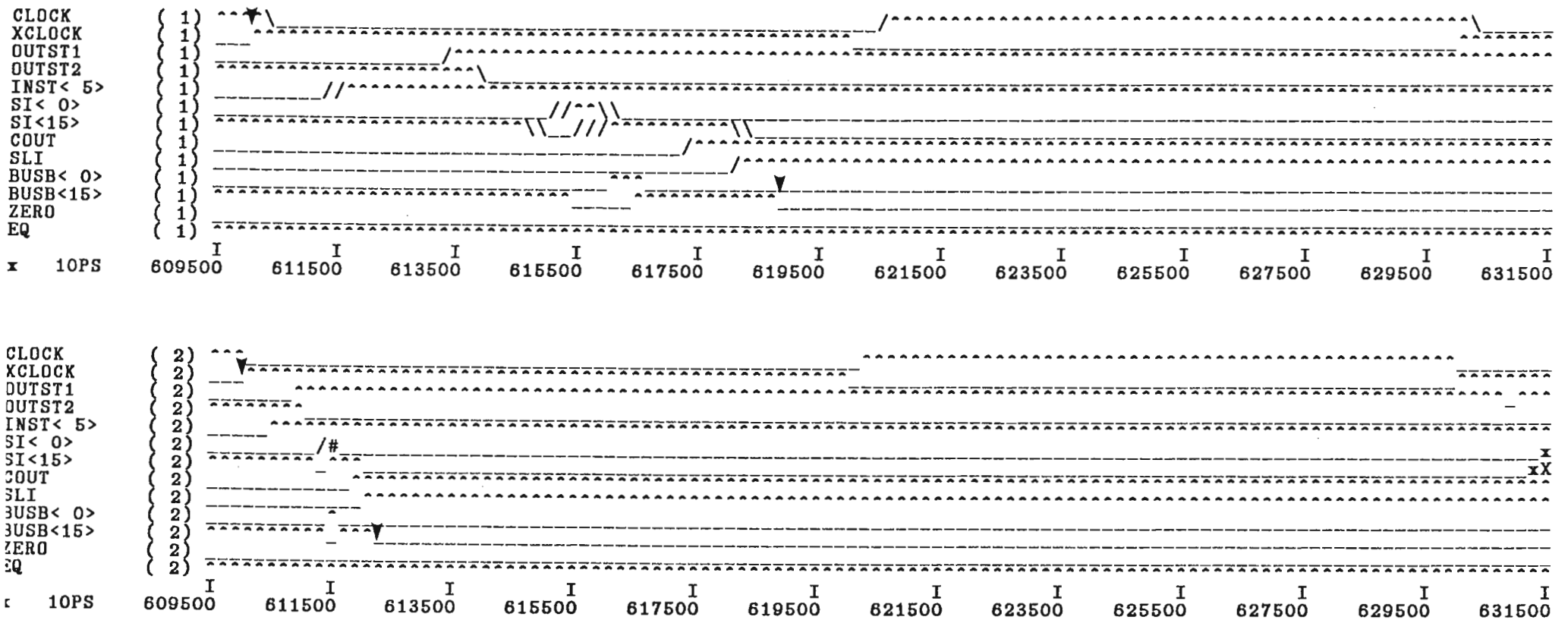


Figure 5.4(a): Critical timing path analysis1 using window mode: no track loads included.

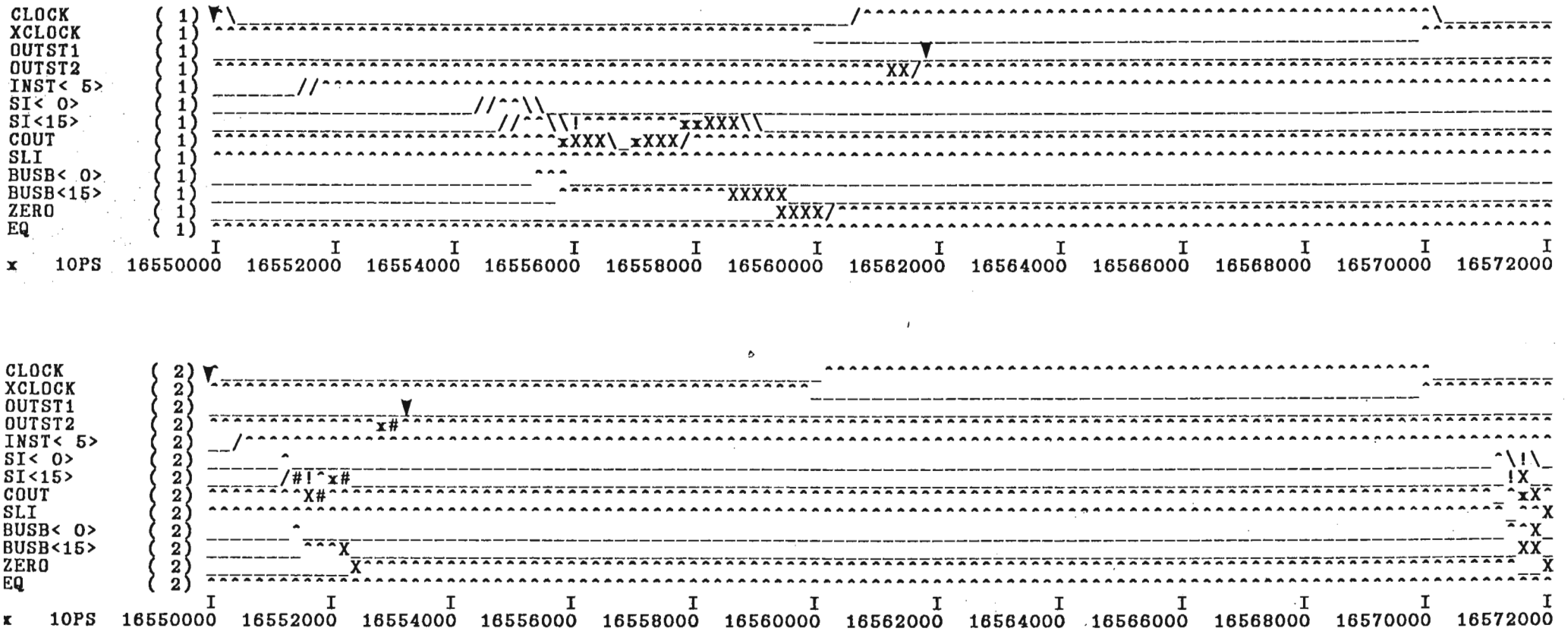


Figure 5.4(b): Critical timing path analysis2 using window mode: no track loads included.

NODAL TIMING INFORMATION

NODE	INDEX	No	INHERENT TIMING		LOAD TIMING T/LOAD		LOAD		MAX LOAD	TOTAL UNITS	TIMING 10PS
			TFALL	TRISE	TFALL	TRISE	GATE	TRACK		TFALL	TRISE
INSTO		604	110	100	2	2	21	24	200	200	190
INST1		606	110	100	2	2	52	34	200	282	272
INST2		608	110	100	2	2	53	41	200	298	288
INST3		610	110	100	2	2	48	34	200	274	264
INST4		612	110	100	2	2	73	46	200	348	338
INST5		614	110	100	2	2	53	34	200	284	274
SIBAR	14	177	110	100	10	10	9	8	50	280	270
SIBAR	15	178	110	100	10	10	9	8	50	280	270
SINX		5576	110	100	10	10	1	7	50	190	180
SINXB		5730	110	100	40	40	2	0	12	190	180
SINXU		5573	140	140	40	70	1	0	7	180	210
SINY		5577	110	100	10	10	1	7	50	190	180
SINYB		5745	110	100	40	40	2	0	12	190	180
SINYU		5572	140	140	40	70	1	0	7	180	210
SLI		951	110	100	10	10	2	8	50	210	200
SLIBAR		949	110	100	40	40	5	3	12	430	420
SLIU		1242	160	150	50	40	1	0	10	210	190
SMAN		5600	110	100	10	10	1	6	50	180	170
SMANB		5787	110	100	40	40	2	0	12	190	180
SNC1		5585	160	220	40	100	4	0	5	320	620
SOC1		5592	140	140	40	70	4	0	7	300	420
SPC1		5599	140	140	40	70	4	0	7	300	420
SQC1		5609	160	220	40	100	4	0	5	320	620
SRA		2875	110	100	10	10	1	6	50	180	170

Figure 5.5: Nodal loadings and other timing data.

The critical timing paths shown before are now longer after resimulation with the correct loadings: see figure 5.6(a) and 5.6(b). The settling times become 102ns worst and 26ns best in (a) and 132ns worst and 36ns best in (b).

The 132ns delay in figure 5.6(b) implies a worst case maximum clock rate of 7.6MHz.

5.4.4. Initial testability analysis

The program CLASP was used to try and gain some insight into the testability of the circuit. The results are shown in figure 5.7. Note that all the so called uncontrollable nodes are in fact fixed value nodes and so do not represent a problem.

5.5. Miscellaneous problems

5.5.1. Timing problems

The display editor was also used to scan the data files for potential timing problems using the RACE, GLITCH and the ERROR modes.

The race condition that the display editor searches for, in the RACE analysis, is the input to a clocked cell changing too near the clock edge. No race conditions were ever a problem unless the machine was clocked at too high a speed.

The structure of the machine is such that glitches and spikes will occur in large numbers during the settle period. This is not a problem because the machine is totally edge triggered; on the odd occasion where an output drives an asynchronous input such as the preset or reset of a register, the output is synchronized with an extra flip-flop. GLITCH analysis therefore produces numerous errors. The analysis does not help to show problems in this circuit: race detection is sufficient.

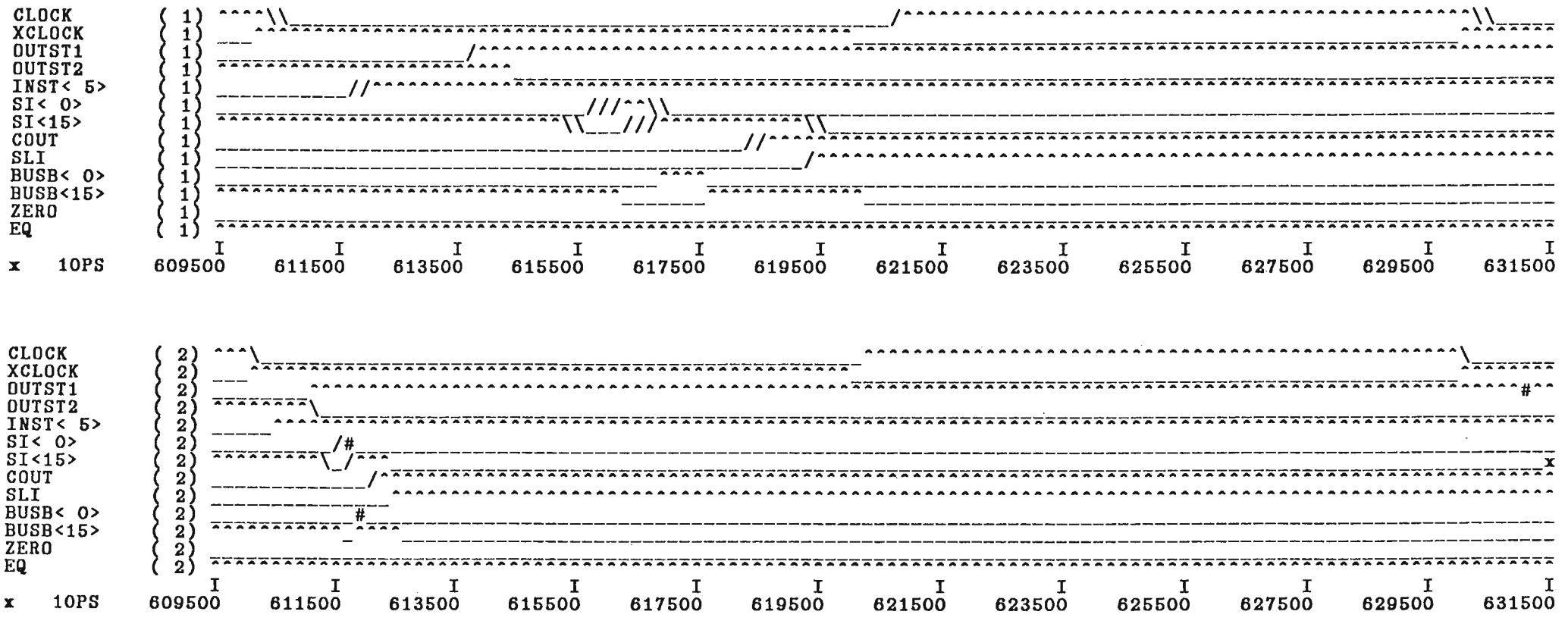


Figure 5.6(a): Critical timing path analysis1 using window mode: track loads included.

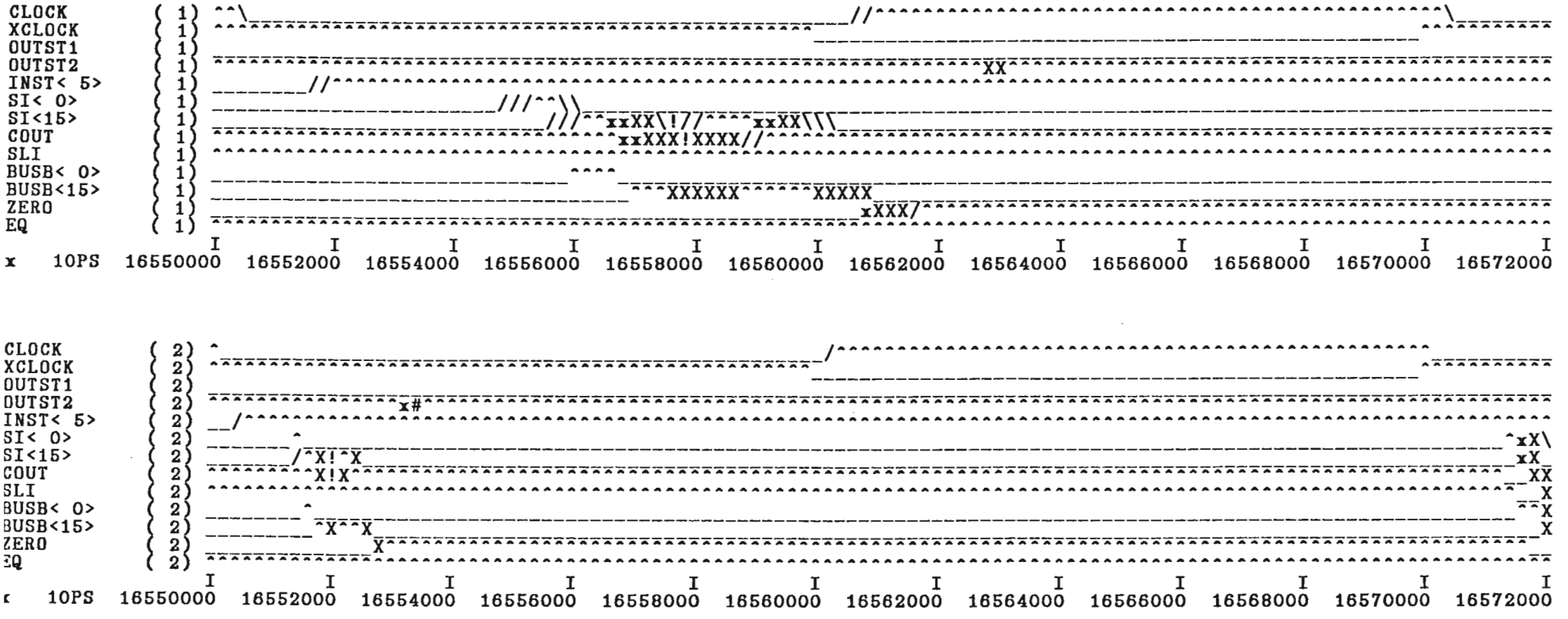


Figure 5.6(b): Critical timing path analysis2 using window mode: track loads included.

```
TEST ANALYSIS REVIEW

          Uncontrollable nodes

BUSAA<0>      BUSA11<0>     BUSA21<0>     BUSA21<1>     BUSA21<2>     BUSA21<3>
BUSA21<4>     BUSA21<5>     BUSA21<6>     BUSA21<7>     BUSA21<8>     BUSA21<9>
BUSA21<10>    BUSA21<11>    BUSA21<12>    BUSA21<13>    BUSA21<14>    BUSA21<15>
T1__OB1<0>    T1__OB2<0>    T5__OB1<0>    T5__OB2<0>    T9__OB1<0>    T9__OB1<1>
T9__OB1<2>    T9__OB1<3>    T9__OB1<4>    T9__OB1<5>    T9__OB1<6>    T9__OB1<7>
T9__OB1<8>    T9__OB1<9>    T9__OB1<10>   T9__OB1<11>   T9__OB1<12>   T9__OB1<13>
T9__OB1<14>   T9__OB1<15>   T9__OB2<0>    T9__OB2<1>    T9__OB2<2>    T9__OB2<3>
T9__OB2<4>    T9__OB2<5>    T9__OB2<6>    T9__OB2<7>    T9__OB2<8>    T9__OB2<9>
T9__OB2<10>   T9__OB2<11>   T9__OB2<12>   T9__OB2<13>   T9__OB2<14>   T9__OB2<15>
Total number of uncontrollable nodes :    54

          Unobservable nodes

Total number of unobservable nodes :     0

Max value for com cont :    30517 Node name : P5C4
Max value for seq cont :     4447 Node name : P5B10
Max value for com obsv :    30675 Node name : CC1_INCK
Max value for seq obsv :     4467 Node name : CC1_ENB


TEST ANALYSIS SUMMARY

Number of circuit nets = 2495
Total number of uncontrollable or unobservable nodes =    54
This is    2.1% of total number of nodes
Combinational uncontrollable 0 nodes =     3
Combinational uncontrollable 1 nodes =    51
Combinational unobservable nodes =     0
Sequential uncontrollable 0 nodes =     3
Sequential uncontrollable 1 nodes =    51
Sequential unobservable nodes =     0
Number of primary inputs =    13
Number of primary outputs =    44
Number of test points =     0
Number of fanout branches = 5552
Number of feedback loops or reconvergent fanouts = 3057
Total fault population = 4990
```

Figure 5.7: Test analysis with CLASP.

The ERROR analysis mode turned up the set of clock-skew errors shown in figure 5.8. These errors are indicated as causing a problem on the select lines of the transmission gates that act as tristate bus drivers. The select lines of a transmission gate need to be driven by clock drivers. The condition on the lines is, however, incorrectly interpreted by CLASSIC. Close inspection shows that there are in fact narrow decoding spikes on the select lines: see figure 5.9. X2A2 is such a select line and the T11__CKn signals are outputs from the clock driver for X2A2. The spikes are not acceptable for normal clocked elements such as flip-flops but are no problem for the drivers. The errors can therefore be ignored.

5.5.2. Component constraints

The most severe difficulty experienced, caused by the limited flexibility of gate array components, was matching gate loads with adequate drive capability. In many cases several levels of inverters are required to bolster the drive sufficiently.

The problem is especially serious in the case of the transmission gate. The only component that may be used as a tristate bus driver is the transmission gate. The transmission gate has no drive capability of its own. The node it is driving must be driven by the gate driving the transmission gate; however, the transmission gate can only carry a small current and so the designer is discouraged from driving it with more than one small inverter. The gates may be paralleled to improve the problem, but this solution offers diminishing returns since, the transmission gate itself is so capactive that the bus node (that is already difficult to drive) is made more difficult to drive. The transmission gate is bidirectional and hence, not only does its own capacitance add to the driven node, but, when switched on, the capacitance of the driving node also adds to the driven node.

The result of all this is that, if the transmission gate is used as a tristate bus driver, the philosophy behind using buses must be altered. Busses are supposed to be long nodes that carry a variety

```
ERROR ANALYSIS
Tmin =          0 Tmax =  40000000 Sim =  1

CK_SKEW  error in block T11_G3   at time    574384
CK_SKEW  error in block T11_G3X  at time    574384
CK_SKEW  error in block T2__G3   at time    575102
CK_SKEW  error in block T2__G3X  at time    575102
CK_SKEW  error in block T10_G3   at time    575104
CK_SKEW  error in block T10_G3X  at time    575104
CK_SKEW  error in block T4__G3   at time    575872
CK_SKEW  error in block T4__G3X  at time    575872
CK_SKEW  error in block T11_G3   at time   4574384
CK_SKEW  error in block T11_G3X  at time   4574384
CK_SKEW  error in block T2__G3   at time   4575102
CK_SKEW  error in block T2__G3X  at time   4575102
CK_SKEW  error in block T10_G3   at time   4575104
CK_SKEW  error in block T10_G3X  at time   4575104
CK_SKEW  error in block T4__G3   at time   4575872
CK_SKEW  error in block T4__G3X  at time   4575872
CK_SKEW  error in block T11_G3   at time  20574384
CK_SKEW  error in block T11_G3X  at time  20574384
CK_SKEW  error in block T2__G3   at time  20575102
CK_SKEW  error in block T2__G3X  at time  20575102
CK_SKEW  error in block T10_G3   at time  20575104
CK_SKEW  error in block T10_G3X  at time  20575104
CK_SKEW  error in block T4__G3   at time  20575872
CK_SKEW  error in block T4__G3X  at time  20575872
CK_SKEW  error in block T11_G3   at time  24574384
CK_SKEW  error in block T11_G3X  at time  24574384
CK_SKEW  error in block T2__G3   at time  24575102
CK_SKEW  error in block T2__G3X  at time  24575102
CK_SKEW  error in block T10_G3   at time  24575104
CK_SKEW  error in block T10_G3X  at time  24575104
CK_SKEW  error in block T4__G3   at time  24575872
CK_SKEW  error in block T4__G3X  at time  24575872

 Total number of errors found =     32
```

Figure 5.8: Results of error analysis on the rasterizer system.

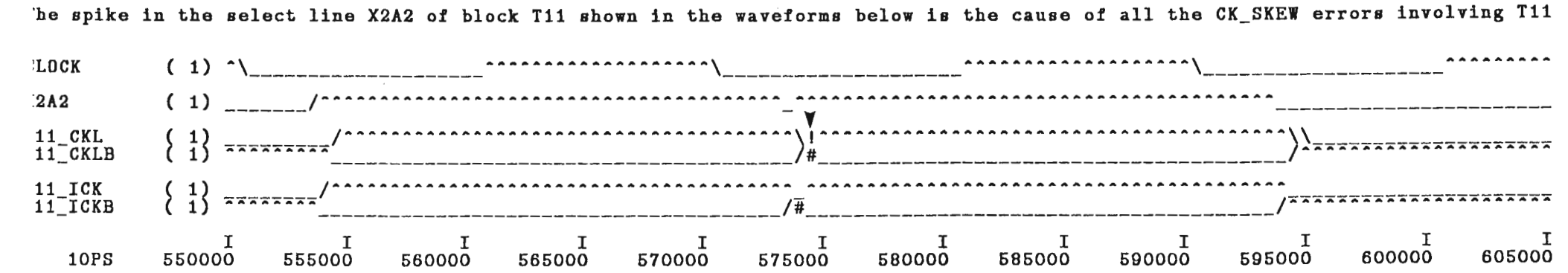


Figure 5.9: The spikes causing the problems on the select lines.

of signals around a circuit, using little track space in comparison to tracking all those signals around separately. Now, because of the problem with driving the bus nodes the buses must be made short and near the elements they are driving, while the variety of signals must be tracked all the way to the bus. This poses a grave routing difficulty and reduces busing to multiplexing.

Another interesting difference between standard busing and transmission gate busing is the issue of tristating the bus. It is poor practice to allow a bus to remain tristate in CMOS circuits because the p– and n-branches of the driven gates may both end up on, causing a large current drain on the power supply. Hence, one bus user must be selected at all times.

5.6. Test vector generation

The modern ideal for IC testing is to build the circuit with switchable data paths and dual purpose IO pins; the circuit has a test mode in which individual blocks are tested separately. The test sequences that result from this structure are very much reduced because no circuit nodes are too deep in the circuit to be inaccessible. That method was not used in the case of the rasterizer, however, because it requires a large number of extra multiplexers and other ancillary circuitry to implement, and the available gate arrays were too small. The alternative is clearly to do a functional test and check that the results are correct. This system can result in very long test vector sets (more than 50 000 vectors). In the case of the rasterizer the length of a good set of vectors is only about 2000; however, a test that at least exercises the most significant bits of all the registers and data paths requires well over 64k vectors because a very long line must be drawn.

The functional test consists of drawing the set of vectors in figure 5.10. The set completely tests the state machine and control logic, but does not exercise every bit in the data structure.

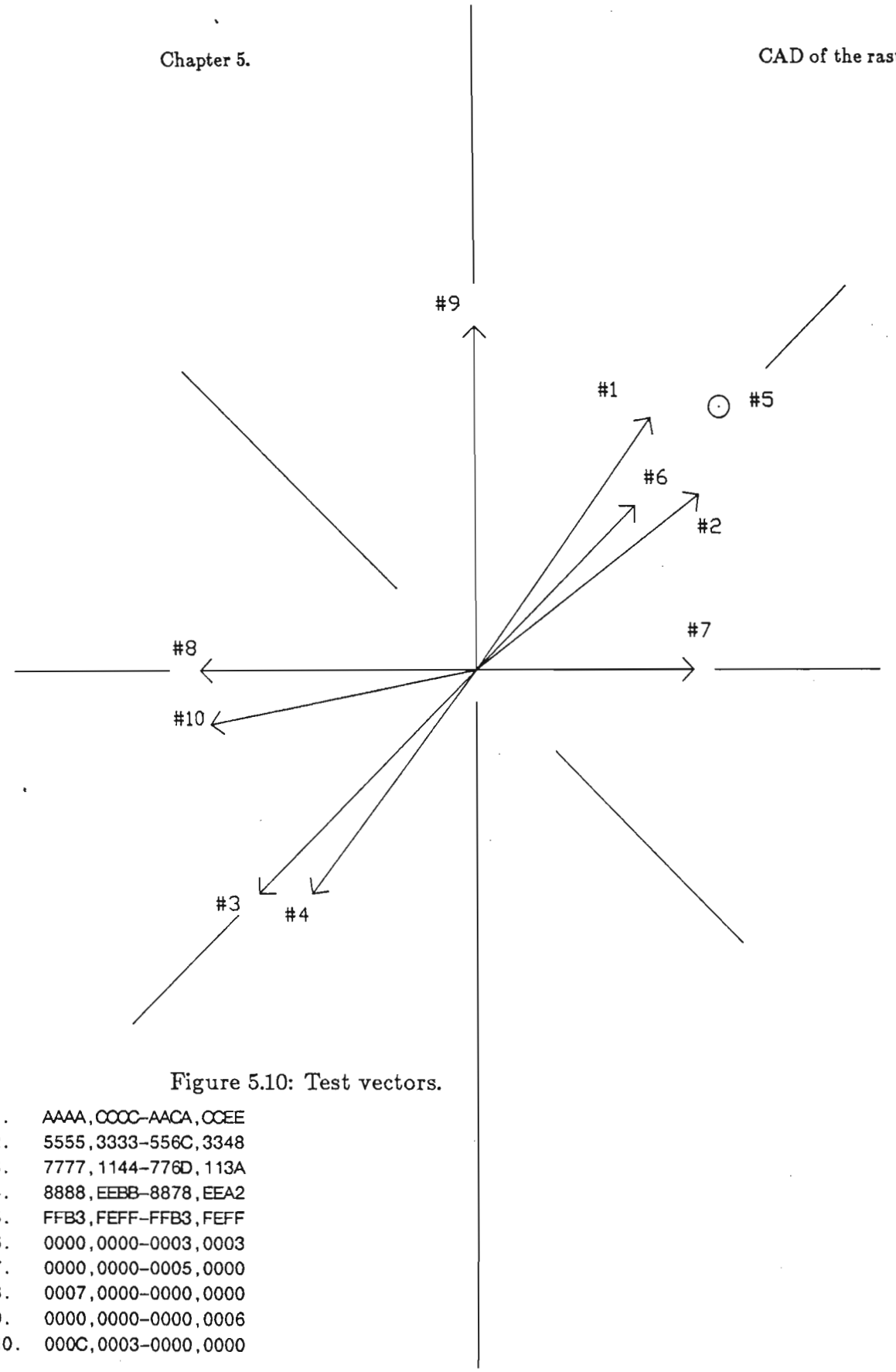


Figure 5.10: Test vectors.

1. AAAA,CCCC-AACA,CCEE
2. 5555,3333-556C,3348
3. 7777,1144-776D,113A
4. 8888,EEBB-8878,EEA2
5. FFB3,FEFF-FFB3,FEFF
6. 0000,0000-0003,0003
7. 0000,0000-0005,0000
8. 0007,0000-0000,0000
9. 0000,0000-0000,0006
10. 000C,0003-0000,0000

5.7. Test vector evaluation

It is possible from a knowledge of the circuit to judge how much of the circuit is tested by a set of vectors but this is not strictly sufficient. The CLASSIC suite provides two ways of evaluating the effectiveness of a vector set. The first, and least time consuming, is an EVENTS analysis of simulation data by the display editor. This analysis lists all untoggled nodes in the circuit. The EVENTS data for the rasterizer test vector set is shown in figure 5.11. There is a large number of nodes that are not toggled simply because they have fixed inputs (these are in fact the uncontrolable nodes in figure 5.7). The remainder are not toggled due to an inexhaustive test vector set.

The fault simulation is the second method of test evaluation. The process (as described in chapter 4) is very time consuming even though an option exists to cover only a random selection of vectors and allow the display editor to infer the actual fault cover. The summary-results of three random fault simulations are given in figure 5.12. Note that undefined faults are those that cause outputs to be in the state X and can usually be detected. Hence, the results indicate that fault cover by the test vectors is between 91% and 97%, which is a more than acceptable level.

:VENTS ANALYSIS

93.1% of listed nodes were toggled in the time specified

TABLE OF LISTED NODES NOT TOGGLED

BUSA11< 0>	BUSA11< 7>	BUSA11< 8>	BUSA11< 9>	BUSA11<10>	BUSA11<11>	BUSA11<12>	BUSA11<13>	BUSA11<14>	BUSA11<15>
BUSA12<10>	BUSA21< 0>	BUSA21< 1>	BUSA21< 2>	BUSA21< 3>	BUSA21< 4>	BUSA21< 5>	BUSA21< 6>	BUSA21< 7>	BUSA21< 8>
BUSA21< 9>	BUSA21<10>	BUSA21<11>	BUSA21<12>	BUSA21<13>	BUSA21<14>	BUSA21<15>	BUSAA< 0>	BUSAA< 7>	BUSAA< 8>
BUSAA< 9>	BUSAA<10>	BUSAA<11>	BUSAA<12>	BUSAA<13>	BUSAA<14>	BUSAA<15>	BUSAC< 5>	BUSAC< 6>	BUSAC< 7>
BUSAC< 8>	BUSAC< 9>	BUSAC<10>	BUSAC<11>	BUSAC<12>	BUSAC<13>	BUSAC<14>	BUSAC<15>	BUSBA< 5>	BUSBA< 6>
BUSBA< 7>	BUSBA< 8>	BUSBA< 9>	BUSBA<10>	BUSBA<11>	BUSBA<12>	BUSBA<13>	BUSBA<14>	BUSBA<15>	T13_OB1< 5>
T13_OB1< 6>	T13_OB1< 7>	T13_OB1< 8>	T13_OB1< 9>	T13_OB1<10>	T13_OB1<11>	T13_OB1<12>	T13_OB1<13>	T13_OB1<14>	T13_OB1<15>
T13_OB2< 5>	T13_OB2< 6>	T13_OB2< 7>	T13_OB2< 8>	T13_OB2< 9>	T13_OB2<10>	T13_OB2<11>	T13_OB2<12>	T13_OB2<13>	T13_OB2<14>
T13_OB2<15>	T1__OB1< 0>	T1__OB1< 7>	T1__OB1< 8>	T1__OB1< 9>	T1__OB1<10>	T1__OB1<11>	T1__OB1<12>	T1__OB1<13>	T1__OB1<14>
T1__OB1<15>	T1__OB2< 0>	T1__OB2< 7>	T1__OB2< 8>	T1__OB2< 9>	T1__OB2<10>	T1__OB2<11>	T1__OB2<12>	T1__OB2<13>	T1__OB2<14>
T1__OB2<15>	T3__OB1< 5>	T3__OB1< 6>	T3__OB1< 7>	T3__OB1< 8>	T3__OB1< 9>	T3__OB1<10>	T3__OB1<11>	T3__OB1<12>	T3__OB1<13>
T3__OB1<14>	T3__OB1<15>	T3__OB2< 5>	T3__OB2< 6>	T3__OB2< 7>	T3__OB2< 8>	T3__OB2< 9>	T3__OB2<10>	T3__OB2<11>	T3__OB2<12>
T3__OB2<13>	T3__OB2<14>	T3__OB2<15>	T5__OB1< 0>	T5__OB1< 7>	T5__OB1< 8>	T5__OB1< 9>	T5__OB1<10>	T5__OB1<11>	T5__OB1<12>
T5__OB1<13>	T5__OB1<14>	T5__OB1<15>	T5__OB2< 0>	T5__OB2< 7>	T5__OB2< 8>	T5__OB2< 9>	T5__OB2<10>	T5__OB2<11>	T5__OB2<12>
T5__OB2<13>	T5__OB2<14>	T5__OB2<15>	T6__OB1<10>	T6__OB2<10>	T9__OB1< 0>	T9__OB1< 1>	T9__OB1< 2>	T9__OB1< 3>	T9__OB1< 4>
T9__OB1< 5>	T9__OB1< 6>	T9__OB1< 7>	T9__OB1< 8>	T9__OB1< 9>	T9__OB1<10>	T9__OB1<11>	T9__OB1<12>	T9__OB1<13>	T9__OB1<14>
T9__OB1<15>	T9__OB2< 0>	T9__OB2< 1>	T9__OB2< 2>	T9__OB2< 3>	T9__OB2< 4>	T9__OB2< 5>	T9__OB2< 6>	T9__OB2< 7>	T9__OB2< 8>
T9__OB2< 9>	T9__OB2<10>	T9__OB2<11>	T9__OB2<12>	T9__OB2<13>	T9__OB2<14>	T9__OB2<15>			

Figure 5.11: Event analysis for test vector set.

```
            * CLASSIC RANDOM FAULT ANALYSIS *     TMIN=    18000  STEP=   20000

    * RESULTS OF STATISTICAL ANALYSIS #1 *
      ----------------------------------


Total fault population = 3210
Number of faults analysed =  266
Percentage of faults analysed =  8
90% Confidence limits are  75- 83 %
This will be increased to  94- 97 % if all undefined faults are detected


            * CLASSIC RANDOM FAULT ANALYSIS *     TMIN=    18000  STEP=   20000

    * RESULTS OF STATISTICAL ANALYSIS #2 *
      ----------------------------------


Total fault population = 3210
Number of faults analysed =  266
Percentage of faults analysed =  8
90% Confidence limits are  80- 86 %
This will be increased to  91- 96 % if all undefined faults are detected


            * CLASSIC RANDOM FAULT ANALYSIS *     TMIN=    18000  STEP=   20000

    * RESULTS OF STATISTICAL ANALYSIS #3 *
      ----------------------------------


Total fault population = 3210
Number of faults analysed =  266
Percentage of faults analysed =  8
90% Confidence limits are  82- 88 %
This will be increased to  94- 97 % if all undefined faults are detected
```

Figure 5.12: Results of three random fault analyses on the rasterizer system.

5.8. Placement and routing

The main circuit description file in appendix 3 is summarized by the compiler (in the .CDL file) as follows:

Total number of logic blocks used = 752

Total number of array elements used = 3442

Utilisation is 78% on a 4408 gate array

Total number of peripheral cells = 57

Total number of circuit nets = 2554

Total number of nodes generated = 5822

Total number of bytes of machine code generated = 307268

Each of the 752 logic blocks mentioned above required hand placement by assigning it a co-ordinate on the grid of array elements (clearly some logic blocks use multiple array elements). The input and output nodes are each assigned by hand to a peripheral cell. There were 57 signal inputs or outputs. The remainder of the 68 pins available on the package were assigned to power connections 3 V+ and 8 V−. The "nets" above are electrical nodes and the "nodes" are the physical nodes or points on the array that have to be interconnected. The greatest number of nodes on a net is 90 and the least is 1.

The rasterizer consists of two distinct types of circuitry. The first is the control block, which is a six bit register, and a large block of random logic (disorderly groupings of NAND, NOR, etc cells). The logic implements the next state and output logic equations (here this logic replaces a PLA type logic structure). The second is the arithmetic architecture, which is a very orderly set of logic and 16 bit registers connected by five 16 bit buses and their associated tristate drivers. These two types of structures are the most

difficult to route because of the random logic in the one case, and the wide buses in the other case. The array is 78% utilized — this is a very high utilization for a gate array especially in view of the structures involved.

Firstly the library components were placed on an independent co-ordinate system so that they could be placed on the array as macro-cells (see appendix 5 for the layouts of the library). Placement was done by hand on the main array, with the two main blocks separated slightly; they may be identified on plate 5.1 — the random logic in the top righthand corner.

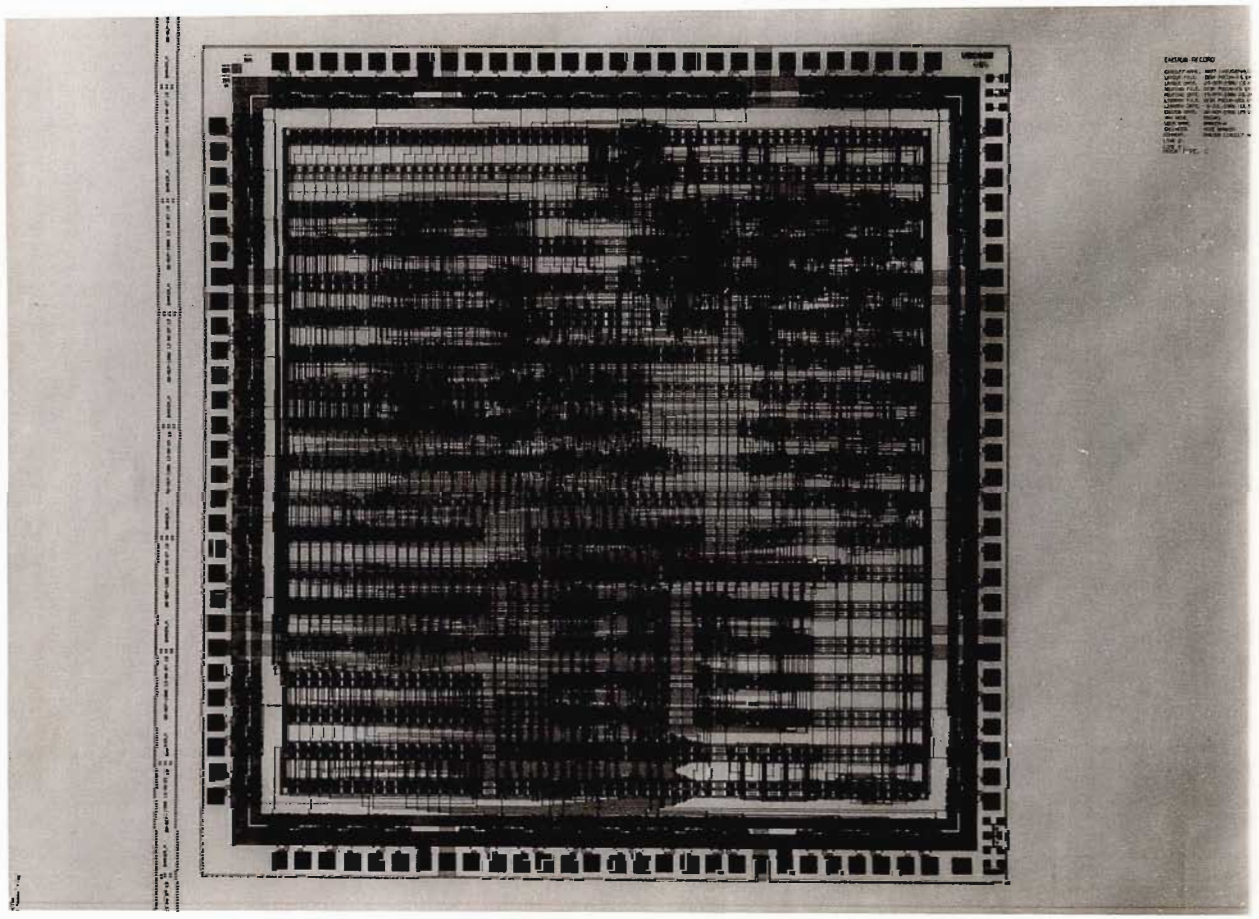
The array was successfully autorouted after four iterations and without use of net prioritizing (in each case net prioritizing caused the solution to be less complete than without). Plate 5.1 shows the metal-1 and metal-2 plot. Successful routing was achieved by careful analyzing of problem areas and shifting blocks of cells to create space. An important factor contributing to the success of the routing was that the interconnection channel width, for the array used, was 16 tracks, so that a bus could fit in one channel. If the channel was narrower the route would have failed at such a high utilization.

Apart from the essential IO signals assigned to pins, the state vector was also routed via IO cells to pins to aid evaluation and debugging of the completed IC. The IC required a 68 pin PLCC package with 57 signal pins and 11 power supply pins (3 V+ and 8 V-). The pin assignments are shown on figure 5.13.

5.9. Design handover

Ultimately, the circuit was simulated with a full set of test vectors and including track loadings, ensuring that no artificial conditions existed, that would make the simulation optimistic. These conditions are, for instance, circuit initialization at nodes internal to the circuit, or simulation with XPROP off. The history file is used by the Plessey test engineers during probe testing of the processed wafers. At a certain time in each clock cycle

Plate 5.1: Plot of metalization layers.



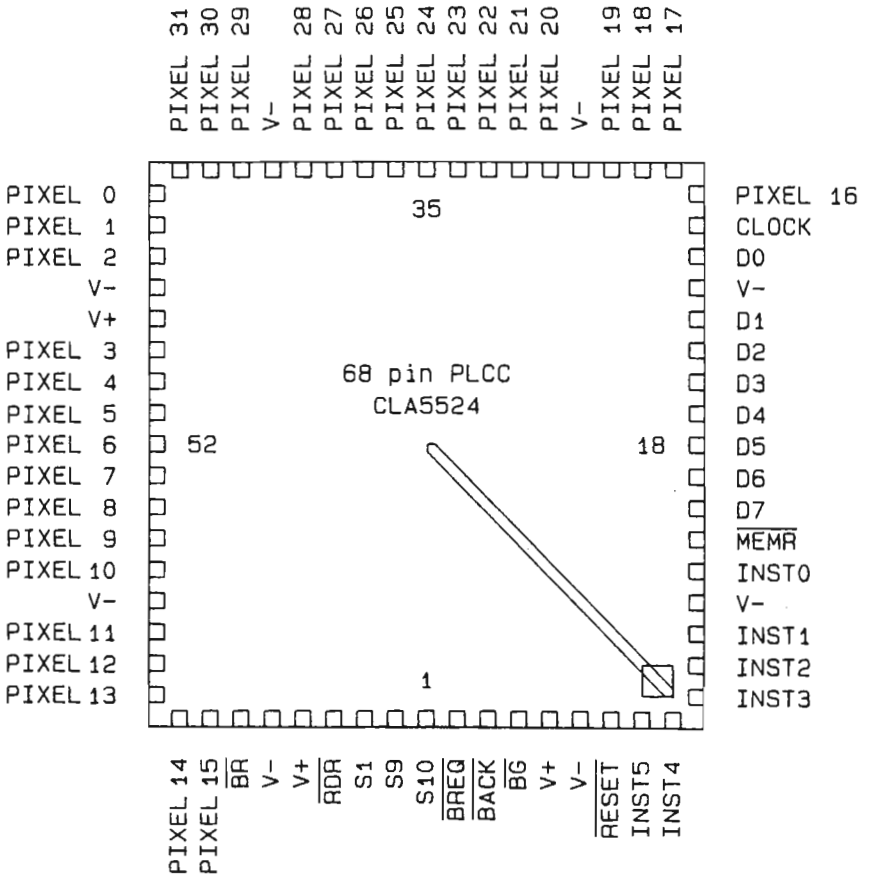


Figure 5.13: Pinout diagram.

(usually 70% of cycle) the history file is compared with the device outputs and if the results compare favourably, the device is passed. The devices received by the customer are therefore guaranteed to perform in the same way that the simulation performs.

The completed design with the routed array in digital form, and circuit description files, test vectors and final history files were handed over to Plessey who delivered working prototypes for evaluation after eight weeks. Part of the handover process with Plessey is the completion of "Design Reviews" in which the design process is checked by a senior engineer and the conditions of the contract between Plessey and the customer are laid down in writing. The final design review for CLA5524 (CLA5524 is the serial number for the rasterizer array) is reprinted in appendix 7.

Chapter 6.
Rasterizer evaluation: results and conclusion

6.1. Semicustom IC testing

A completed semicustom IC must be tested to ensure that the fabrication process has been physically successful. The IC must also be tested in its working environment to ensure that it meets the functional specifications. Apart from these two major tests there are other minor tests that may be performed. For instance, the device may be tested to ensure that the relative timing of certain signals is within specification. In some cases the supply current drawn by the device may be a critical parameter.

6.2. Physical process verification

The functional tests are only necessary in the development phase of a device, while the physical tests are always necessary, even in the production phase of the life of a product. It is for this reason that "design for testability" is very important for a device intended for high volume production, since easy testability (physical process verification) can cut costs. Many semicustom designs are products in the development phase or are not intended for large volume production. For this reason the physical process verification tests are less important for semicustom (although they are still essential) and hence so called "design for testability" is a secondary consideration. This is true for the rasterizer. The verification of the success of the physical process in the case of the rasterizer was done by the Plessey test engineers on a Teredyne test bed using the set of functional tests (the test vectors) mentioned in the previous chapter.

Ultimately, the circuit was simulated with a full set of test vectors, while including track loadings in the circuit model, and ensuring that no artificial conditions existed that would make the simulation optimistic. These conditions are, for instance, circuit

initialization at nodes internal to the circuit, or simulation with XPROP off. The history file was then used by the test engineers during testing of the processed arrays. The tests are done as follows: at a certain instant in each clock cycle (usually 70% of cycle) the simulated outputs in the history file are compared with the real device outputs and if the results compare favourably for the whole test vector set, the device is passed. The devices received by the author are therefore guaranteed to perform in the same way that the simulated system performed.

6.3. Functional tests

Clearly the tests done by the Plessey test engineers also serve as a functional test although their effectiveness is limited by the accuracy and faithfulness of the simulation. There is a possibility, for instance, that the simulated inputs are not realistic or do not properly represent the environment in which the device is required to work. It remains to test the IC in the system for which it is intended. This would constitute the most effective functional test.

At the time of writing this document, the target system was still under construction and so the author built a small circuit to further test the functionality of the device in a real environment as well as to check the findings of the Plessey test engineers. The rasterizer is required to accept data from an INTEL 8237 DMA controller. A IBM PC printer port was used as a simple way of simulating the stream of data (apart from the speed) from the DMA controller. See figure 6.1 for a schematic of the test jig. Simple non-inverting buffers were sufficient glue logic on the PC side. A TEKTRONIX 7D02 logic analyzer and a PHILLIPS pulse generator were used to control the device and analyze the resulting output data, respectively. The IBM DOS program PRINT was used to dump the appropriate bytes to the rasterizer. Plate 6.1 shows the test setup.

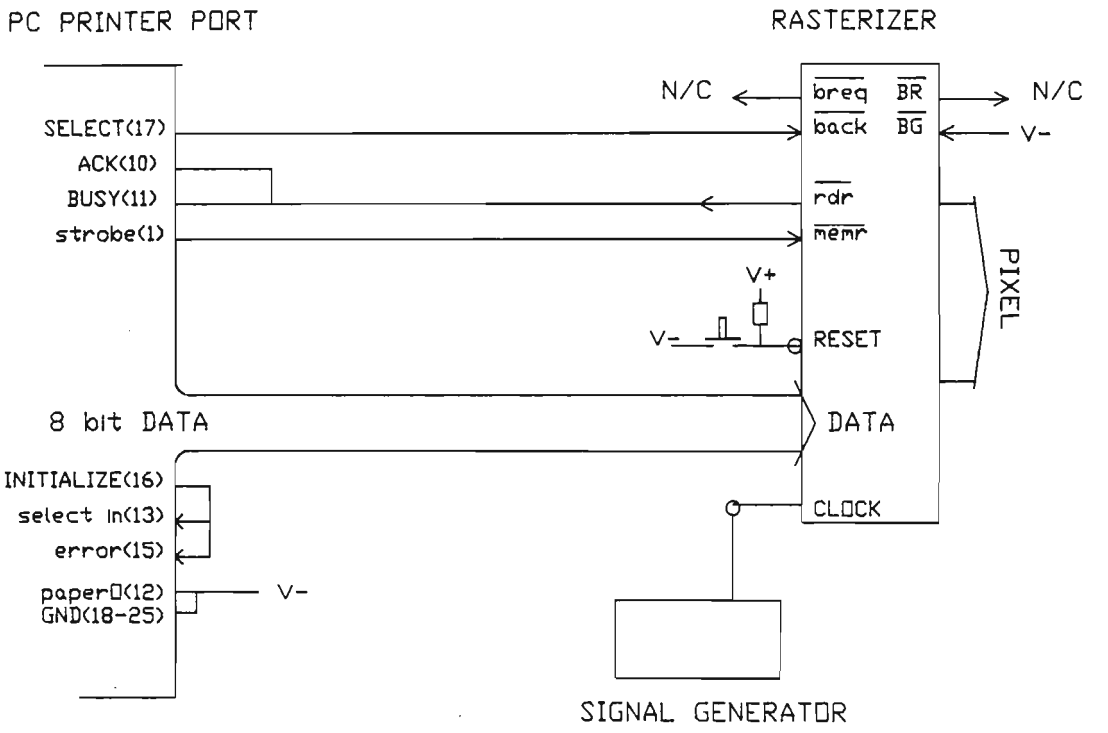
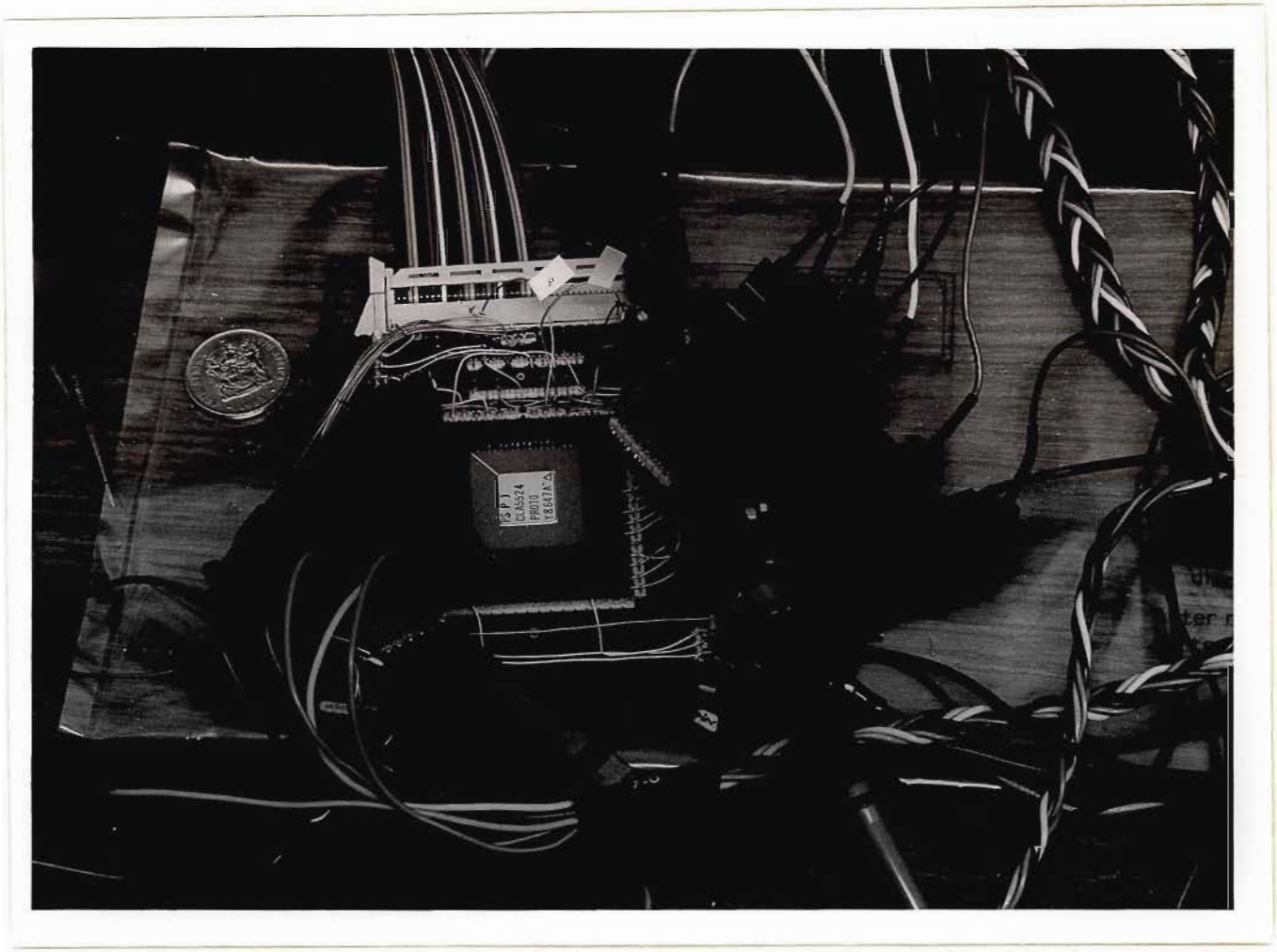


Figure 6.1: Schematic of test jig.

Plate 6.1: Photograph of test jig.



6.4. Results

The rasterizer was seen to gather data correctly. The rasterization of a range of different types of lines, manhattan and general, short and long, and lines of different direction, was tested. The various algorithm terminations were verified and all branches exercised. The flow control on the pixel bus was checked. A major aid in functional verification was the availability of the state vector in the form of extra output lines.

Although the special test bed was not optimised for performance, the device performed correctly with a 10 MHz clock (design specification 8 MHz) which results in a maximum rasterization speed of 5 MHz for 0° and 90° lines and between 2.5 MHz and 3.3 MHz for general lines.

It is expected that the chip will perform well in the overall graphics display system. The chip has been handed over to the "systems group" and will be incorporated into the graphics display system which is scheduled for testing in the second half of 1987.

Conclusion.
A graphics rasterizer IC.

The objective of this thesis was the design of a semicustom IC to improve the redraw response of a graphics display system in a simple and flexible way.

Two basic bottlenecks were identified: firstly, rasterization of the basic display elements or primitives, which amounts to a lack of raw processor power; secondly, the fundamental limitation of memory bandwidth on the speed at which pixels (the results of rasterization) may be written into the pixel memory.

The solution proposed was to provide some specialized hardware in the form of an IC to do the rasterization. The hardware was to remain simple and flexible.

The concept of parallel rasterization in which individual rasterizers queue for graphics primitives was developed. The rasterizers work independently while they can. They then queue to access the pixel memory bus in order to store the results of their work.

A line rasterizer of this nature was designed and implemented on a Plessey CLA5000 gate array. The array was completed in the United Kingdom by the author and manufactured by Plessey Semiconductors Limited also in the United Kingdom.

Plessey delivered ten prototypes. The chips had been tested on a Teredyne tester using the simulated test vectors and results supplied.

Further tests by the author verified the functionality of the rasterizer and showed that it is capable of rasterizing manhattan lines at speeds of up to 5 million pixels per second, and general lines at speeds of 3 million pixels per second.

The rasterizer design is complete and has been shown to operate according to specification. The IC's have been handed over to a system-design group for incorporation into a graphics display system.

[Beatt82] Beatty, J. C., Booth, K. S., eds. 1982. *Tutorial, computer graphics.* 2nd ed. Los Angeles: IEEE,Computer Society.

[Beres83] Beresford, Roderic. Comparing Gate Arrays and Standard-Cell ICs. *VLSI Design.* December 1983, pp. 31–36.

[Bere84a] Beresford, Roderic. Evaluating Gate-Array Techonologies. *VLSI Design.* February 1984, pp. 34–39.

[Bere84b] Beresford, Roderic. Gate-Array and Standard-Cell Design Methods. *VLSI Design.* May 1984, pp. 42–47.

[Berge84] Berger, Michael. Graphics system draws 200,000 short vectors in 1s. *Electronics.* 17 May 1984, pp.5E–6E.

[Clark82] Clark, James H. The Geometry Engine: A VLSI geometry system for graphics. *Computer Graphics (Proc. Siggraph 82).* Vol.16, No.3. July 1982, pp. 127–133.

[Cole__85] Cole, Bernard C. A chip business that is still growing. *Electronics.* 22 July 1985, pp. 40–45.

[Cole__86] Cole, Bernard C. Stretching the limits of ASIC software. *Electronics.* 23 June 1986, pp. 34–38.

[Comer84] Comer, David J. 1984. *Digital logic and state machine design.* New York: Holt, Rinehart and Winston.

[Conra85] Conrac Division, Conrac Corporation. 1985. *Raster graphics handbook.* 2nd ed. New York: Van Nostrand Reinhold.

[Conwa80] Conway, L., Mead, C. 1980. *Introduction to VLSI systems.* Addison-Wesley.

[DePal83] De Palma, G., Olson, M., Jollis, R. Dedicated VLSI chip lightens graphics display design load. *Electronic Design.* 20 January 1983.

[Davis85] Davis, Dwight B. Chip-based Graphics. *High Technology.* February 1985, pp. 46–53.

[Einsp82] Einspruch, N. G., ed. 1982. *VLSI Electronics microstructure science.* Academic Press.

[Fletc80] Fletcher, William I. 1980. *An engineering approach to digital design.* New Jersey: Prentice-Hall.

[Folbe84] Folberth, O. G., Grobman, W. D., eds. 1984. *VLSI, technology and design.* New York: IEEE Press.

[Foley82] Foley, J. D., Van Dam, A. 1982. *Fundamentals of interactive computer graphics.* Reading, Mass.: Addison-Wesley.

[Green82] Greenberg, D., Marcus, A., Schmidt, A. H., Gorter, V. 1982. *The computer image: applications of computer graphics.* Addison-Wesley.

[Gutta86] Guttag, K., Van Aken, J., Asal, M. Requirements for a VLSI Graphics Processor. *IEEE Computer Graphics and Applications.* January 1986, pp.32–47.

[Hamac83] Hamachi, Gorden. 1983. *Designing finite state machines with PEG.* University of California at Berkley.

[Hanse86] Hansen, R., Randall, M. INTEL designs a graphics chip for both CAD and business use. *Electronics.* 19 May 1986, pp. 57–60.

[Hende86] Henderson, D., Robbins, J., Herbert, B., Hackney, D., Lahey, M., Johnson, S. NCR aims its graphics chips at PC instead of work station. *Electronics.* 19 May 1986, pp. 61–63.

[Hwang79] Hwang, Kai. 1979. *Computer Arithmetic: principles, architecture, and design.* Wiley.

[Intel86] Intel Corporation. 1986. *Intel Microsystem Component Handbook, Microprocessors Volume 1.* Intel Literature Sales.

[Kisne84] Kisner, M., Ladd, J. A new-generation video processor boosts resolution. *Electronics.* 28 June 1984, pp. 121–124.

[Kraft79] Kraft, G. D., Toy, W. N. 1979. *Mini/Microcomputer hardware design.* Prentice-Hall.

[Kraft81] Kraft, G. D., Toy, W. N. 1981. *Microprogrammed control and reliable design of small computers.* Prentice-Hall.

[Lewin85] Lewin, Douglas. 1985. *Design of logic systems.* Wokingham, Berks.: Van Nostrand Reinhold.

[Lineb86] Lineback, J. R. The scramble to win in graphics chips. *Electronics.* 19 May 1986, pp. 64–65.

[Manue84] Manuel, Tom. Computer graphics. *Electronics.* 28 June 1984, pp. 113–117.

[Motor82] Motorola Semiconductor Products Inc. *MCM6256 Product Preview.* 1982, pp.1–11.

[Mowle76] Mowle, F. J. 1976. *A systematic approach to digital logic design.* Addison-Wesley.

[Mufti83] Mufti, A. A. 1983. *Elementary computer graphics.* Reston, Va.: Reston.

[Myers84] Myers, Ware. Staking out the Graphics Display Pipeline. *IEEE Computer Graphics & Applications.* July 1984, pp. 60–65.

[Newki83] Newkirk, J. A., Mathews, R. 1983. *The VLSI designers library: nMOS.* Reading, Mass: Addison-Wesley.

[Newma79] Newman, W. M., Sproull, R. F. 1979. *Principles of interactive computer graphics.* 2nd ed. McGraw-Hill.

[Nicke84] Nickel, Randy. The IRIS Workstation. *IEEE Computer Graphics & Applications.* August 1984, pp. 30–34.

[Oakle84] Oakley, D., Jones, M. E., Parsons, D., Burke, G. Pixel phasing smoothes out jagged lines. *Electronics.* 28 June 1984, pp. 118–120.

[Pavli82] Pavlidis, Theo. 1982. *Alogorithms for graphics and image processing.* Berlin-Heidelberg: Springer-Verlag.

[Peatm72] Peatman, John B. 1972. *The design of digital systems.* New York: Mc Graw-Hill.

[PlesC86] Plessey Semiconductors. *CLASP Reference Manual.* 1986.

[PlesD86] Plessey Semiconductors. *CLA5000 Design Manual.* 1986.

[PlesR86] Plessey Semiconductors. *CAD Reference Manual.* 1986.

[PlesS86] Plessey Semiconductors. *SCARP Reference Manual.* 1986.

[Rose__85] Rose, Craig. DEC's latest work station uses the VAX-on-a-chip. *Electronics Week.* 27 May 1985, pp. 68–69.

[Rose__86] Rose, Craig D. It's high noon for work-station makers. *Electronics.* 17 March 1986, pp. 54–55.

[Roth__79] Roth, Charles H., Jr. 1979. *Fundamentals of logic design.* 2nd edition. Minnesota: West Publishing Co.

[Schac83] Schachter, Bruce J., ed. 1983. *Computer image generation.* New York: Wiley.

[Staud86] Staudhammer, John. Reaping the benefits of the hardware revolution. *IEEE Computer Graphics & Applications.* January 1984, pp. 14–15.

[Whitt84] Whitton, Mary C. Memory design for raster graphics displays. *IEEE Computer Graphics & Applications.* March 1984, pp. 48–64.

[Willi83] Williams, T. W., Parker, K. P. Design for Testibility-A Survey. *Proceedings of the IEEE.* Vol.71, No.1. January 1983, pp. 98–112

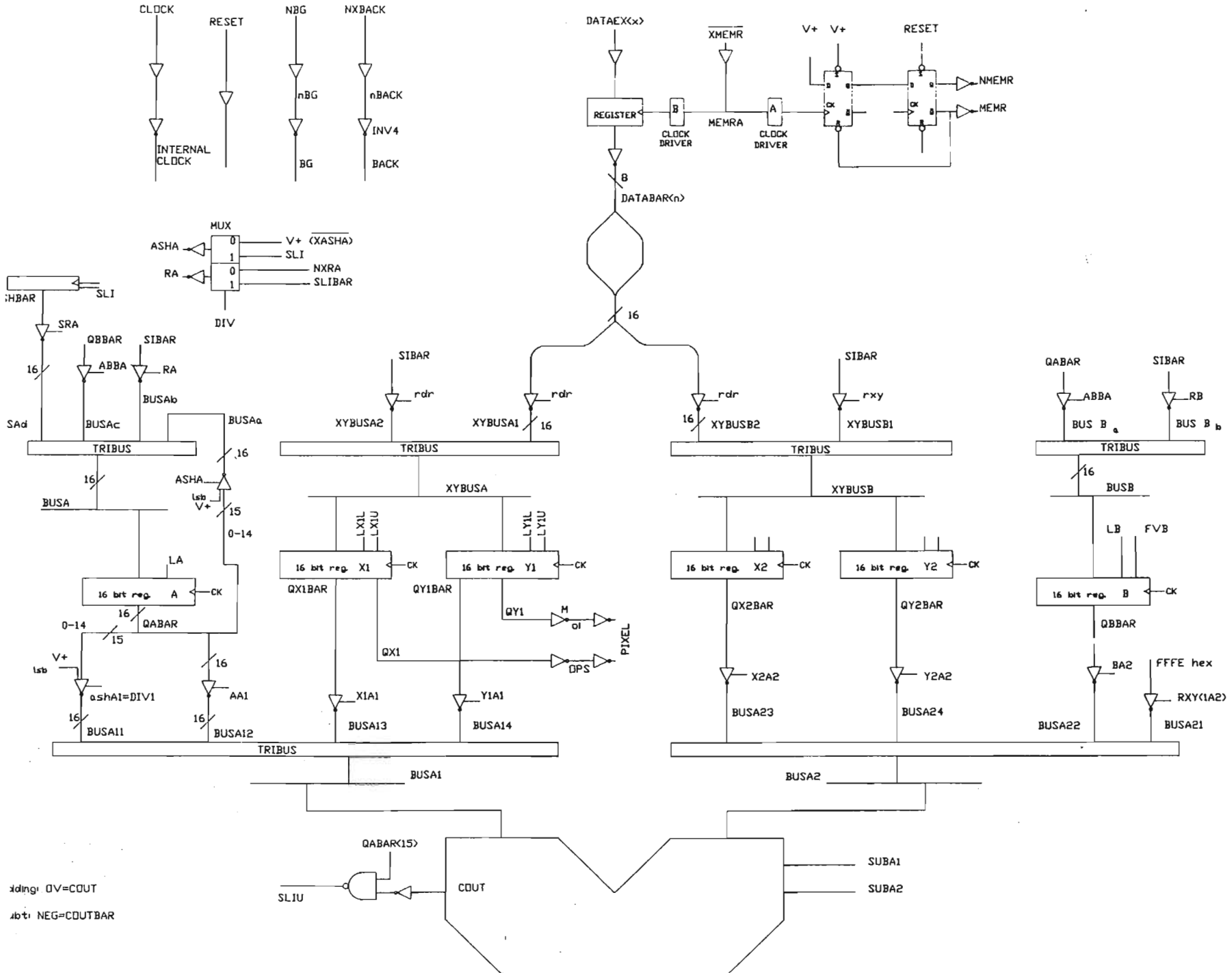
[Winke80] Winkel, D., Prosser, F. 1980. *The art of digital design.* New Jersey: Prentice-Hall.

[Zollo84] Zollo, Steve. System displays 1,664 by 1,248 pixels. *Electronics.* 3 May 1984, pp. 183–184.

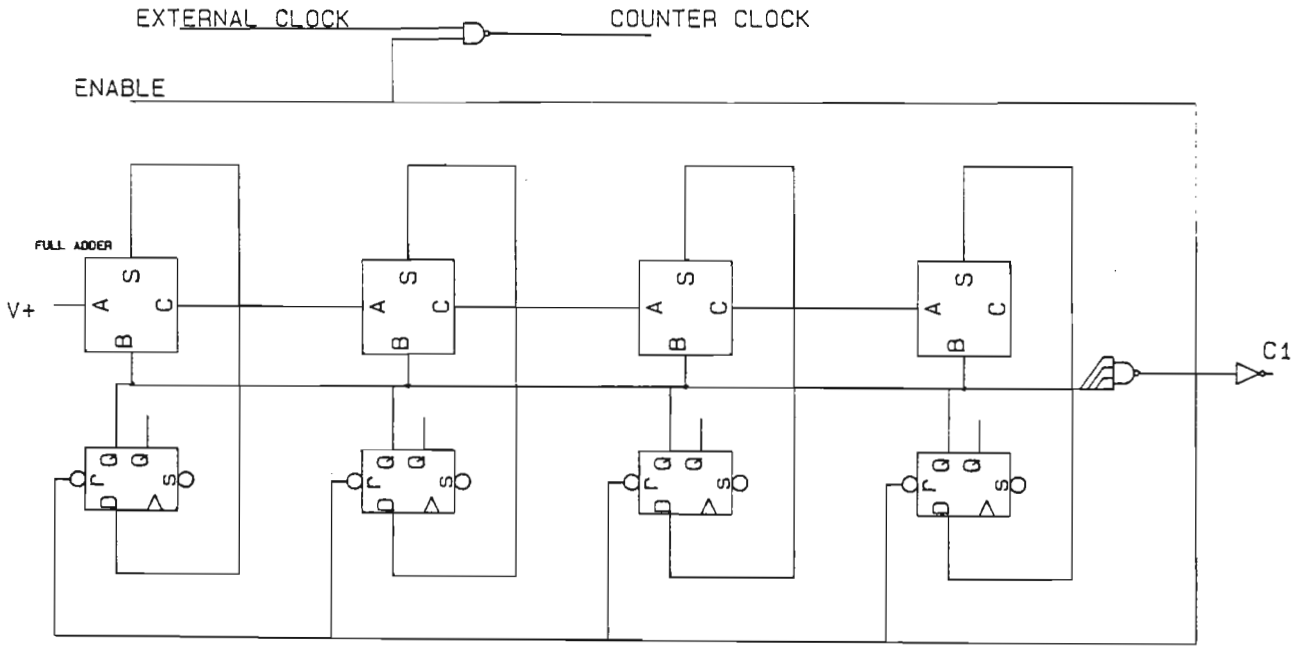
[Zollo85] Zollo, Steve. Cards do graphics fast. *Electronics.* 1 April 1986, pp. 60–61.

Appendix 1.
Circuit diagrams.

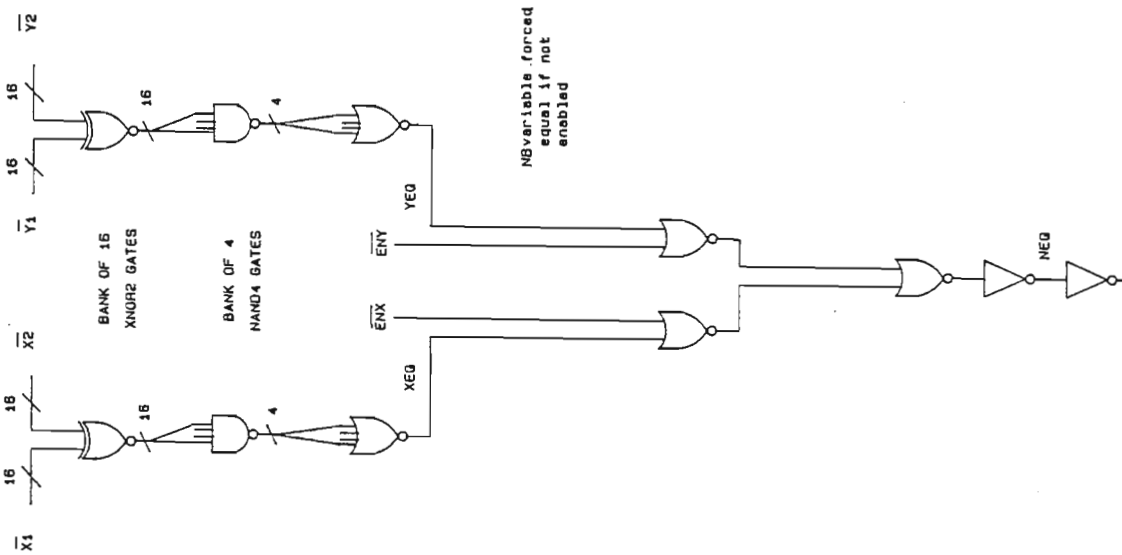
This appendix consists of a diagram of the true internal circuit (not including the state machine) with all the net names used in the .ccl files of appendix 5.


CLOCK
RESET
NBG
NXBACK
INTERNAL CLOCK
nBG
nBACK
INV4
BG
BACK
DATAEX<x>
XMEMR
V+ V+
RESET
REGISTER
B
CLOCK DRIVER
MEMRA
A
CLOCK DRIVER
NMEMR
MEMR
B
DATABAR<n>
MUX
ASHA
RA
V+ (XASHA)
SLI
NXRA
SLIBAR
DIV
16
SLI
SRA
QBBAR
SIBAR
ABBA
RA
BUSAb
BUSAc
BUSAa
TRIBUS
16
ASHA
lsb
V+
15
0-14
BUSA
LA
16 bit reg. A
CK
16
QABAR
0-14
15
V+
lsb
ashA1=DIV1
AA1
16
BUSA11
16
BUSA12
TRIBUS
BUSA1
SIBAR
rdr
XYBUSA2
rdr
XYBUSA1
16
TRIBUS
XYBUSA
LX1L
LX1U
16 bit reg. X1
CK
QX1BAR
LY1L
LY1U
16 bit reg. Y1
CK
QY1BAR
QY1
M
ol
QX1
DPS
PIXEL
X1A1
BUSA13
Y1A1
BUSA14
16
rdr
XYBUSB2
SIBAR
rxy
XYBUSB1
TRIBUS
XYBUSB
16 bit reg. X2
CK
QX2BAR
16 bit reg. Y2
CK
QY2BAR
X2A2
BUSA23
Y2A2
BUSA24
TRIBUS
BUSA2
QABAR
ABBA
BUS B a
SIBAR
RB
BUS B b
TRIBUS
16
BUSB
LB
FVB
16 bit reg. B
CK
QBBAR
BA2
BUSA22
FFFE hex
RXY(1A2)
BUSA21
QABAR<15>
SLIU
COUT
SUBA1
SUBA2
dding OV=COUT
bt NEG=COUTBAR

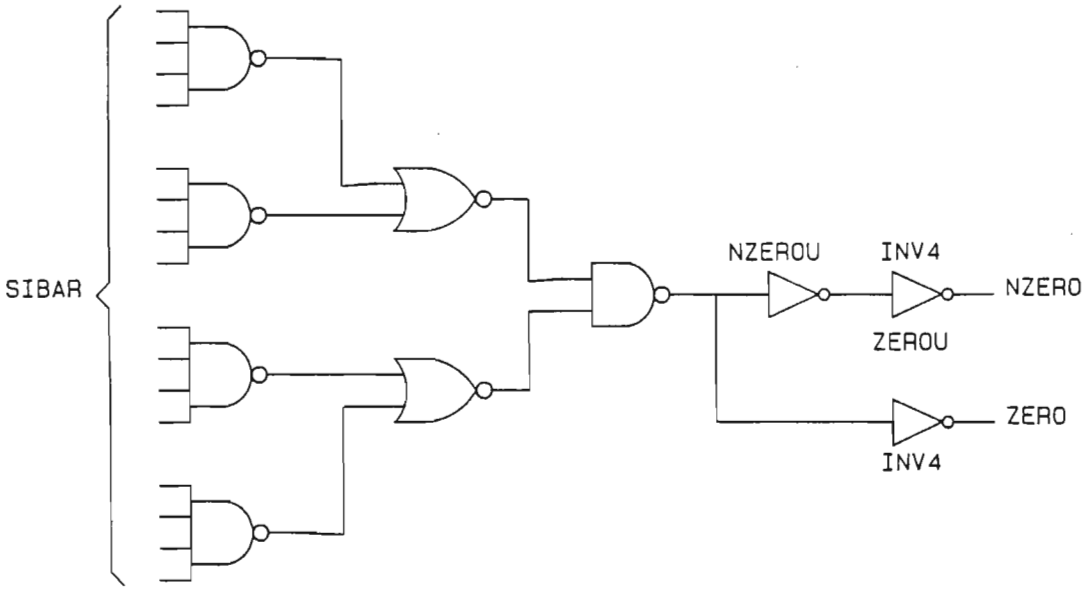
MODULO SIXTEEN COUNTER C16



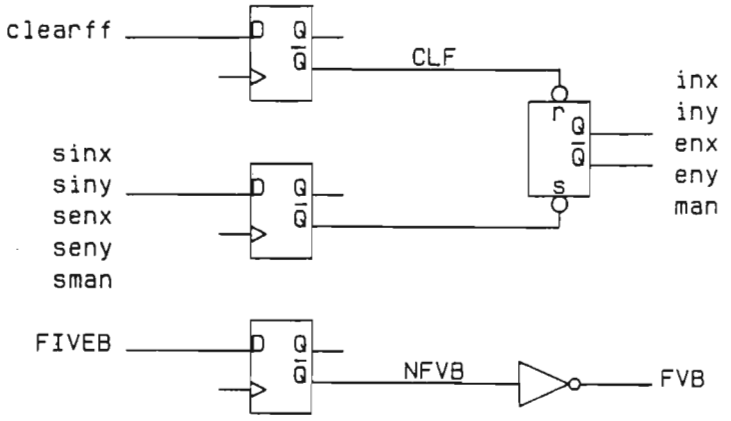
COMPARE CIRCUIT



ZERO DETECTOR



SYNCHRONIZING CIRCUITS AND EXTRA BITS



Appendix 2.
DMA timing considerations.

There are some clock speed constraints on the rasterizer in order to ensure that it works with the PC DMA controller properly.

The 8237 DMA controller is used in demand transfer mode. Note that the clock cycle for the PC (PCT) is 210ns, 70ns high and 140ns low. In demand transfer mode the DMA DREQ must be removed TQS before the 8237 state four, S4, of the last byte transfer cycle (see timing diagram).

The DMA arbitration system needs to know 1) that the rasterizer is about to receive, or is receiving, its last byte 2) when the last byte transfer cycle has begun, in order to remove the DREQ. Condition 2) is indicated by the falling edge of the 8237 MEMR and condition 1) may be learned from two rasterizer signals: S9 and S10. S9 and S10 indicate that the rasterizer is ready for the second-last and last bytes respectively.

S10 may be used with MEMR to stop transfer, if it arrives with– or before MEMR. S10 may be up to two rasterizer cycles (2T) after the 8237 IOW (rasterizer memr). Referring to the timing diagram, this means that

$$PCT - TDCTW + TDCL \geq 2T \leq 210 - 130 + 190 = 270$$

that is, the rasterizer must be clocked at $\geq$ 7.4 MHz.

If a slower clock is to be used, S9 must be used. This does, however, mean that S9 must be remembered external to the rasterizer. S9 must be latched with the rising edge of IOW (with the incoming data). This latched S9 may be used with the falling edge of MEMR as before to remove DREQ.

In this mode the rasterizer needs to be clocked at two thirds of the speed of the 8237 in order to keep up with the data rate. This is because the rasterizer will spend at least two clock cycles in the each read-state, while a DMA transfer is three clock cycles.

DMA TRANSFER TIMING

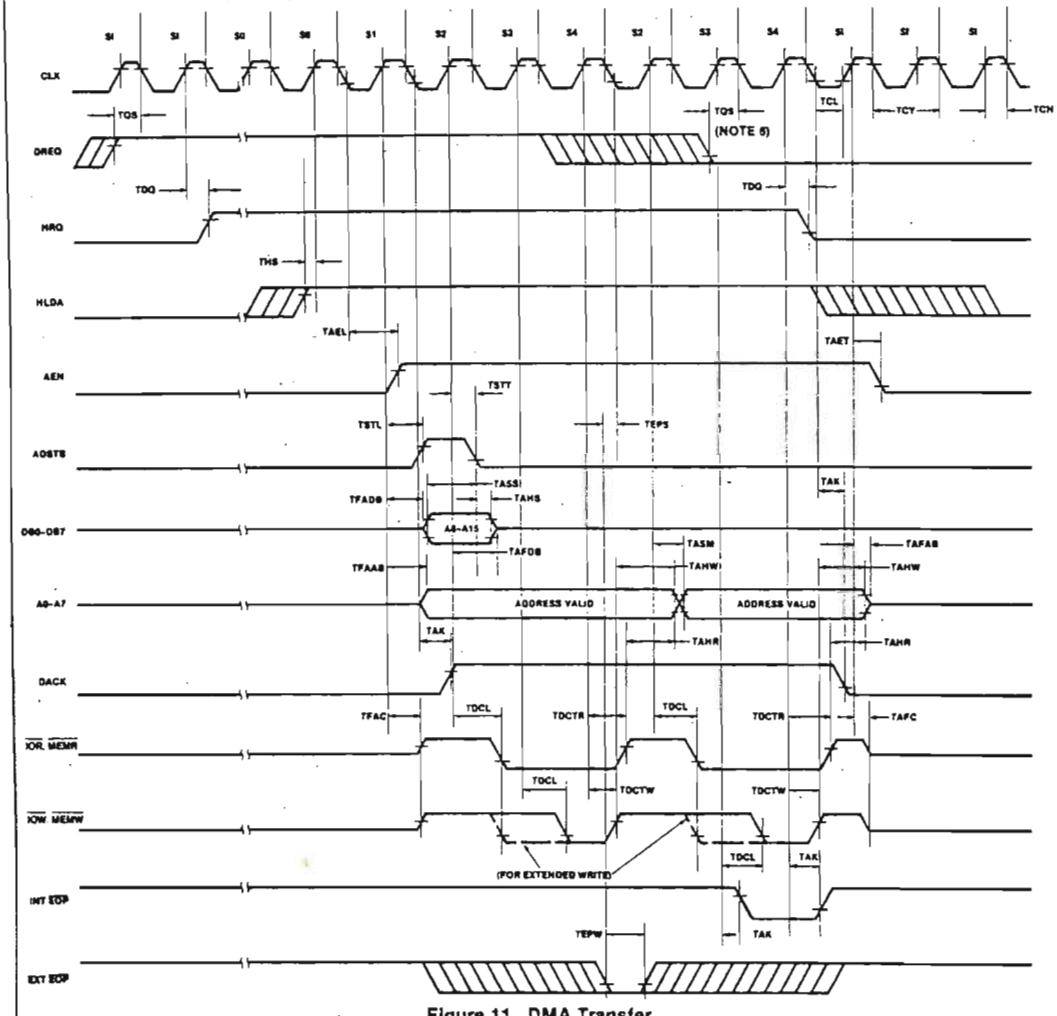


Figure 11. DMA Transfer

Appendix 3.
State machine.

This appendix contains the PEG form of the state diagram and the circuit diagram of the state machine along with complete next state and output forming logic.

The next state and output forming logic shown here is the result of manipulation of the equations produced by PEG.

This is an example of an equation (nsl bit 0) produced by PEG:

```
OutSt0*=
(!RESET& MAN&!ENX& ENY& BG&!EQ& InSt0*&!InSt1*&!InSt2*&!InSt3*& InSt4*&!InSt5*)
(!RESET& ENY&!BG& InSt0*&!InSt1*&!InSt2*&!InSt3*& InSt4*&!InSt5*)
(!RESET& MAN& ENX&!ENY&!BG&!EQ& InSt0*&!InSt1*&!InSt2*&!InSt3*& InSt4*&!InSt5*)
(!RESET& MAN&!ENY&!BG& EQ& InSt0*&!InSt1*&!InSt2*&!InSt3*& InSt4*&!InSt5*)
(!RESET& MAN&!ENX&!ENY&!EQ& InSt0*&!InSt1*&!InSt2*&!InSt3*& InSt4*&!InSt5*)
(!RESET&!MAN&!ENY&!BG& InSt0*&!InSt1*&!InSt2*&!InSt3*& InSt4*&!InSt5*)
(!RESET& InSt0*&!InSt1*&!InSt2*&!InSt3*&!InSt4*)
(!RESET&!InSt0*& InSt1*& InSt2*& InSt3*)
(!RESET&!ENX&!OV&!InSt0*& InSt1*& InSt2*&!InSt3*&!InSt4*& InSt5*)
(!RESET&!ENX&!InSt0*& InSt1*&!InSt2*& InSt3*&!InSt4*&!InSt5*)
(!RESET& MAN&!ENX&!InSt0*& InSt1*&!InSt2*&!InSt3*& InSt4*);
```

The equations may be converted into a truth table by EQNTOT and then minimized by PRESTO, however, the manipulation in this case was done manually with the aid of a logic minimization program (a modified version of a Departmental program, MINIM) since the Berkley software was not running on a local machine at the time. A number of logical changes were affected by hand at a late stage in the design. The logic shown in the diagrams that follow is a result of hand translation of logic equations.

PEG representation of the rasterizer's state diagram

The state diagram is transformed into the PEG language formatshown below as input to the program PEG which generates PLAequations.

The structures of the language are largely self evident; each state is represented by a colon and may or may not be labled; signals are defined for each state by the ASSERT construct; flow control is achieved in a similar manner to Pascal or any other structured language. The PEG program is isomorphic to the state diagram.

```
INPUTS  :BACK MEMR
 ZERO NEG MAN C16 ENX ENY INX INY
 BG EQ OV;

OUTPUTS :BREQU LX1L
 LX1U   LY1L LY1U    LX2L LX2U
 LY2L   LY2U CLEARFF   X1A1 SENX
 X2A2   RA  SINX SENY   BA2
 LA SUBA2 SUBA1     LB  SMAN
 SINY   Y1A1 Y2A2   RB  AA1
 ABBA  ASHA1
 DIV    SRA RDR FVB
 IA2    RXY BR

import    :ASSERT BREQU;
     IF NOT BACK THEN LOOP;

          :ASSERT RDR;
     IF NOT MEMR THEN LOOP;
          :ASSERT RDR;
     IF NOT MEMR THEN LOOP;

          :ASSERT RDR LX1L;
     IF NOT MEMR THEN LOOP;
```

PEG representation of the rasterizer's state diagram.

```
        :ASSERT RDR LX1U;
    IF NOT MEMR THEN LOOP;

        :ASSERT RDR LY1L;
    IF NOT MEMR THEN LOOP;
        :ASSERT RDR LY1U;
    IF NOT MEMR THEN LOOP;

        :ASSERT RDR LX2L;
    IF NOT MEMR THEN LOOP;
        :ASSERT RDR LX2U;
    IF NOT MEMR THEN LOOP;

        :ASSERT RDR LY2L;
    IF NOT MEMR THEN LOOP;
        :ASSERT RDR LY2U;
    IF NOT MEMR THEN LOOP;


analyse :ASSERT CLEARFFS X1A1 X2A2 RA LA SUBA2;

    CASE (ZERO NEG)
            0    0    => deltaY;
            0    1    => swopX;
            1    0    => zeroX;
            1    1    => import;
    ENDCASE;

swopX    :ASSERT X1A1 X2A2 RA LA SUBA1 SINX; GOTO
deltaY;

zeroX    :ASSERT SMAN SENY; GOTO deltaY;

deltaY   :ASSERT Y1A1 Y2A2 RB LB SUBA2;
```

PEG representation of the rasterizer's state diagram.

```
    CASE (ZERO NEG)
            0     0 => dltaXY;
            0     1 => swopY;
            1     0 => zeroY;
            1     1 => import;
    ENDCASE;

swopY    :ASSERT Y1A1 Y2A2 RB LB SUBA1 SINY; GOTO
dltaXY;

zeroY    :ASSERT SMAN SENX; GOTO dltaXY;

dltaXY   :ASSERT AA1 BA2 SUBA1;

    CASE (ZERO NEG)
            0     0 => everY;
            0     1 => everX;
            1     0 => fortyf;
            1     1 => import;
    ENDCASE;

fortyf   : ASSERT SENX SENY SMAN; GOTO busrq;

everY    :ASSERT SENY; IF MAN busrq ELSE divide;

everX    :ASSERT ABBA LA LB SENX; IF MAN busrq ELSE
divide;



divide   :


         :ASSERT ASHA1 BA2 SUBA2 DIV LA;
    IF NOT C16 THEN LOOP;

         :ASSERT SRA LA;
```

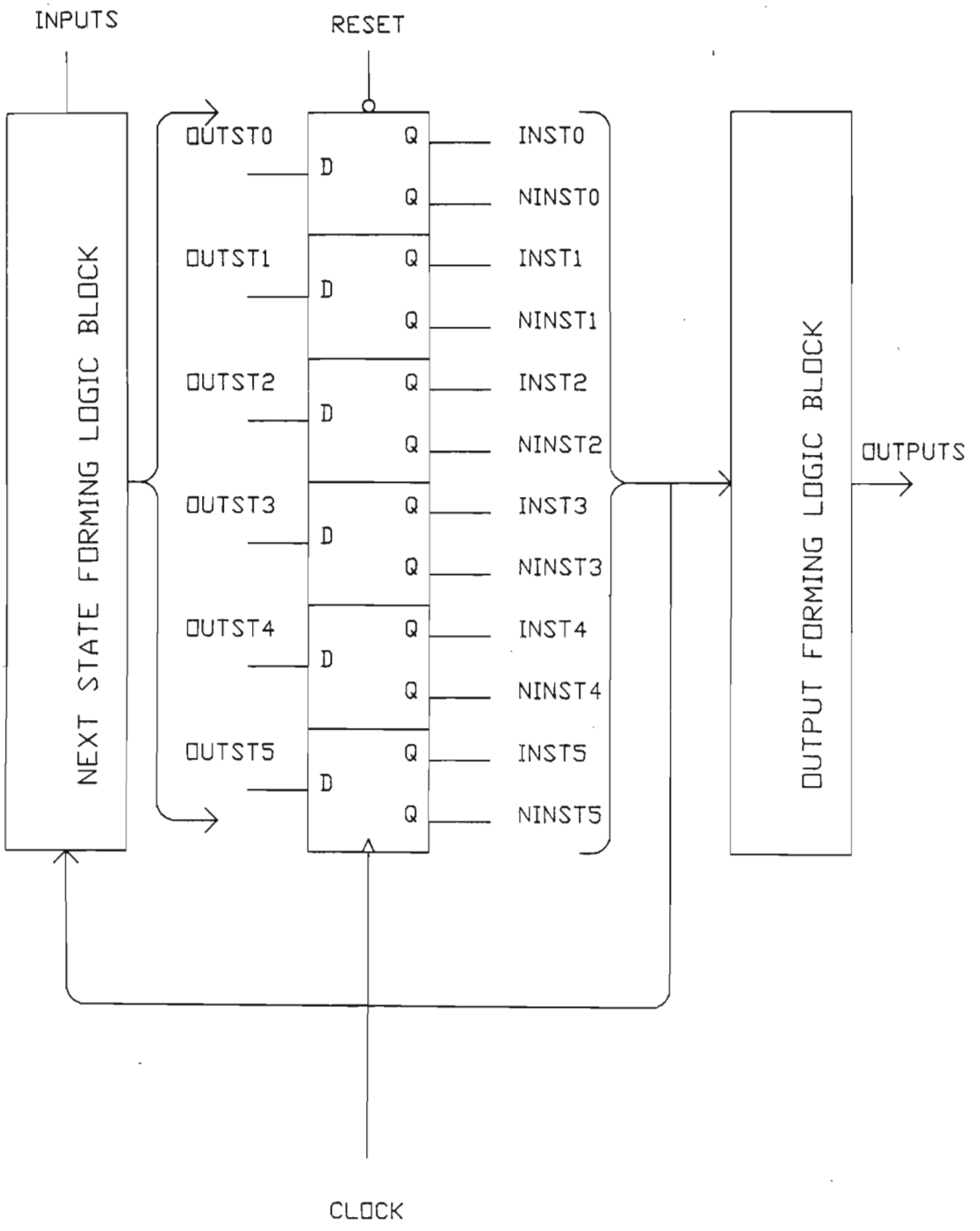
PEG representation of the rasterizer's state diagram.

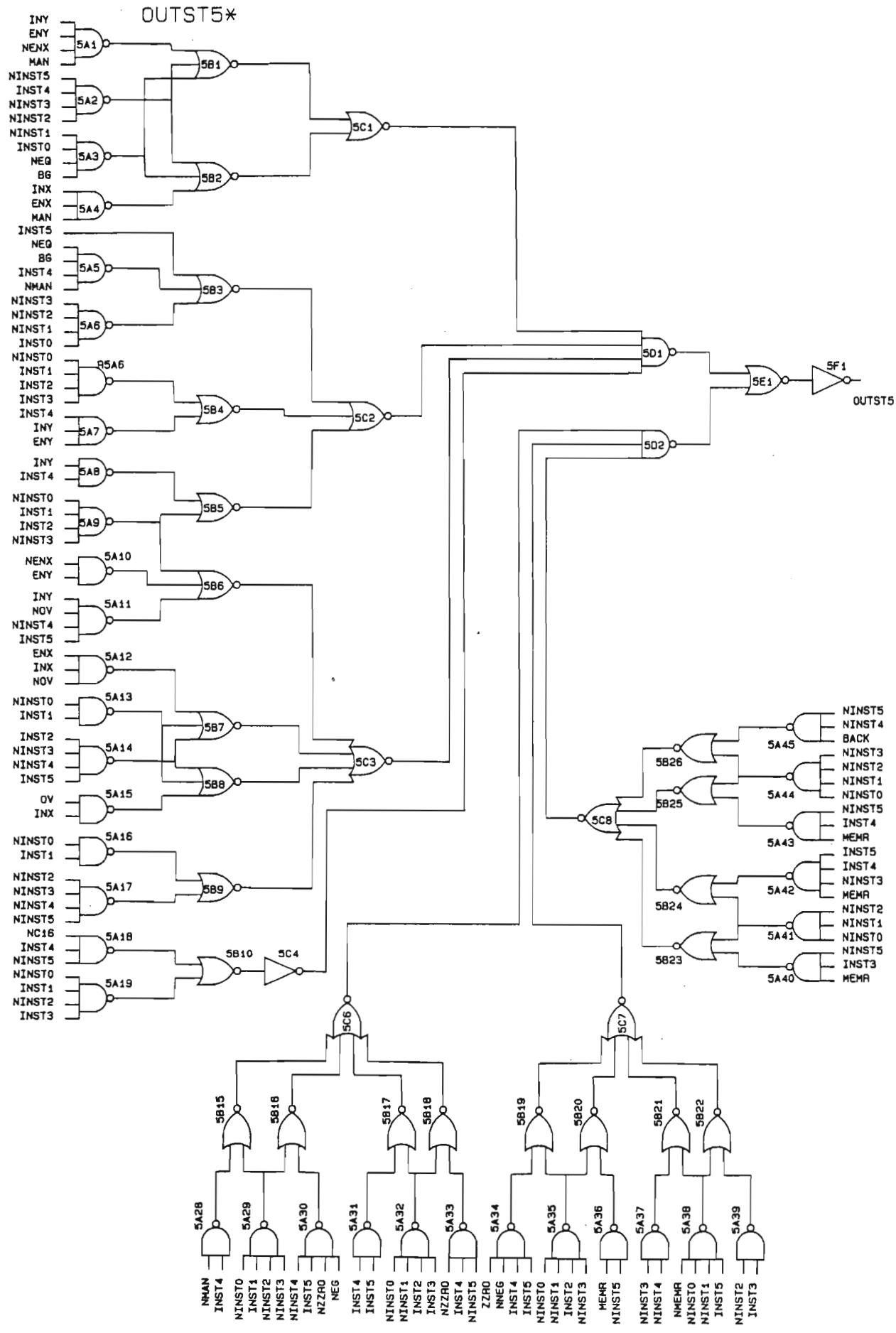
```
            :ASSERT FVB; GOTO busrq;


chkfrc      :ASSERT AA1 BA2 LB RB;
 CASE (OV ENX INX ENY INY)
            0    1    0    ?    ?    => dX;
            0    1    1    ?    ?    => iX;
            0    0    ?    1    0    => dY;
            0    0    ?    1    1    => iY;
            0    0    ?    0    ?    => busrq;
            1    ?    0    ?    ?    => decX;
            1    ?    1    ?    ?    => incX;
 ENDCASE;

decX        :ASSERT SUBA2 X1A1 IA2 RXY LX1U LX1L;
     IF INY THEN incY ELSE decY;

incX        :ASSERT X1A1 IA2 RXY LX1U LX1L;
     IF INY THEN incY ELSE decY;

decY        :ASSERT SUBA2 Y1A1 IA2 RXY LY1U LY1L; GOTO
busrq;

incY        :ASSERT Y1A1 IA2 RXY LY1U LY1L; GOTO busrq;

dX          :ASSERT SUBA2 X1A1 IA2 RXY LX1U LX1L;
     CASE (ENY INY)
            0  ? => busrq;
            1  0 => dY;
            1  1 => iY;
     ENDCASE;

iX          :ASSERT X1A1 IA2 RXY LX1U LX1L;
  CASE (ENY INY)
     0      ? => busrq;
     1      0 => dY;
     1      1 => iY;
  ENDCASE;
```

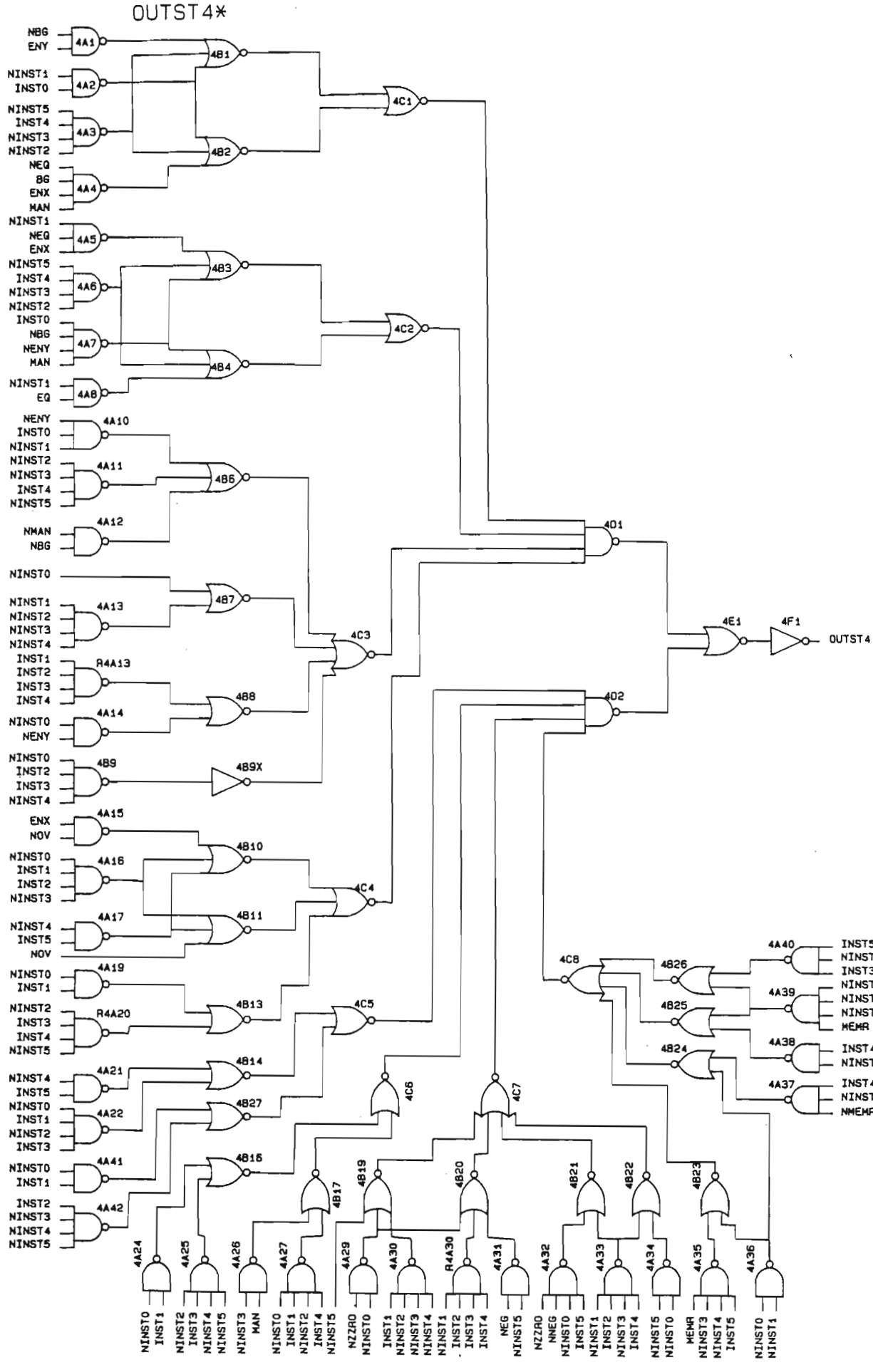
PEG representation of the rasterizer's state diagram.

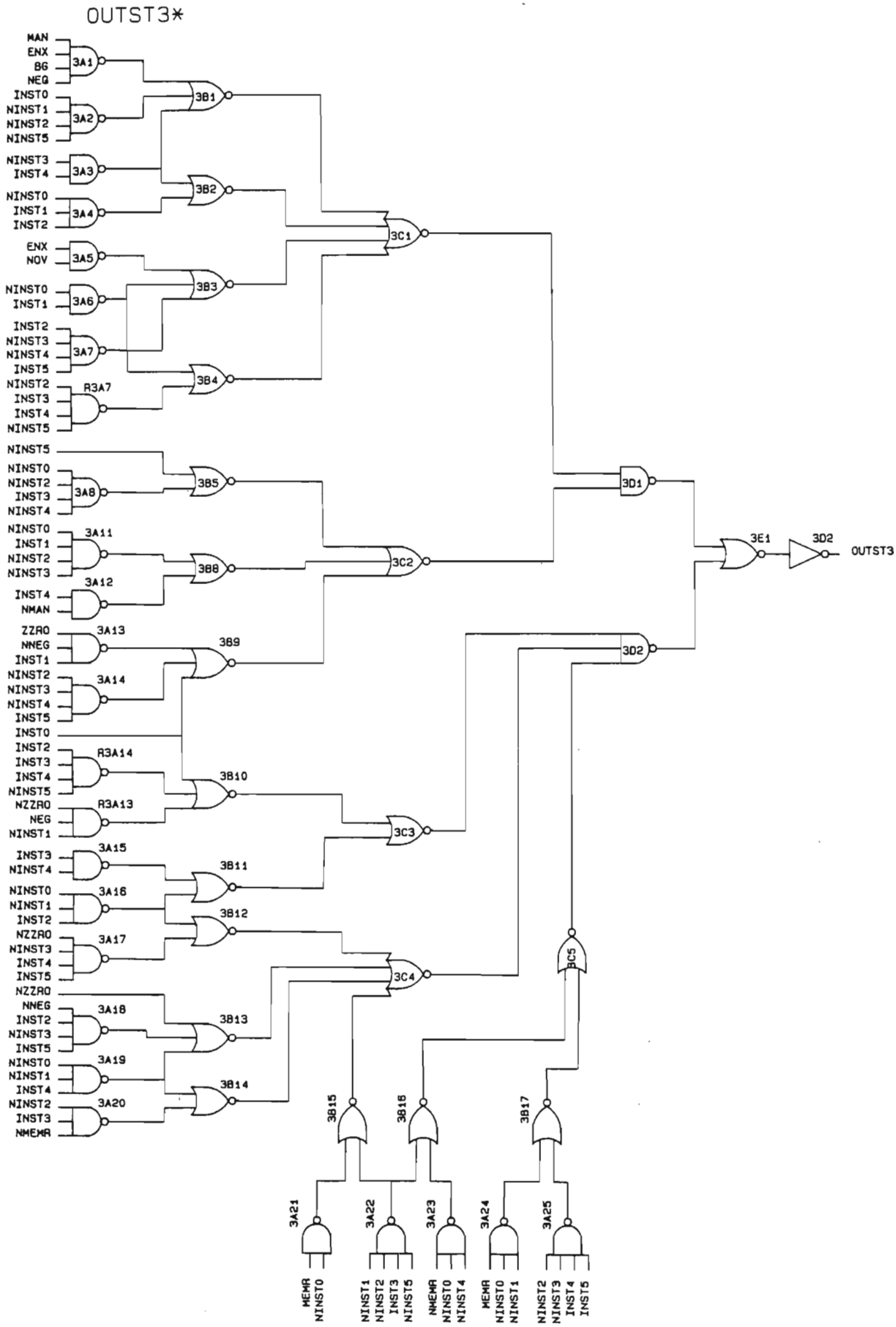
```
dY          :ASSERT SUBA2 Y1A1 IA2 RXY LY1U LY1L; GOTO
busrq;

iY          :ASSERT Y1A1 IA2 RXY LY1U LY1L; GOTO busrq;

busrq       :ASSERT BR;
     CASE (BG EQ MAN ENX INX ENY INY)
          0  ?  ?  ?  ?  ?  ?  =>LOOP;
          1  1  ?  ?  ?  ?  ?  =>import;
          1  0  0  ?  ?  ?  ?  =>chkfrc;
          1  0  1  1  0  ?  ?  => dX;
          1  0  1  1  1  ?  ?  => iX;
          1  0  1  0  ?  1  0  => dY;
          1  0  1  0  ?  1  1  => iY;
          1  0  1  0  ?  0  ?  => busrq;
          ENDCASE;

          : GOTO import;

The end.
```

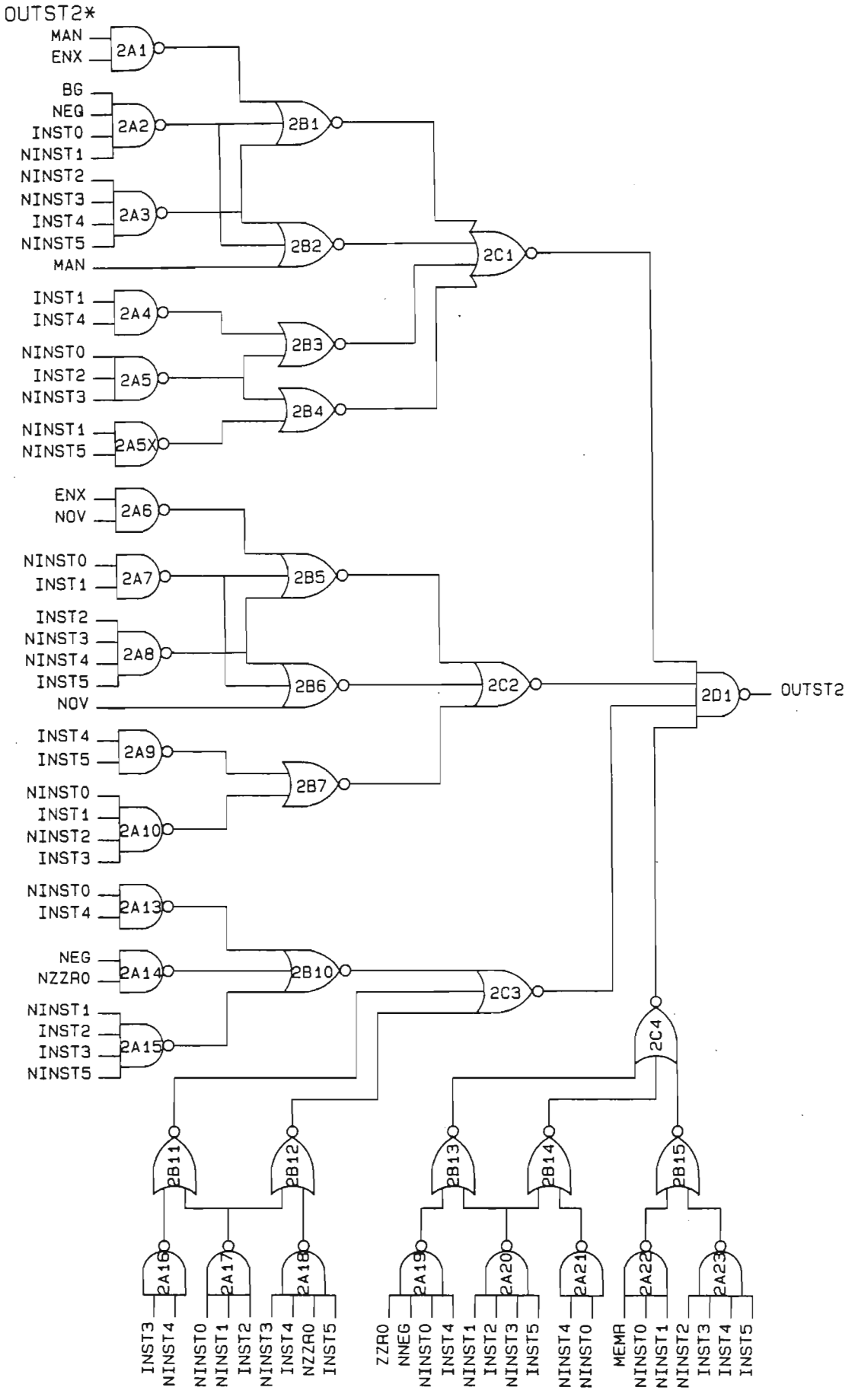
STATE MACHINE STRUCTURE

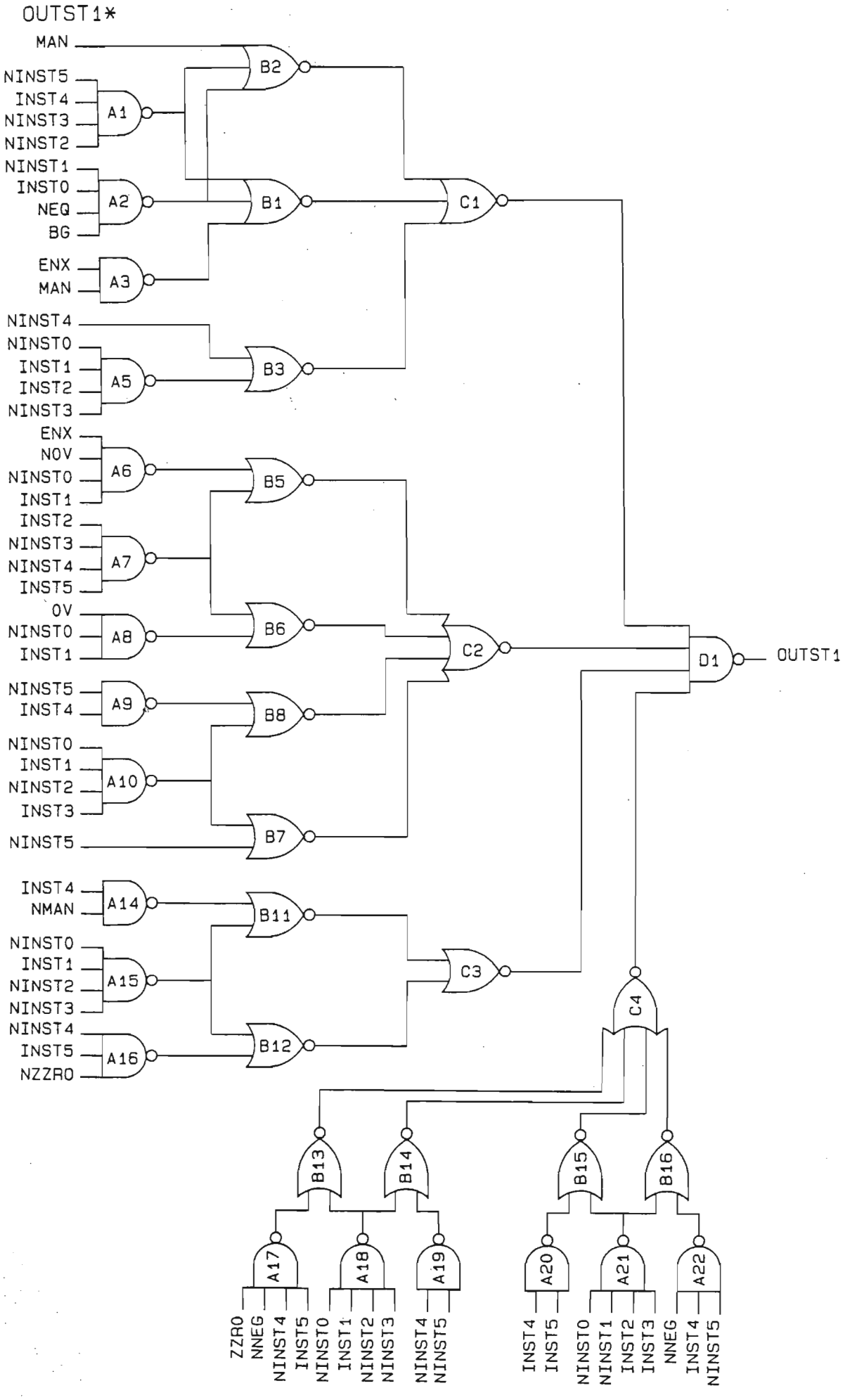


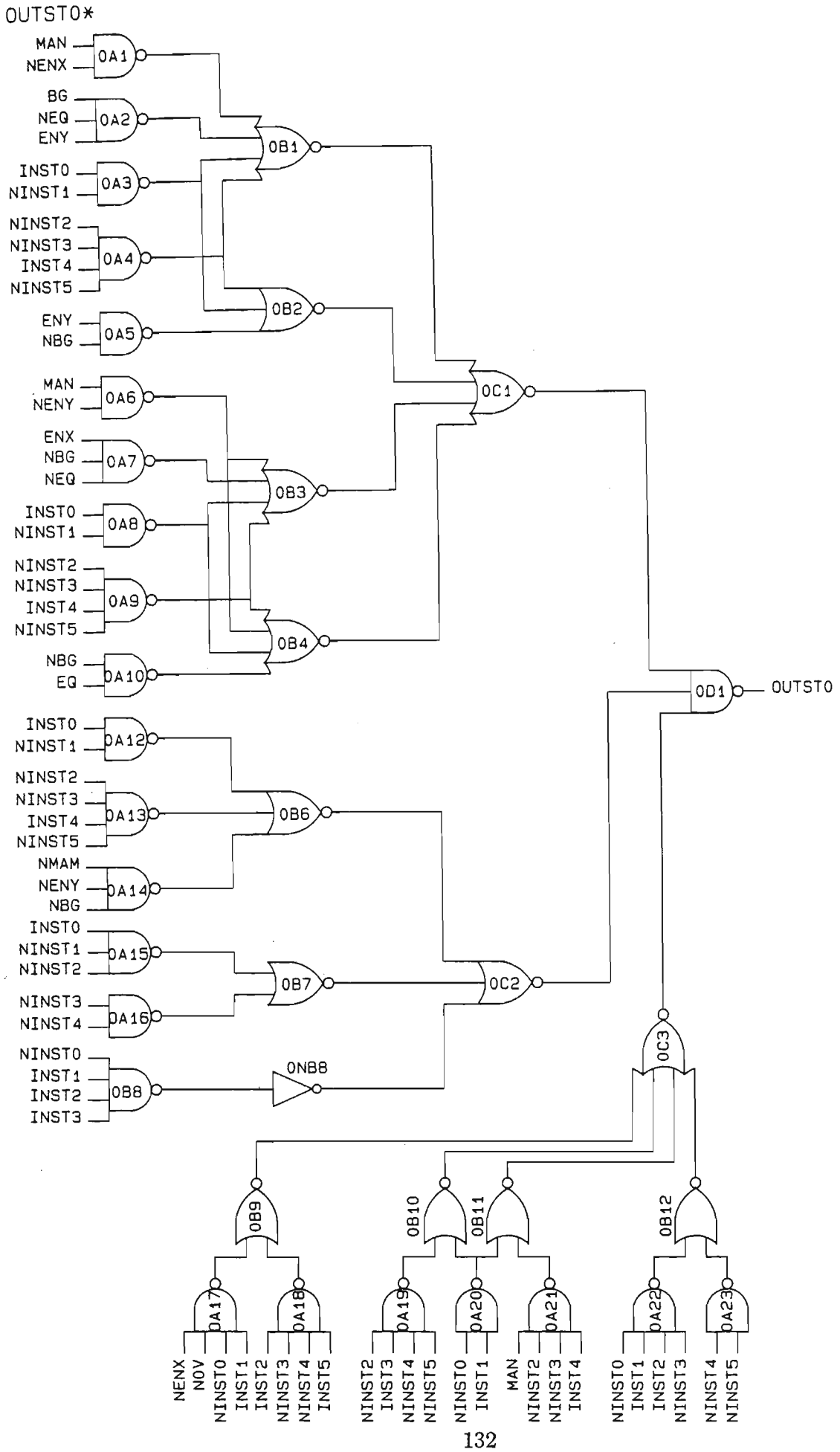

OUTST5*
INY
ENY
NENX
MAN
5A1
5B1
NINST5
INST4
NINST3
NINST2
5A2
NINST1
INST0
NEQ
BG
5A3
5B2
INX
ENX
MAN
5A4
5C1
INST5
NEQ
BG
INST4
NMAN
5A5
5B3
NINST3
NINST2
NINST1
INST0
5A6
NINST0
INST1
INST2
INST3
INST4
INY
ENY
5A7
5B4
5C2
INY
INST4
5A8
5B5
NINST0
INST1
INST2
NINST3
5A9
NENX
ENY
5A10
5B6
INY
NOV
NINST4
INST5
5A11
ENX
INX
NOV
5A12
NINST0
INST1
5A13
5B7
INST2
NINST3
NINST4
INST5
5A14
5C3
OV
INX
5A15
5B8
NINST0
INST1
5A16
NINST2
NINST3
NINST4
NINST5
5A17
5B9
NC16
INST4
NINST5
NINST0
INST1
NINST2
INST3
5A18
5A19
5B10
5C4
5D1
5E1
5F1
OUTST5
5D2
5C8
5A45
NINST5
NINST4
BACK
5B26
5A44
NINST3
NINST2
NINST1
NINST0
5B25
5A43
NINST5
INST4
MEMR
5B24
5A42
INST5
INST4
NINST3
MEMR
5A41
NINST2
NINST1
NINST0
5B23
5A40
NINST5
INST3
MEMR
5C6
5C7
5B15
5B16
5B17
5B18
5B19
5B20
5B21
5B22
5A28
5A29
5A30
5A31
5A32
5A33
5A34
5A35
5A36
5A37
5A38
5A39
NMAN
INST4
NINST0
INST1
NINST2
NINST3
NINST4
INST5
NZZRO
NEG
INST4
INST5
NINST0
NINST1
INST2
INST3
NZZRO
INST4
NINST5
ZZRO
NNEG
INST4
INST5
NINST0
NINST1
INST2
NINST3
MEMR
NINST5
NINST3
NINST4
NMEMR
NINST0
NINST1
INST5
NINST2
INST3

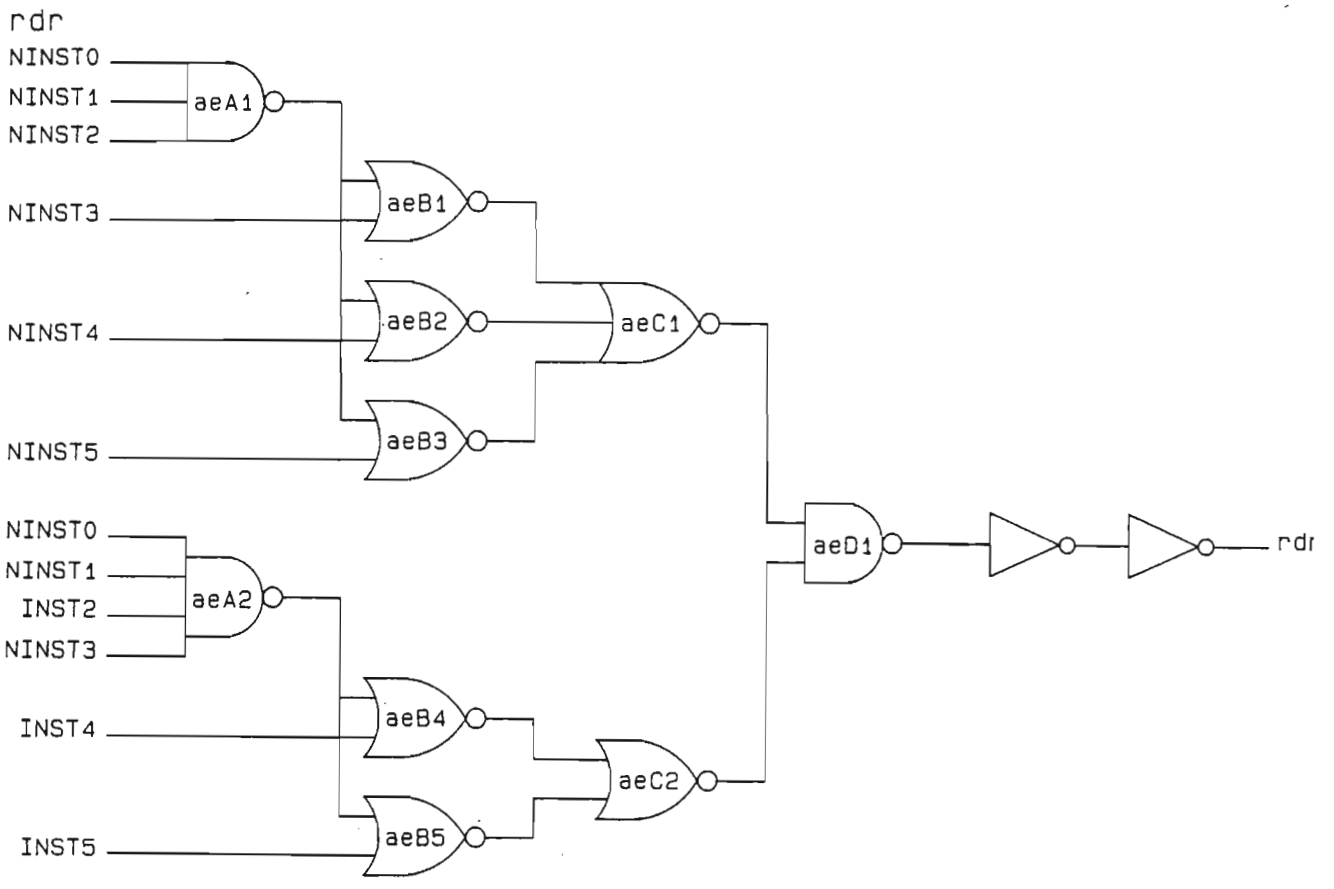

OUTST4*
NBG
ENY
4A1
4B1
NINST1
INST0
4A2
NINST5
INST4
NINST3
NINST2
4A3
4B2
4C1
NEQ
BG
ENX
MAN
4A4
NINST1
NEQ
ENX
4A5
4B3
NINST5
INST4
NINST3
NINST2
4A6
4C2
INST0
NBG
NENY
MAN
4A7
4B4
NINST1
EQ
4A8
NENY
INST0
NINST1
NINST2
NINST3
INST4
NINST5
4A10
4A11
4B6
NMAN
NBG
4A12
4D1
NINST0
NINST1
NINST2
NINST3
NINST4
4A13
4B7
4C3
4E1
4F1
OUTST4
INST1
INST2
INST3
INST4
R4A13
4B8
NINST0
NENY
4A14
4D2
NINST0
INST2
INST3
NINST4
4B9
4B9X
ENX
NOV
4A15
4B10
NINST0
INST1
INST2
NINST3
4A16
4C4
NINST4
INST5
NOV
4A17
4B11
NINST0
INST1
4A19
NINST2
INST3
INST4
NINST5
R4A20
4B13
4C5
4C8
4A40
INST5
NINST4
INST3
4B26
NINST2
NINST1
NINST0
MEMR
4A39
4B25
4A38
INST4
NINST5
4B24
4A37
INST4
NINST2
NMEMR
NINST4
INST5
4A21
4B14
4C6
4C7
NINST0
INST1
NINST2
INST3
4A22
4B27
NINST0
INST1
4A41
4B15
INST2
NINST3
NINST4
NINST5
4A42
4B17
4B19
4B20
4B21
4B22
4B23
4A24
4A25
4A26
4A27
4A29
4A30
R4A30
4A31
4A32
4A33
4A34
4A35
4A36
NINST0
INST1
NINST2
INST3
INST4
NINST5
NINST3
MAN
NINST0
INST1
NINST2
INST4
NINST5
NZERO
NINST0
INST1
NINST2
INST3
INST4
NINST1
INST2
INST3
INST4
NEG
NINST5
NZERO
NNEG
NINST0
INST5
NINST1
INST2
NINST3
INST4
NINST5
NINST0
MEMR
NINST3
NINST4
INST5
NINST0
NINST1

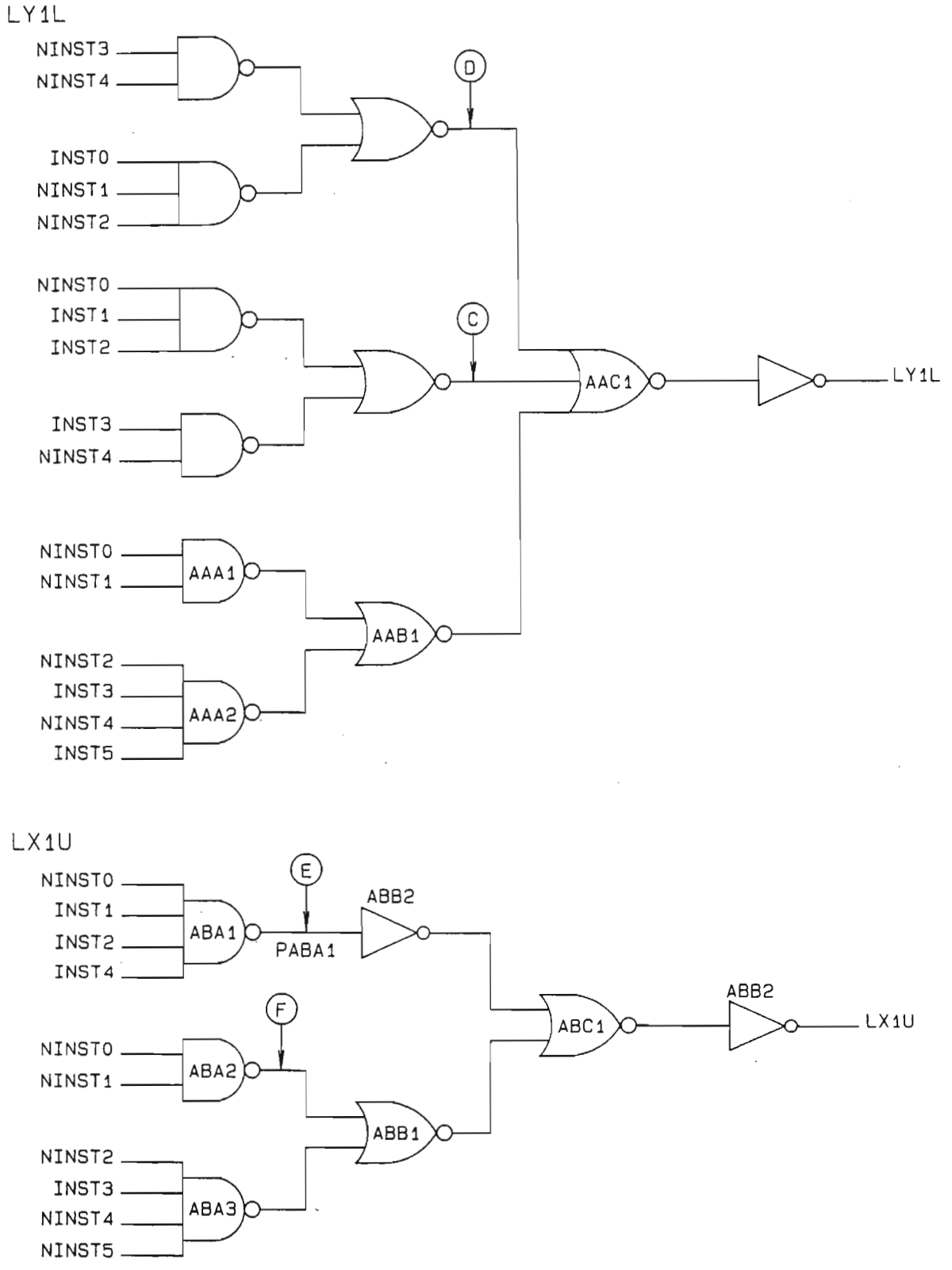

OUTST3*
MAN
ENX
BG
NEG
3A1
INST0
NINST1
NINST2
NINST5
3A2
3B1
NINST3
INST4
3A3
3B2
NINST0
INST1
INST2
3A4
3C1
ENX
NOV
3A5
3B3
NINST0
INST1
3A6
INST2
NINST3
NINST4
INST5
3A7
3B4
NINST2
INST3
INST4
NINST5
R3A7
NINST5
NINST0
NINST2
INST3
NINST4
3A8
3B5
3D1
NINST0
INST1
NINST2
NINST3
3A11
3B8
3C2
3E1
3D2
OUTST3
INST4
NMAN
3A12
ZZR0
NNEG
INST1
3A13
3B9
3D2
NINST2
NINST3
NINST4
INST5
3A14
INST0
INST2
INST3
INST4
NINST5
R3A14
3B10
NZZR0
NEG
NINST1
R3A13
3C3
INST3
NINST4
3A15
3B11
NINST0
NINST1
INST2
3A16
3B12
NZZR0
NINST3
INST4
INST5
3A17
3C4
3C5
NZZR0
NNEG
INST2
NINST3
INST5
3A18
3B13
NINST0
NINST1
INST4
3A19
3B14
NINST2
INST3
NMEMR
3A20
3B15
3B16
3B17
3A21
3A22
3A23
3A24
3A25
MEMR
NINST0
NINST1
NINST2
INST3
NINST5
NMEMR
NINST0
NINST4
MEMR
NINST0
NINST1
NINST2
NINST3
INST4
INST5


OUTST2*
MAN
ENX
2A1
BG
NEG
INST0
NINST1
2A2
NINST2
NINST3
INST4
NINST5
2A3
MAN
2B1
2B2
2C1
INST1
INST4
2A4
2B3
NINST0
INST2
NINST3
2A5
2B4
NINST1
NINST5
2A5X
ENX
NOV
2A6
NINST0
INST1
2A7
2B5
INST2
NINST3
NINST4
INST5
NOV
2A8
2B6
2C2
2D1
OUTST2
INST4
INST5
2A9
2B7
NINST0
INST1
NINST2
INST3
2A10
NINST0
INST4
2A13
NEG
NZZR0
2A14
2B10
2C3
NINST1
INST2
INST3
NINST5
2A15
2C4
2B11
2B12
2B13
2B14
2B15
2A16
2A17
2A18
2A19
2A20
2A21
2A22
2A23
INST3
NINST4
NINST0
NINST1
INST2
NINST3
INST4
NZZR0
INST5
ZZR0
NNEG
NINST0
INST4
NINST1
INST2
NINST3
INST5
NINST4
NINST0
MEMR
NINST0
NINST1
NINST2
INST3
INST4
INST5

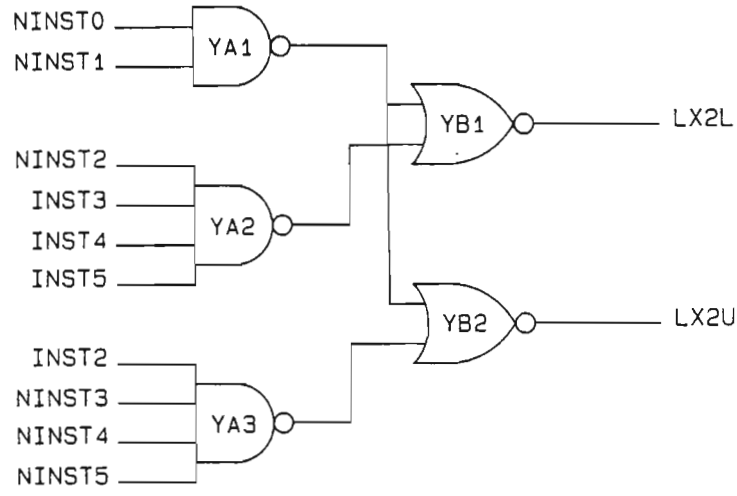

OUTST1*
MAN
NINST5
INST4
NINST3
NINST2
NINST1
INST0
NEQ
BG
ENX
MAN
NINST4
NINST0
INST1
INST2
NINST3
ENX
NOV
NINST0
INST1
INST2
NINST3
NINST4
INST5
OV
NINST0
INST1
NINST5
INST4
NINST0
INST1
NINST2
INST3
NINST5
INST4
NMAN
NINST0
INST1
NINST2
NINST3
NINST4
INST5
NZZR0
A1
A2
A3
A5
A6
A7
A8
A9
A10
A14
A15
A16
B2
B1
B3
B5
B6
B8
B7
B11
B12
C1
C2
C3
C4
D1
OUTST1
B13
B14
B15
B16
A17
A18
A19
A20
A21
A22
ZZR0
NNEG
NINST4
INST5
NINST0
INST1
NINST2
NINST3
NINST4
NINST5
INST4
INST5
NINST0
NINST1
INST2
INST3
NNEG
INST4
NINST5


OUTST0*
MAN
NENX
0A1
BG
NEQ
ENY
0A2
0B1
INST0
NINST1
0A3
NINST2
NINST3
INST4
NINST5
0A4
0B2
ENY
NBG
0A5
MAN
NENY
0A6
0C1
ENX
NBG
NEQ
0A7
0B3
INST0
NINST1
0A8
NINST2
NINST3
INST4
NINST5
0A9
0B4
NBG
EQ
0A10
0D1
OUTST0
INST0
NINST1
0A12
NINST2
NINST3
INST4
NINST5
0A13
0B6
NMAM
NENY
NBG
0A14
INST0
NINST1
NINST2
0A15
0B7
0C2
NINST3
NINST4
0A16
NINST0
INST1
INST2
INST3
0B8
0NB8
0C3
0B9
0B10
0B11
0B12
0A17
0A18
0A19
0A20
0A21
0A22
0A23
NENX
NOV
NINST0
INST1
INST2
NINST3
NINST4
INST5
NINST2
INST3
NINST4
NINST5
NINST0
INST1
MAN
NINST2
NINST3
INST4
NINST0
INST1
INST2
NINST3
NINST4
NINST5

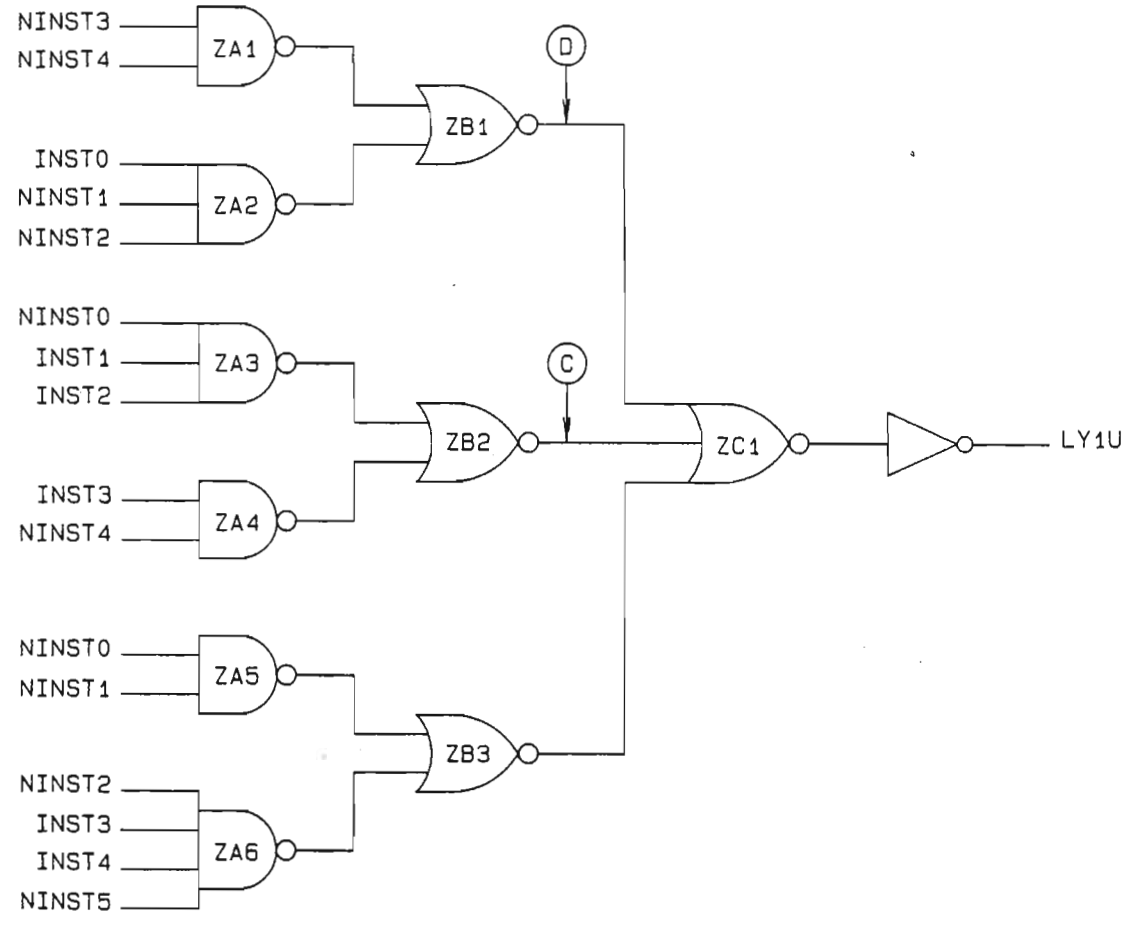

rdr
NINST0
NINST1
NINST2
NINST3
NINST4
NINST5
aeA1
aeB1
aeB2
aeB3
aeC1
NINST0
NINST1
INST2
NINST3
INST4
INST5
aeA2
aeB4
aeB5
aeC2
aeD1
rdr

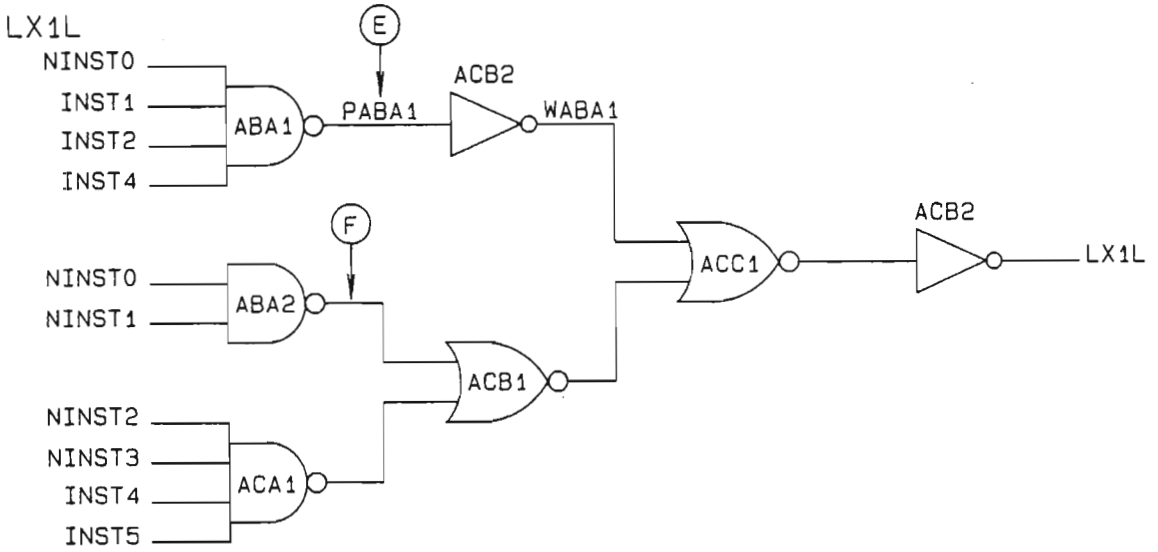

LY1L
NINST3
NINST4
INST0
NINST1
NINST2
NINST0
INST1
INST2
INST3
NINST4
NINST0
NINST1
NINST2
INST3
NINST4
INST5
D
C
AAA1
AAA2
AAB1
AAC1
LY1L
LX1U
NINST0
INST1
INST2
INST4
NINST0
NINST1
NINST2
INST3
NINST4
NINST5
E
F
ABA1
ABA2
ABA3
PABA1
ABB2
ABB1
ABC1
ABB2
LX1U

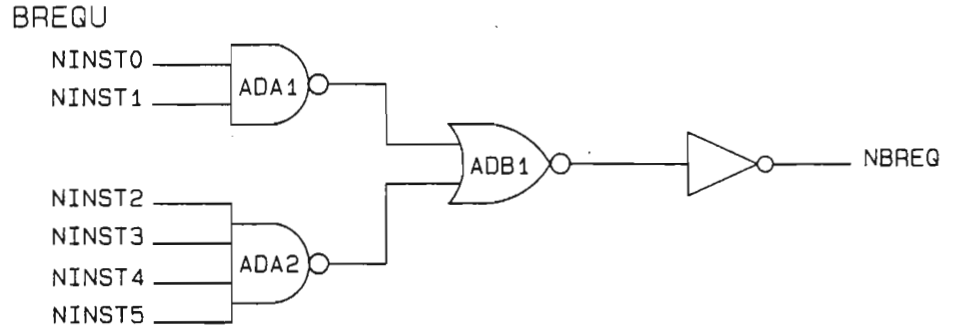
LX2L+LX2U

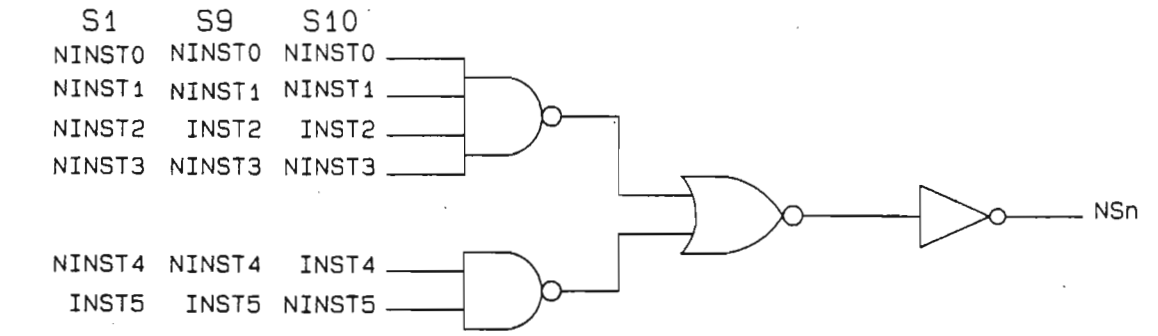


LY1U

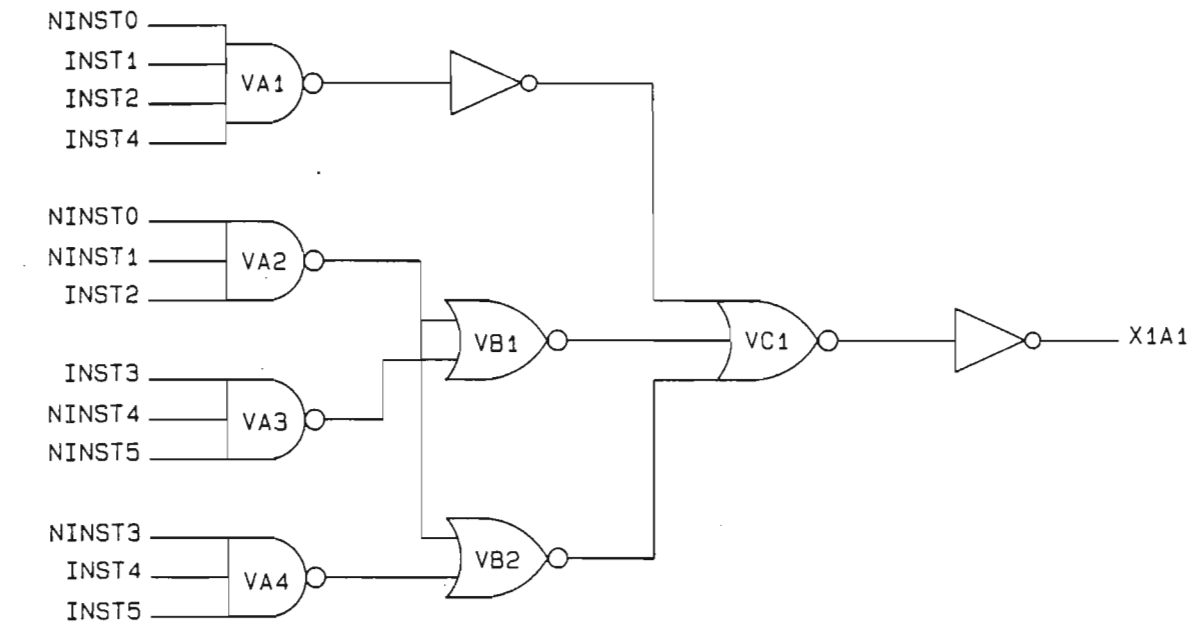



LX1L
NINST0
INST1
INST2
INST4
E
ABA1
PABA1
ACB2
WABA1
F
NINST0
NINST1
ABA2
ACB1
NINST2
NINST3
INST4
INST5
ACA1
ACC1
ACB2
LX1L

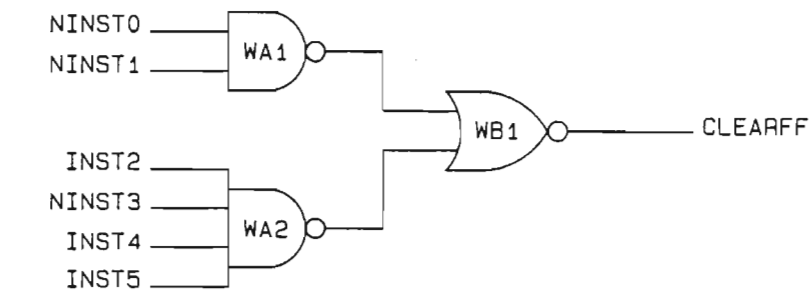

BREQU
NINST0
NINST1
ADA1
ADB1
NBREQ
NINST2
NINST3
NINST4
NINST5
ADA2


S1
S9
S10
NINST0
NINST0
NINST0
NINST1
NINST1
NINST1
NINST2
INST2
INST2
NINST3
NINST3
NINST3
NINST4
NINST4
INST4
INST5
INST5
NINST5
NSn

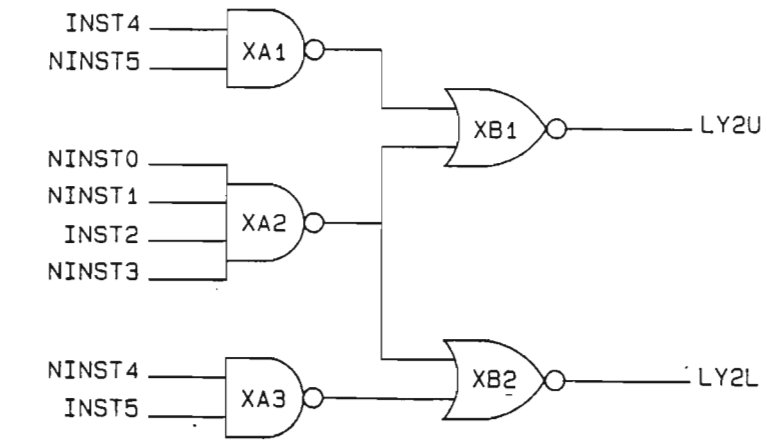
X1A1

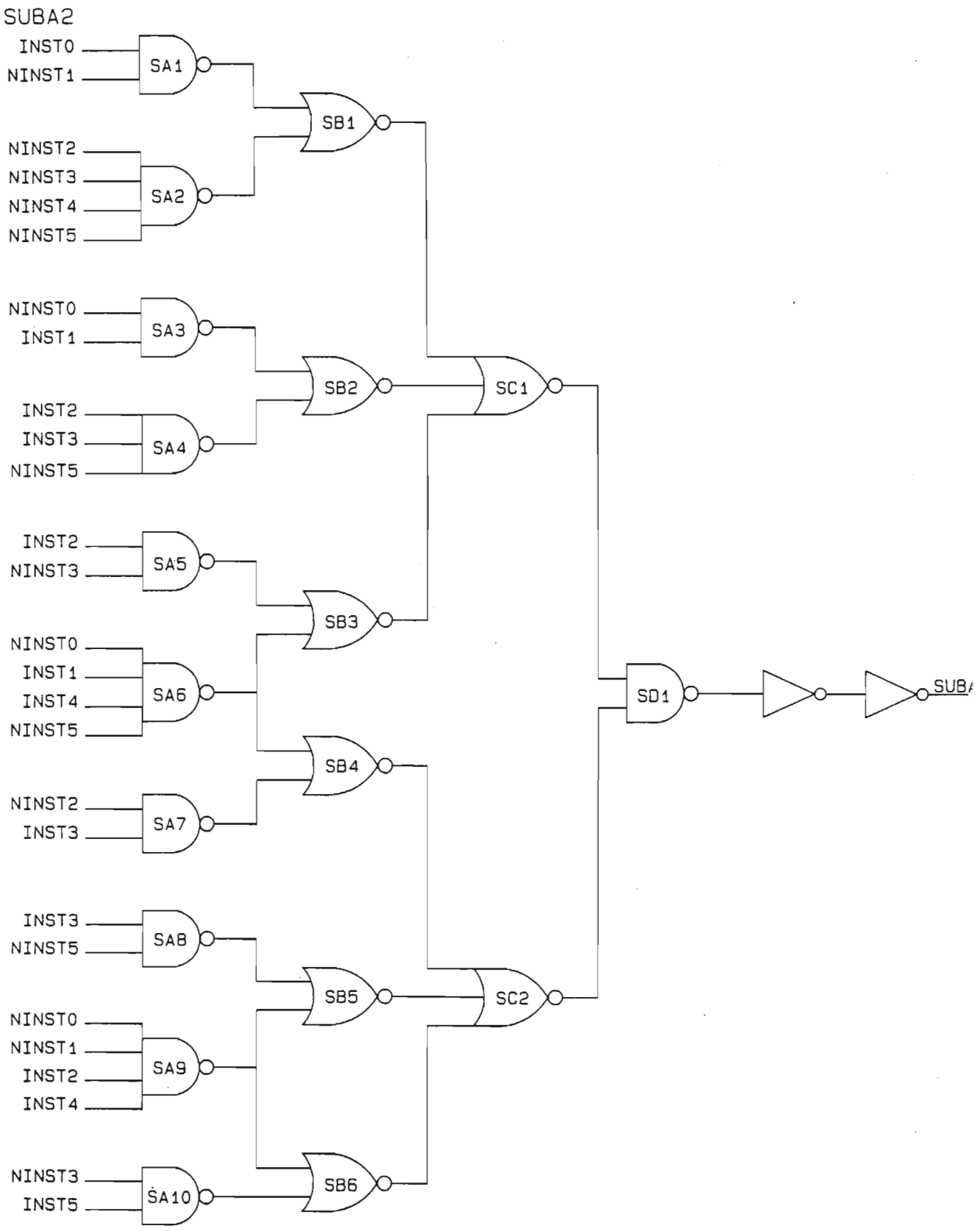


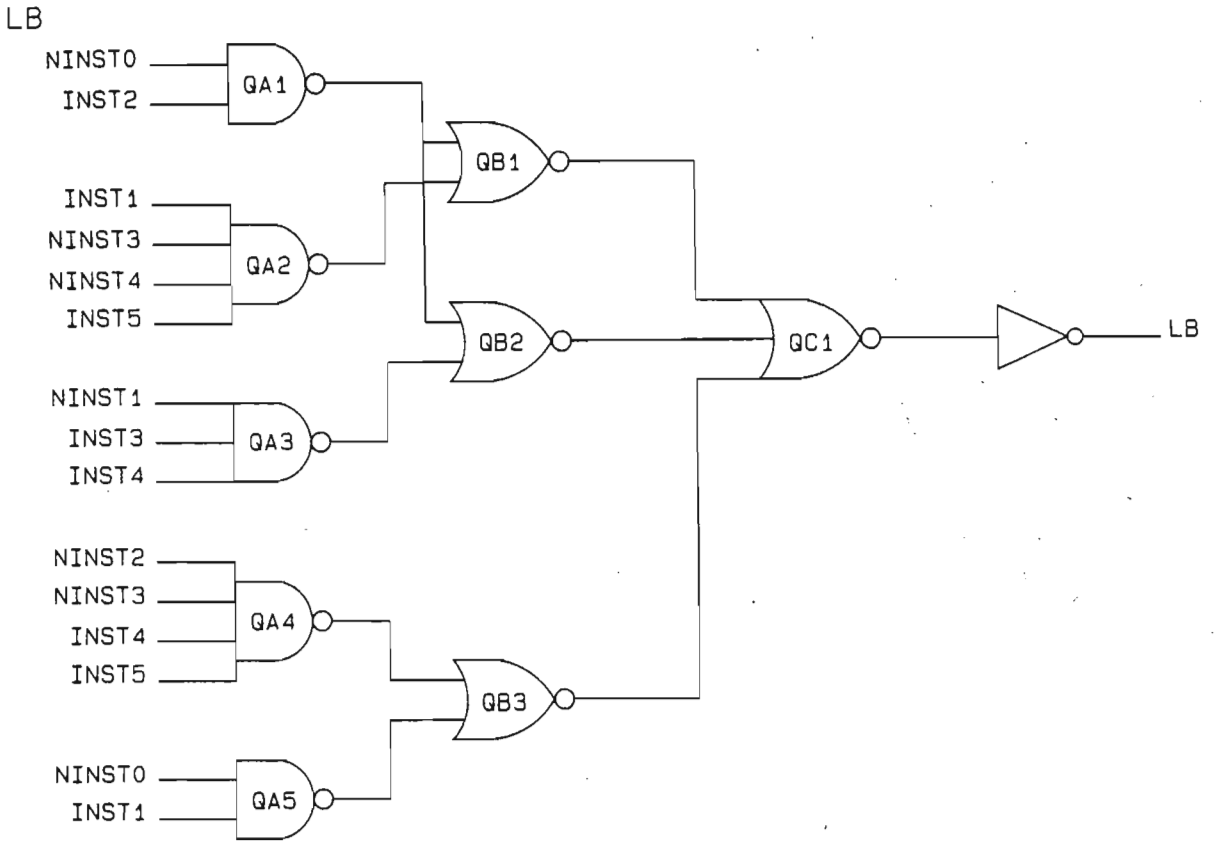
CLEARFF

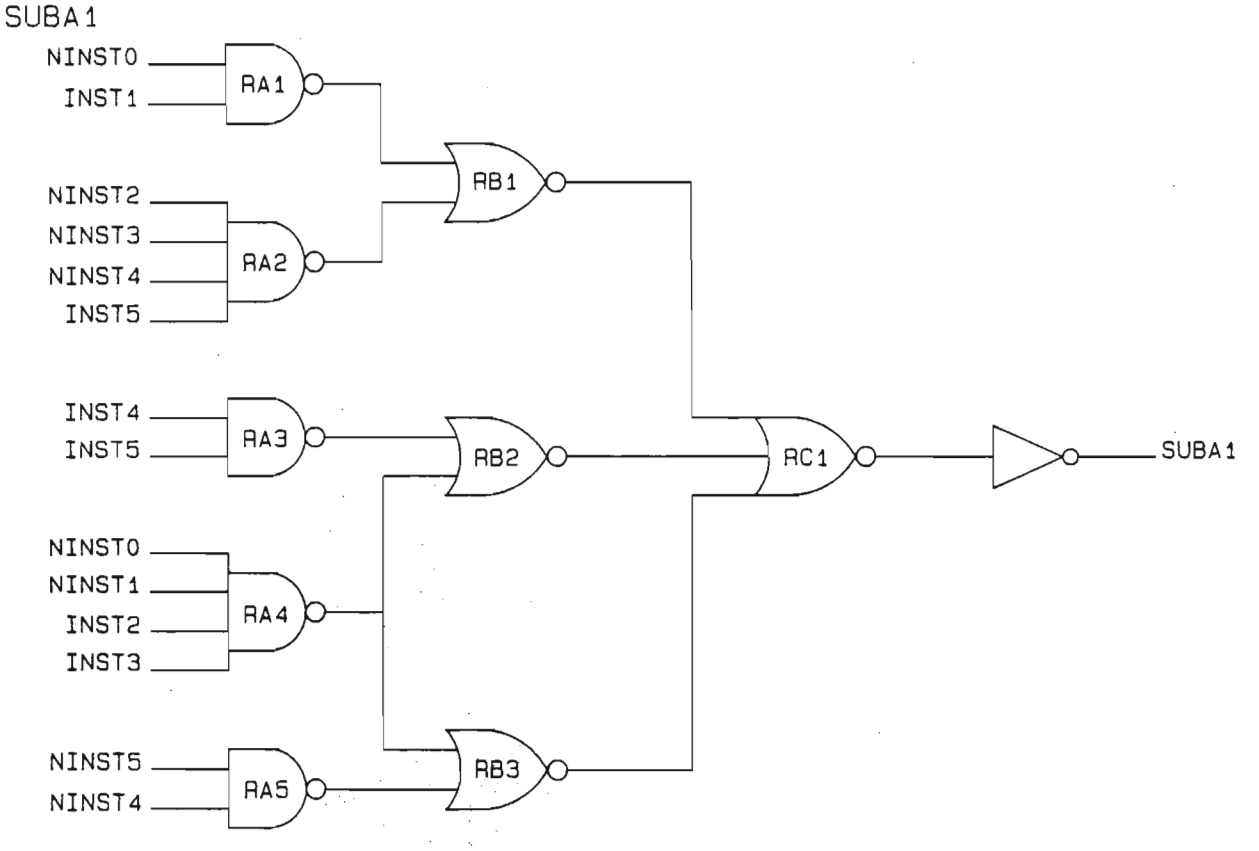


LY2L+LY2U

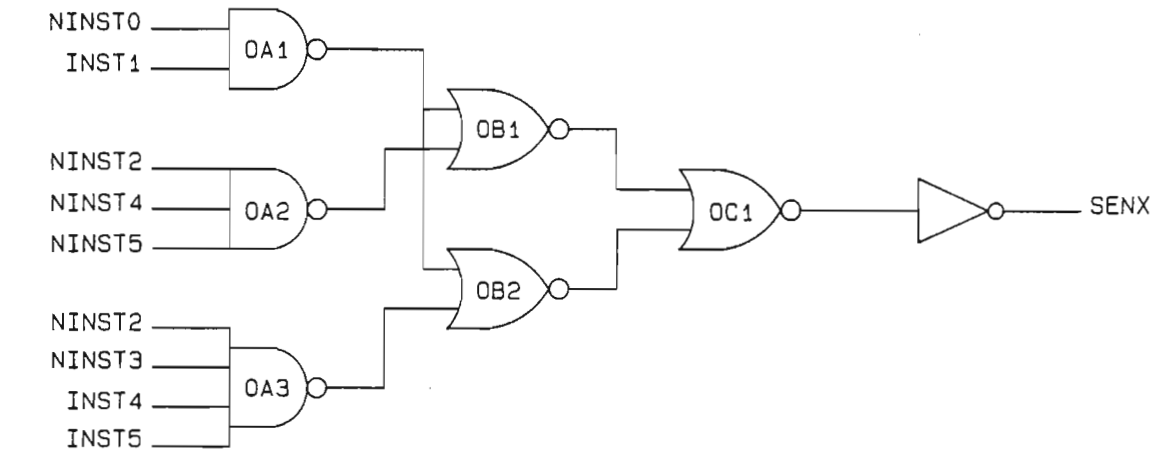



SUBA2
INST0
NINST1
SA1
NINST2
NINST3
NINST4
NINST5
SA2
SB1
NINST0
INST1
SA3
INST2
INST3
NINST5
SA4
SB2
SC1
INST2
NINST3
SA5
NINST0
INST1
INST4
NINST5
SA6
SB3
NINST2
INST3
SA7
SB4
SD1
SUB
INST3
NINST5
SA8
NINST0
NINST1
INST2
INST4
SA9
SB5
SC2
NINST3
INST5
SA10
SB6

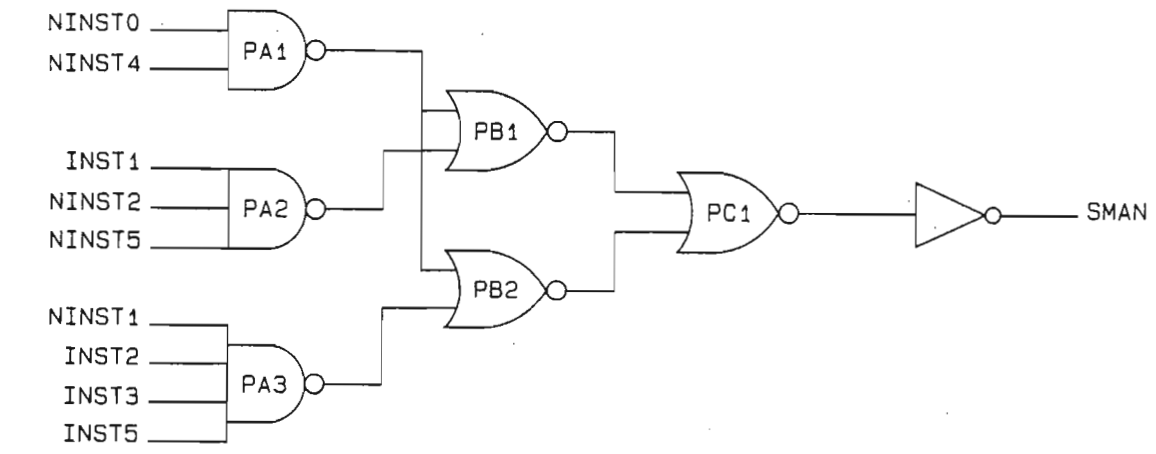

LB
NINST0
INST2
QA1
INST1
NINST3
NINST4
INST5
QA2
NINST1
INST3
INST4
QA3
NINST2
NINST3
INST4
INST5
QA4
NINST0
INST1
QA5
QB1
QB2
QB3
QC1
LB

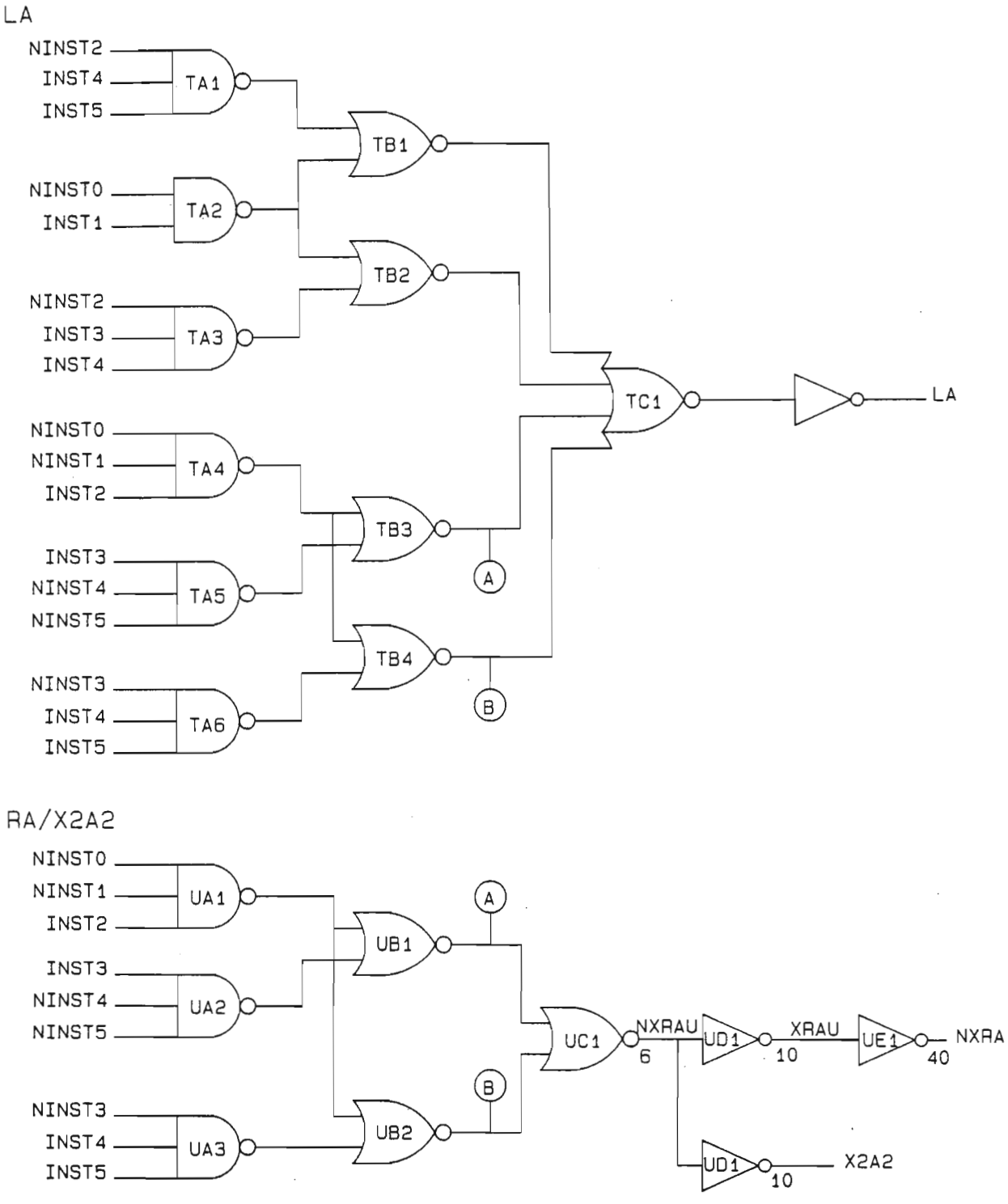

SUBA1
NINST0
INST1
RA1
NINST2
NINST3
NINST4
INST5
RA2
INST4
INST5
RA3
NINST0
NINST1
INST2
INST3
RA4
NINST5
NINST4
RA5
RB1
RB2
RB3
RC1
SUBA1

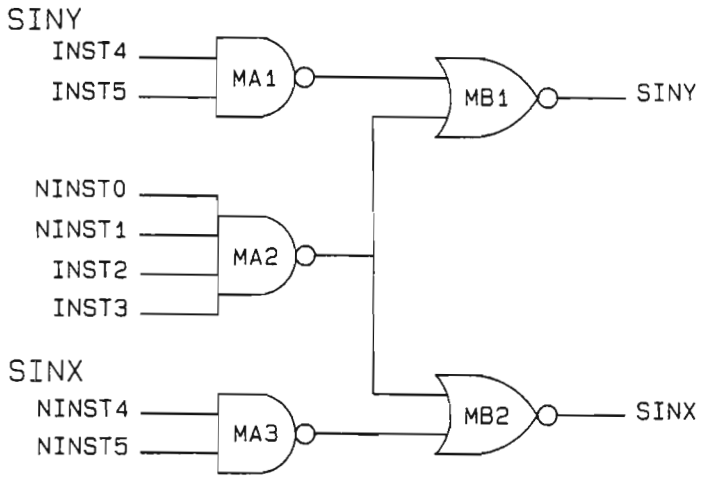
SENX

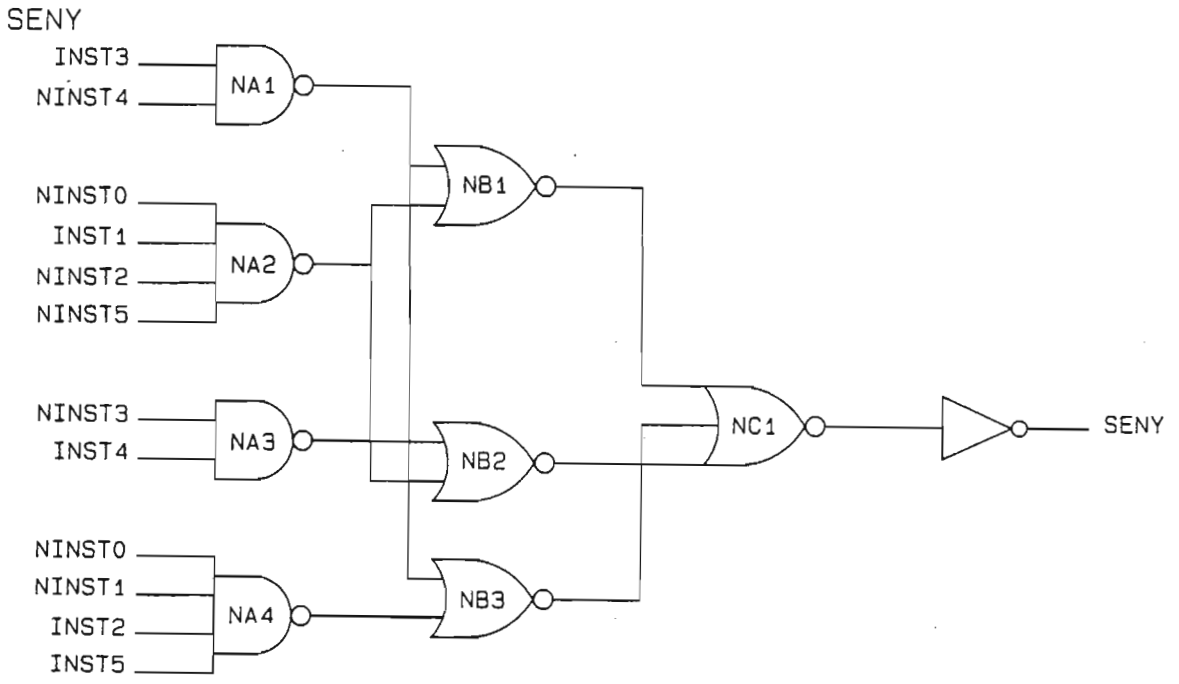

NINST0
INST1
NINST2
NINST4
NINST5
NINST2
NINST3
INST4
INST5
0A1
0A2
0A3
0B1
0B2
0C1
SENX

SMAN

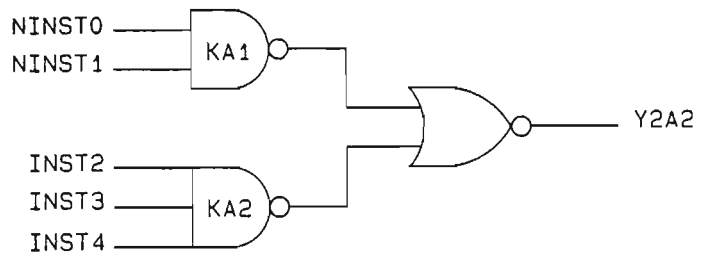

NINST0
NINST4
INST1
NINST2
NINST5
NINST1
INST2
INST3
INST5
PA1
PA2
PA3
PB1
PB2
PC1
SMAN


LA
NINST2
INST4
INST5
TA1
NINST0
INST1
TA2
NINST2
INST3
INST4
TA3
NINST0
NINST1
INST2
TA4
INST3
NINST4
NINST5
TA5
NINST3
INST4
INST5
TA6
TB1
TB2
TB3
TB4
A
B
TC1
LA
RA/X2A2
NINST0
NINST1
INST2
UA1
INST3
NINST4
NINST5
UA2
NINST3
INST4
INST5
UA3
UB1
UB2
A
B
UC1
NXRAU
6
UD1
XRAU
10
UE1
NXRA
40
UD1
10
X2A2

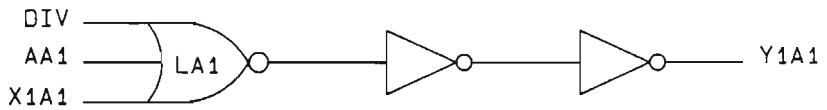

SINY
INST4
INST5
MA1
MB1
SINY
NINST0
NINST1
INST2
INST3
MA2
SINX
NINST4
NINST5
MA3
MB2
SINX


SENY
INST3
NINST4
NA1
NB1
NINST0
INST1
NINST2
NINST5
NA2
NINST3
INST4
NA3
NB2
NC1
SENY
NINST0
NINST1
INST2
INST5
NA4
NB3

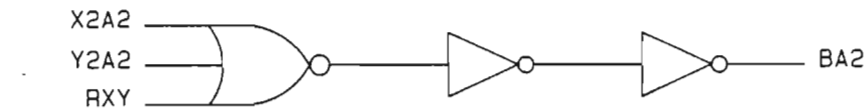
Y2A2


NINST0
NINST1
KA1
INST2
INST3
INST4
KA2
Y2A2

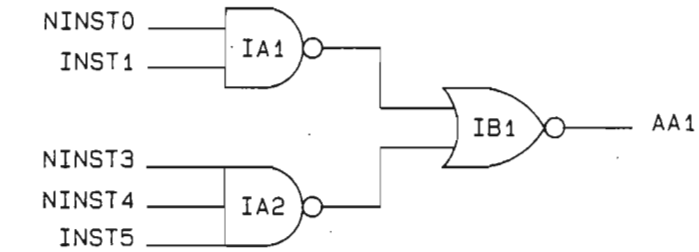
Y1A1


DIV
AA1
X1A1
LA1
Y1A1

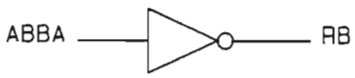
BA2



AA1

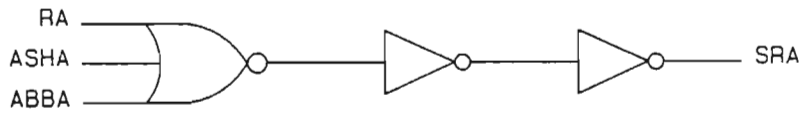


RB

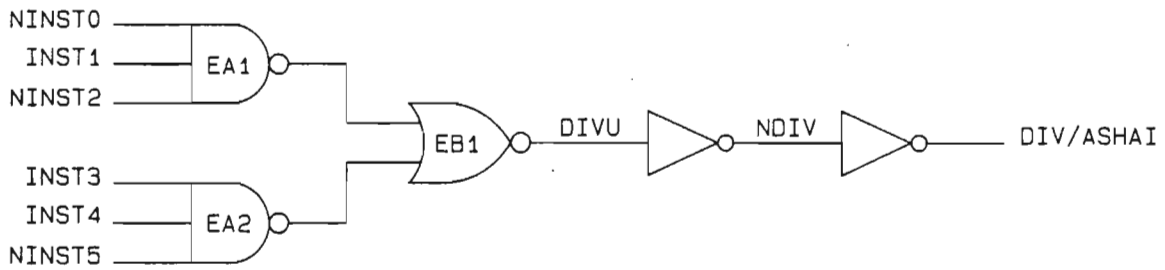


DXY

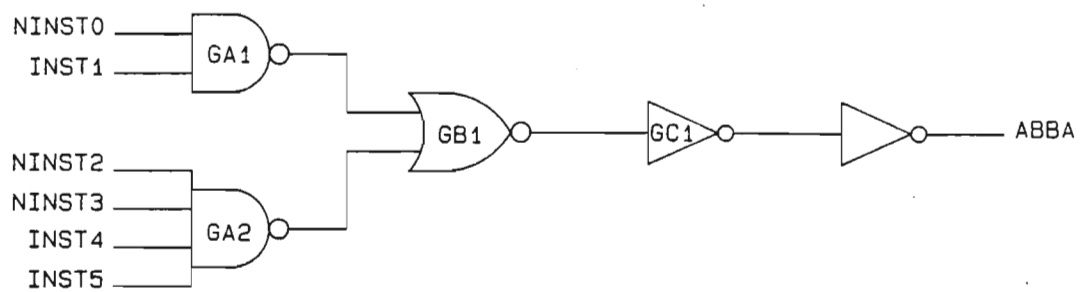
SRA

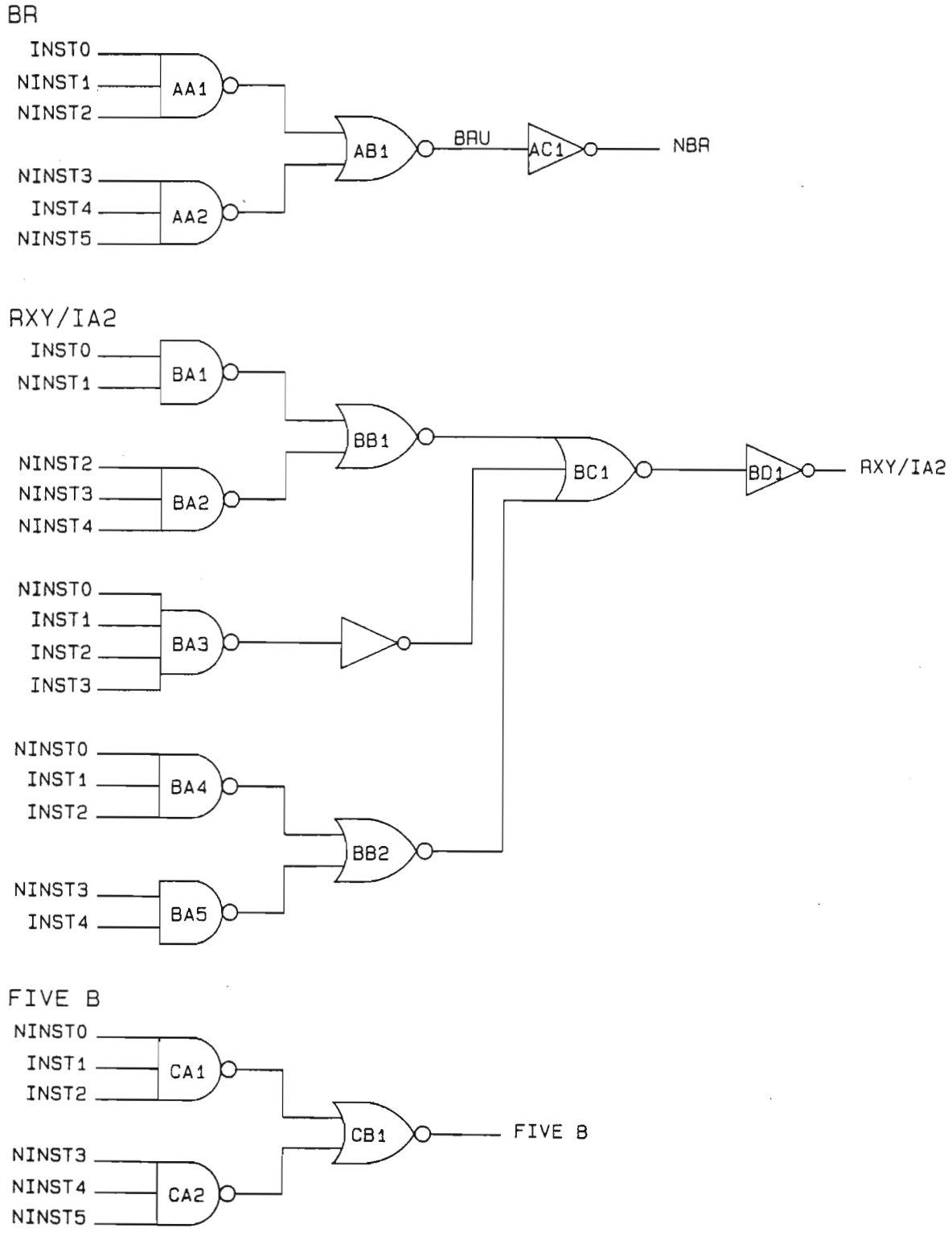


ASHAI/DIV



ABBA




BR
INST0
NINST1
NINST2
AA1
NINST3
INST4
NINST5
AA2
AB1
BRU
AC1
NBR
RXY/IA2
INST0
NINST1
BA1
NINST2
NINST3
NINST4
BA2
BB1
BC1
BD1
RXY/IA2
NINST0
INST1
INST2
INST3
BA3
NINST0
INST1
INST2
BA4
NINST3
INST4
BA5
BB2
FIVE B
NINST0
INST1
INST2
CA1
NINST3
NINST4
NINST5
CA2
CB1
FIVE B

Appendix 4.
Adder block.

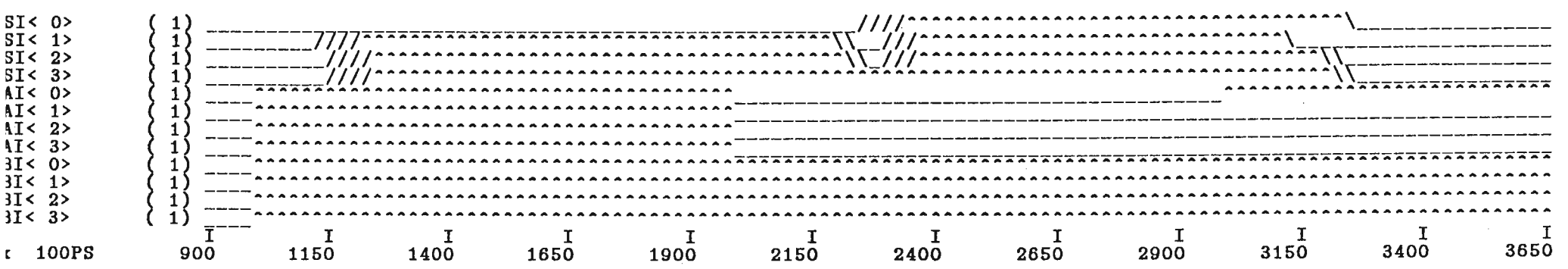
This appendix consists of a the adder block .CCL file and the appropriate user library, as well as the .SCL file used to exercise the adder and some results.

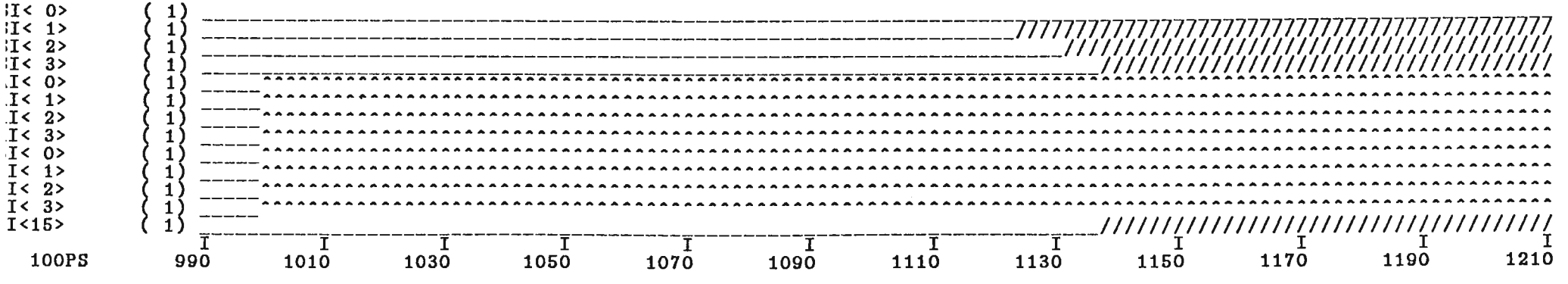
```
options
        techlib=cla3000
        userlib=adderlib
        arraysize=6000
        connect/nets/count
        iolist
        timing/alpha/all
        units=100ps
        summ
        netlist
;libdata
;        calls/techlib
;        calls/userlib
circuit
        inputs/AI<0:15>,BI<0:15>,CIN
        outputs/SI<0:15>,COUT
        declare/CI<0:14>


; carry generate/propagate block

        B1:     PGBL    [  PI<0:15>, GIBAR<0:15>, AI<0:15>, BI<0:15>  ]

; first level of carry lookahead

        B2:     BCLA    [  CI<0:2>,   PS1, GSB1,  PI<0:3>,   GIBAR<0:3>,   CIN     ]
        B3:     BCLA    [  CI<4:6>,   PS2, GSB2,  PI<4:7>,   GIBAR<4:7>,   CI<3>   ]
        B4:     BCLA    [  CI<8:10>,  PS3, GSB3,  PI<8:11>,  GIBAR<8:11>,  CI<7>   ]
        B5:     BCLA    [  CI<12:14>, PS4, GSB4,  PI<12:15>, GIBAR<12:15>, CI<11>  ]

; second level of carry lookahead with cout=gstar + cin.pstar

        B6:     BCLA    [CI<3>, CI<7>, CI<11>, PS, GSB, $
                           PS1, PS2, PS3, PS4, GSB1, GSB2, GSB3, GSB4, CIN ]
        G1:     NAND2   [PSCIN, CIN, PS ]
        G2:     NAND2   [COUT, GSB, PSCIN ]

; summer block s = p(i) .exor. c(i-1)

        B7:     SUMBL   [SI<0:15>, PI<0:15>, CIN, CI<0:14> ]
end
```

```
options
        ALL
        PERIOD=100ns
        XPROP
!        TMAX=8
        NPARAL=2
!        NPARAL=1
        SIM(1)=MAX
        SIM(2)=MIN
!faults
!  clasp
!  AUTO
!        STROBE
inits
        ai<15:0>/hex=0000
        bi<15:0>/hex=0000
        cin = 0
inputs
        ai<15:0>/hex = 0000 > 1, FFFF > 2, 0000 > 3, 0001 > 4, 0001 > 5, 0001 > 6
        bi<15:0>/hex = 0000 > 1, FFFF > 2, 0A0F > 3, FFFF > 4, 000F > 5, 0004 > 6
end
```

C L A S S I C D I S P L A Y E D I T O R

```
Circuit compiled using CLASSIC  compiler  version    E.2   on  3-DEC-86 at 14:58:35
Circuit file spec DUA0:[IZZARD.DIRECTORY.CLASSIC.RA3000.THESIS]ADD.CCL;7
Circuit name ADD
Compilation time units   100PS

Circuit simulated using CLASSIC  simulator  version    E.2  on  3-DEC-86 at 14:59:34
Simulation file spec DUA0:[IZZARD.DIRECTORY.CLASSIC.RA3000.THESIS]ADD.SCL;2
Simulation name ADD
Simulation period   100NS
Min_mask(01) Typ_mask(00) Max_mask(10)
Simulation to spec : Supply =   5V  10%  Temp =    0->  70
Circuit simulated with X propagation ON

ZZARD                                    16-JAN-87 14:55:56   CLASSIC display editor version  E.2                Page    1
erminal                                  CLA3000   V1R2       DUA0:[IZZARD.DIRECTORY.CLASSIC.RA3000.THESIS]ADD.DAT;1
```





```
options
        techlib=cla3000


        arraysize=6000
;       connect/nets/count
;       iolist
;       timing/alpha/all
        units=100ps
        summ
;       netlist


define

;*******************************************************
;   pg cell with asserted low gi output     ***********
;*******************************************************

subcir/ucatal
pgbar [pi,gibar:ai,bi]
        G1:     NAND2   [gibar,AI,BI]
        G2:     NAND2   [AABI,AI,gibar]
        G3:     NAND2   [ABBI,gibar,BI]
        G5:     NAND2   [PI,AABI,ABBI]

endm

;*******************************************************
; SINGLE PG CELL                                       *
;*******************************************************

subcir/ucatal

pg [pi,gi:ai,bi]
        G1:     NAND2   [ABI,AI,BI]
        G2:     NAND2   [AABI,AI,ABI]
        G3:     NAND2   [ABBI,ABI,BI]
        G4:     2INV    [,GI,+,ABI]
        G5:     NAND2   [PI,AABI,ABBI]

endm

;*********************************************
;16 BIT PG BLOCK using asserted high gi ********
;*********************************************

subcir/ucatal

        PGBLAH  [PI<0:15>,GI<0:15>:  AI<0:15>,BI<0:15>]

  ;using the pg block from adderlib create a 16 bit pg block

  SUPPLIES/+VDD

;       c0 :    PG      [PI<0>,GI<0>,  AI<0>,BI<0>]
 C0__G1  : NAND2[C0__ABI,AI<0>,BI<0>]
 C0__G2  : NAND2[C0__AABI,AI<0>,C0__ABI]
 C0__G3  : NAND2[C0__ABBI,C0__ABI,BI<0>]
 C0__G4  : 2INV[,GI<0>,VDD,C0__ABI]
 C0__G5  : NAND2[PI<0>,C0__AABI,C0__ABBI]
;       c1 :    PG      [PI<1>,GI<1>,  AI<1>,BI<1>]
 C1__G1  : NAND2[C1__ABI,AI<1>,BI<1>]
 C1__G2  : NAND2[C1__AABI,AI<1>,C1__ABI]
 C1__G3  : NAND2[C1__ABBI,C1__ABI,BI<1>]
 C1__G4  : 2INV[,GI<1>,VDD,C1__ABI]
 C1__G5  : NAND2[PI<1>,C1__AABI,C1__ABBI]
;       c2 :    PG      [PI<2>,GI<2>,  AI<2>,BI<2>]
 C2__G1  : NAND2[C2__ABI,AI<2>,BI<2>]
 C2__G2  : NAND2[C2__AABI,AI<2>,C2__ABI]
 C2__G3  : NAND2[C2__ABBI,C2__ABI,BI<2>]
 C2__G4  : 2INV[,GI<2>,VDD,C2__ABI]
 C2__G5  : NAND2[PI<2>,C2__AABI,C2__ABBI]
;       c3 :    PG      [PI<3>,GI<3>,  AI<3>,BI<3>]
 C3__G1  : NAND2[C3__ABI,AI<3>,BI<3>]
 C3__G2  : NAND2[C3__AABI,AI<3>,C3__ABI]
 C3__G3  : NAND2[C3__ABBI,C3__ABI,BI<3>]
 C3__G4  : 2INV[,GI<3>,VDD,C3__ABI]
 C3__G5  : NAND2[PI<3>,C3__AABI,C3__ABBI]
;       c4 :    PG      [PI<4>,GI<4>,  AI<4>,BI<4>]
 C4__G1  : NAND2[C4__ABI,AI<4>,BI<4>]
 C4__G2  : NAND2[C4__AABI,AI<4>,C4__ABI]
 C4__G3  : NAND2[C4__ABBI,C4__ABI,BI<4>]
 C4__G4  : 2INV[,GI<4>,VDD,C4__ABI]
```

```
C4__G5  : NAND2[PI<4>,C4__AABI,C4__ABBI]
;       c5 :    PG      [PI<5>,GI<5>,  AI<5>,BI<5>]
C5__G1  : NAND2[C5__ABI,AI<5>,BI<5>]
C5__G2  : NAND2[C5__AABI,AI<5>,C5__ABI]
C5__G3  : NAND2[C5__ABBI,C5__ABI,BI<5>]
C5__G4  : 2INV[,GI<5>,VDD,C5__ABI]
C5__G5  : NAND2[PI<5>,C5__AABI,C5__ABBI]
;       c6 :    PG      [PI<6>,GI<6>,  AI<6>,BI<6>]
C6__G1  : NAND2[C6__ABI,AI<6>,BI<6>]
C6__G2  : NAND2[C6__AABI,AI<6>,C6__ABI]
C6__G3  : NAND2[C6__ABBI,C6__ABI,BI<6>]
C6__G4  : 2INV[,GI<6>,VDD,C6__ABI]
C6__G5  : NAND2[PI<6>,C6__AABI,C6__ABBI]
;       c7 :    PG      [PI<7>,GI<7>,  AI<7>,BI<7>]
C7__G1  : NAND2[C7__ABI,AI<7>,BI<7>]
C7__G2  : NAND2[C7__AABI,AI<7>,C7__ABI]
C7__G3  : NAND2[C7__ABBI,C7__ABI,BI<7>]
C7__G4  : 2INV[,GI<7>,VDD,C7__ABI]
C7__G5  : NAND2[PI<7>,C7__AABI,C7__ABBI]
;       c8 :    PG      [PI<8>,GI<8>,  AI<8>,BI<8>]
C8__G1  : NAND2[C8__ABI,AI<8>,BI<8>]
C8__G2  : NAND2[C8__AABI,AI<8>,C8__ABI]
C8__G3  : NAND2[C8__ABBI,C8__ABI,BI<8>]
C8__G4  : 2INV[,GI<8>,VDD,C8__ABI]
C8__G5  : NAND2[PI<8>,C8__AABI,C8__ABBI]
;       c9 :    PG      [PI<9>,GI<9>,  AI<9>,BI<9>]
C9__G1  : NAND2[C9__ABI,AI<9>,BI<9>]
C9__G2  : NAND2[C9__AABI,AI<9>,C9__ABI]
C9__G3  : NAND2[C9__ABBI,C9__ABI,BI<9>]
C9__G4  : 2INV[,GI<9>,VDD,C9__ABI]
C9__G5  : NAND2[PI<9>,C9__AABI,C9__ABBI]
;       c10:    PG      [PI<10>,GI<10>,  AI<10>,BI<10>]
C10_G1  : NAND2[C10_ABI,AI<10>,BI<10>]
C10_G2  : NAND2[C10_AABI,AI<10>,C10_ABI]
C10_G3  : NAND2[C10_ABBI,C10_ABI,BI<10>]
C10_G4  : 2INV[,GI<10>,VDD,C10_ABI]
C10_G5  : NAND2[PI<10>,C10_AABI,C10_ABBI]
;       c11:    PG      [PI<11>,GI<11>,  AI<11>,BI<11>]
C11_G1  : NAND2[C11_ABI,AI<11>,BI<11>]
C11_G2  : NAND2[C11_AABI,AI<11>,C11_ABI]
C11_G3  : NAND2[C11_ABBI,C11_ABI,BI<11>]
C11_G4  : 2INV[,GI<11>,VDD,C11_ABI]
C11_G5  : NAND2[PI<11>,C11_AABI,C11_ABBI]
;       c12:    PG      [PI<12>,GI<12>,  AI<12>,BI<12>]
C12_G1  : NAND2[C12_ABI,AI<12>,BI<12>]
C12_G2  : NAND2[C12_AABI,AI<12>,C12_ABI]
C12_G3  : NAND2[C12_ABBI,C12_ABI,BI<12>]
C12_G4  : 2INV[,GI<12>,VDD,C12_ABI]
C12_G5  : NAND2[PI<12>,C12_AABI,C12_ABBI]
;       c13:    PG      [PI<13>,GI<13>,  AI<13>,BI<13>]
C13_G1  : NAND2[C13_ABI,AI<13>,BI<13>]
C13_G2  : NAND2[C13_AABI,AI<13>,C13_ABI]
C13_G3  : NAND2[C13_ABBI,C13_ABI,BI<13>]
C13_G4  : 2INV[,GI<13>,VDD,C13_ABI]
C13_G5  : NAND2[PI<13>,C13_AABI,C13_ABBI]
;       c14:    PG      [PI<14>,GI<14>,  AI<14>,BI<14>]
C14_G1  : NAND2[C14_ABI,AI<14>,BI<14>]
C14_G2  : NAND2[C14_AABI,AI<14>,C14_ABI]
C14_G3  : NAND2[C14_ABBI,C14_ABI,BI<14>]
C14_G4  : 2INV[,GI<14>,VDD,C14_ABI]
C14_G5  : NAND2[PI<14>,C14_AABI,C14_ABBI]
;       c15:    PG      [PI<15>,GI<15>,  AI<15>,BI<15>]
C15_G1  : NAND2[C15_ABI,AI<15>,BI<15>]
C15_G2  : NAND2[C15_AABI,AI<15>,C15_ABI]
C15_G3  : NAND2[C15_ABBI,C15_ABI,BI<15>]
C15_G4  : 2INV[,GI<15>,VDD,C15_ABI]
C15_G5  : NAND2[PI<15>,C15_AABI,C15_ABBI]
ENDM
;*************************************************************************
;   pg block using asserted low gi    ******************************
;*************************************************************************

subcir/ucatal
        PGBL    [PI<0:15>,gibar<0:15>:  AI<0:15>,BI<0:15>]

  ;using the pg block from adderlib create a 16 bit pg block

;       c0 :    pgbar   [PI<0>,gibar<0>,  AI<0>,BI<0>]
C0__G1  : NAND2[GIBAR<0>,AI<0>,BI<0>]
C0__G2  : NAND2[C0__AABI,AI<0>,GIBAR<0>]
C0__G3  : NAND2[C0__ABBI,GIBAR<0>,BI<0>]
C0__G5  : NAND2[PI<0>,C0__AABI,C0__ABBI]
;       c1 :    pgbar   [PI<1>,gibar<1>,  AI<1>,BI<1>]
C1__G1  : NAND2[GIBAR<1>,AI<1>,BI<1>]
C1__G2  : NAND2[C1__AABI,AI<1>,GIBAR<1>]
```

```
 C1__G3  : NAND2[C1__ABBI,GIBAR<1>,BI<1>]
 C1__G5  : NAND2[PI<1>,C1__AABI,C1__ABBI]
;        c2 :    pgbar   [PI<2>,gibar<2>,  AI<2>,BI<2>]
 C2__G1  : NAND2[GIBAR<2>,AI<2>,BI<2>]
 C2__G2  : NAND2[C2__AABI,AI<2>,GIBAR<2>]
 C2__G3  : NAND2[C2__ABBI,GIBAR<2>,BI<2>]
 C2__G5  : NAND2[PI<2>,C2__AABI,C2__ABBI]
;        c3 :    pgbar   [PI<3>,gibar<3>,  AI<3>,BI<3>]
 C3__G1  : NAND2[GIBAR<3>,AI<3>,BI<3>]
 C3__G2  : NAND2[C3__AABI,AI<3>,GIBAR<3>]
 C3__G3  : NAND2[C3__ABBI,GIBAR<3>,BI<3>]
 C3__G5  : NAND2[PI<3>,C3__AABI,C3__ABBI]
;        c4 :    pgbar   [PI<4>,gibar<4>,  AI<4>,BI<4>]
 C4__G1  : NAND2[GIBAR<4>,AI<4>,BI<4>]
 C4__G2  : NAND2[C4__AABI,AI<4>,GIBAR<4>]
 C4__G3  : NAND2[C4__ABBI,GIBAR<4>,BI<4>]
 C4__G5  : NAND2[PI<4>,C4__AABI,C4__ABBI]
;        c5 :    pgbar   [PI<5>,gibar<5>,  AI<5>,BI<5>]
 C5__G1  : NAND2[GIBAR<5>,AI<5>,BI<5>]
 C5__G2  : NAND2[C5__AABI,AI<5>,GIBAR<5>]
 C5__G3  : NAND2[C5__ABBI,GIBAR<5>,BI<5>]
 C5__G5  : NAND2[PI<5>,C5__AABI,C5__ABBI]
;        c6 :    pgbar   [PI<6>,gibar<6>,  AI<6>,BI<6>]
 C6__G1  : NAND2[GIBAR<6>,AI<6>,BI<6>]
 C6__G2  : NAND2[C6__AABI,AI<6>,GIBAR<6>]
 C6__G3  : NAND2[C6__ABBI,GIBAR<6>,BI<6>]
 C6__G5  : NAND2[PI<6>,C6__AABI,C6__ABBI]
;        c7 :    pgbar   [PI<7>,gibar<7>,  AI<7>,BI<7>]
 C7__G1  : NAND2[GIBAR<7>,AI<7>,BI<7>]
 C7__G2  : NAND2[C7__AABI,AI<7>,GIBAR<7>]
 C7__G3  : NAND2[C7__ABBI,GIBAR<7>,BI<7>]
 C7__G5  : NAND2[PI<7>,C7__AABI,C7__ABBI]
;        c8 :    pgbar   [PI<8>,gibar<8>,  AI<8>,BI<8>]
 C8__G1  : NAND2[GIBAR<8>,AI<8>,BI<8>]
 C8__G2  : NAND2[C8__AABI,AI<8>,GIBAR<8>]
 C8__G3  : NAND2[C8__ABBI,GIBAR<8>,BI<8>]
 C8__G5  : NAND2[PI<8>,C8__AABI,C8__ABBI]
;        c9 :    pgbar   [PI<9>,gibar<9>,  AI<9>,BI<9>]
 C9__G1  : NAND2[GIBAR<9>,AI<9>,BI<9>]
 C9__G2  : NAND2[C9__AABI,AI<9>,GIBAR<9>]
 C9__G3  : NAND2[C9__ABBI,GIBAR<9>,BI<9>]
 C9__G5  : NAND2[PI<9>,C9__AABI,C9__ABBI]
;        c10:    pgbar   [PI<10>,gibar<10>,  AI<10>,BI<10>]
 C10_G1  : NAND2[GIBAR<10>,AI<10>,BI<10>]
 C10_G2  : NAND2[C10_AABI,AI<10>,GIBAR<10>]
 C10_G3  : NAND2[C10_ABBI,GIBAR<10>,BI<10>]
 C10_G5  : NAND2[PI<10>,C10_AABI,C10_ABBI]
;        c11:    pgbar   [PI<11>,gibar<11>,  AI<11>,BI<11>]
 C11_G1  : NAND2[GIBAR<11>,AI<11>,BI<11>]
 C11_G2  : NAND2[C11_AABI,AI<11>,GIBAR<11>]
 C11_G3  : NAND2[C11_ABBI,GIBAR<11>,BI<11>]
 C11_G5  : NAND2[PI<11>,C11_AABI,C11_ABBI]
;        c12:    pgbar   [PI<12>,gibar<12>,  AI<12>,BI<12>]
 C12_G1  : NAND2[GIBAR<12>,AI<12>,BI<12>]
 C12_G2  : NAND2[C12_AABI,AI<12>,GIBAR<12>]
 C12_G3  : NAND2[C12_ABBI,GIBAR<12>,BI<12>]
 C12_G5  : NAND2[PI<12>,C12_AABI,C12_ABBI]
;        c13:    pgbar   [PI<13>,gibar<13>,  AI<13>,BI<13>]
 C13_G1  : NAND2[GIBAR<13>,AI<13>,BI<13>]
 C13_G2  : NAND2[C13_AABI,AI<13>,GIBAR<13>]
 C13_G3  : NAND2[C13_ABBI,GIBAR<13>,BI<13>]
 C13_G5  : NAND2[PI<13>,C13_AABI,C13_ABBI]
;        c14:    pgbar   [PI<14>,gibar<14>,  AI<14>,BI<14>]
 C14_G1  : NAND2[GIBAR<14>,AI<14>,BI<14>]
 C14_G2  : NAND2[C14_AABI,AI<14>,GIBAR<14>]
 C14_G3  : NAND2[C14_ABBI,GIBAR<14>,BI<14>]
 C14_G5  : NAND2[PI<14>,C14_AABI,C14_ABBI]
;        c15:    pgbar   [PI<15>,gibar<15>,  AI<15>,BI<15>]
 C15_G1  : NAND2[GIBAR<15>,AI<15>,BI<15>]
 C15_G2  : NAND2[C15_AABI,AI<15>,GIBAR<15>]
 C15_G3  : NAND2[C15_ABBI,GIBAR<15>,BI<15>]
 C15_G5  : NAND2[PI<15>,C15_AABI,C15_ABBI]
ENDM


;*************************************************************************
;16 BIT SUMMER BLOCK (just a string of xor gates)                  *******
;*************************************************************************
subcir/ucatal
SUMBL   [SI<0:15>: PI<0:15>,CIN,CI<0:14>]
        G0:     EXOR[ SI<0> ,,,,PI<0>, CIN ,-,-,-]
        G1:     EXOR[ SI<1> ,,,,PI<1>, CI<0>,-,-,-]
        G2:     EXOR[ SI<2> ,,,,PI<2>, CI<1>,-,-,-]
```

```
        G3:     EXOR[ SI<3> ,,,,PI<3>, CI<2>,-,-,-]
        G4:     EXOR[ SI<4> ,,,,PI<4>, CI<3>,-,-,-]
        G5:     EXOR[ SI<5> ,,,,PI<5>, CI<4>,-,-,-]
        G6:     EXOR[ SI<6> ,,,,PI<6>, CI<5>,-,-,-]
        G7:     EXOR[ SI<7> ,,,,PI<7>, CI<6>,-,-,-]
        G8:     EXOR[ SI<8> ,,,,PI<8>, CI<7>,-,-,-]
        G9:     EXOR[ SI<9> ,,,,PI<9>, CI<8>,-,-,-]
        G10:    EXOR[ SI<10> ,,,,PI<10>, CI<9>,-,-,-]
        G11:    EXOR[ SI<11> ,,,,PI<11>, CI<10>,-,-,-]
        G12:    EXOR[ SI<12> ,,,,PI<12>, CI<11>,-,-,-]
        G13:    EXOR[ SI<13> ,,,,PI<13>, CI<12>,-,-,-]
        G14:    EXOR[ SI<14> ,,,,PI<14>, CI<13>,-,-,-]
        G15:    EXOR[ SI<15> ,,,,PI<15>, CI<14>,-,-,-]

endm
;************************************************************
; 4 bit bcla unit                              ****************
;************************************************************

subcir/ucatal
        bcla [ci<0:2>, pstar, gstBar :  pi<0:3>, gibar<0:3>,cin]

        G1:     NAND2   [ci<0>,gibar<0>,pc]
        G2:     NAND2   [pc,cin,pi<0>]
        G3:     NAND2   [pg0,g0,pi<1>]
        G4:     NAND2   [pg1,g1,pi<2>]
        G5:     NAND2   [pg2,g2,pi<3>]

; use extra inverters from the first
;three 3nands to produce high asserted g0 g1 g2

        G6:     NAND3   [ci<1>, g0 , gibar<1>, pg0, ppc, gibar<0>  ]
        G7:     NAND3   [ppc, g1 , pi<1>, pi<0>, cin, gibar<1>  ]
        G8:     NAND3   [ppg0, g2 , pi<2>, pi<1>, g0, gibar<2>  ]
        G9:     NAND3   [ppg1, pstar, pi<3>, pi<2>, g1, pstbar   ]

        G10:    NAND4   [pppc, pi<2>, pi<1>, pi<0>, cin]
        G11:    NAND4   [ci<2>, gibar<2>, pg1, ppg0, pppc]
        G12:    NAND4   [pppg0, g0, pi<1>, pi<2>, pi<3>]
        G13:    NAND4   [gstar, gibar<3>, pg2, ppg1, pppg0]
        G14:    NAND4   [pstbar, pi<0>, pi<1>, pi<2>, pi<3>]

        G15:    INV4    [gstBar, gstar]
endm
;*************************************************************************
;*******   REGISTERS VARIOUS                      *********************
;************************************************************************


SUBCIR/UCATAL

        SHR [ Q<0:15>  :  EXCK, SLI ]

        G1:     CKB     [ CK, CKBAR, EXCK ]

        D0:     DT      [ Q< 0>, , CK, CKBAR, SLI  ]
        D1:     DT      [ Q< 1>, , CK, CKBAR, Q< 0> ]
        D2:     DT      [ Q< 2>, , CK, CKBAR, Q< 1> ]
        D3:     DT      [ Q< 3>, , CK, CKBAR, Q< 2> ]
        D4:     DT      [ Q< 4>, , CK, CKBAR, Q< 3> ]
        D5:     DT      [ Q< 5>, , CK, CKBAR, Q< 4> ]
        D6:     DT      [ Q< 6>, , CK, CKBAR, Q< 5> ]
        D7:     DT      [ Q< 7>, , CK, CKBAR, Q< 6> ]
        D8:     DT      [ Q< 8>, , CK, CKBAR, Q< 7> ]
        D9:     DT      [ Q< 9>, , CK, CKBAR, Q< 8> ]
        D10:    DT      [ Q<10>, , CK, CKBAR, Q< 9> ]
        D11:    DT      [ Q<11>, , CK, CKBAR, Q<10> ]
        D12:    DT      [ Q<12>, , CK, CKBAR, Q<11> ]
        D13:    DT      [ Q<13>, , CK, CKBAR, Q<12> ]
        D14:    DT      [ Q<14>, , CK, CKBAR, Q<13> ]
        D15:    DT      [ Q<15>, , CK, CKBAR, Q<14> ]
ENDM
SUBCIR/UCATAL

        ABREG [ Q<0:15>  :  D<0:15>, EXCK, LOAD, CLR ]

        G1:     NAND2   [ INCK, EXCK, LOAD ]
        G2:     INV8    [ R,  CLR ]

        G3:     CKB     [ CK, CKBAR, INCK ]

        D0:     DTRS    [ Q< 0>, , CK, CKBAR, D< 0>, R, +]
        D1:     DTRS    [ Q< 1>, , CK, CKBAR, D< 1>, R, +]
        D2:     DTRS    [ Q< 2>, , CK, CKBAR, D< 2>, R, +]
        D3:     DTRS    [ Q< 3>, , CK, CKBAR, D< 3>, R, +]
```

```
        D4:     DTRS    [ Q< 4>, , CK, CKBAR, D< 4>, R, + ]
        D5:     DTRS    [ Q< 5>, , CK, CKBAR, D< 5>, R, + ]
        D6:     DTRS    [ Q< 6>, , CK, CKBAR, D< 6>, R, + ]
        D7:     DTRS    [ Q< 7>, , CK, CKBAR, D< 7>, R, + ]
        D8:     DTRS    [ Q< 8>, , CK, CKBAR, D< 8>, R, + ]
        D9:     DTRS    [ Q< 9>, , CK, CKBAR, D< 9>, R, + ]
        D10:    DTRS    [ Q<10>, , CK, CKBAR, D<10>, R, + ]
        D11:    DTRS    [ Q<11>, , CK, CKBAR, D<11>, R, + ]
        D12:    DTRS    [ Q<12>, , CK, CKBAR, D<12>, R, + ]
        D13:    DTRS    [ Q<13>, , CK, CKBAR, D<13>, R, + ]
        D14:    DTRS    [ Q<14>, , CK, CKBAR, D<14>, R, + ]
        D15:    DTRS    [ Q<15>, , CK, CKBAR, D<15>, R, + ]
ENDM

SUBCIR/UCATAL

        XYREG [ Q<0:15>  :  D<0:15>, EXCK, LUPP, LLOW ]

        G1:     NAND2   [ INCKU, EXCK, LUPP  ]
        G2:     NAND2   [ INCKL, EXCK, LLOW  ]

        G3:     CKA     [ CKU, CKBARU, INCKU ]
        G4:     CKA     [ CKL, CKBARL, INCKL ]

        D0:     DT      [ Q< 0>, , CKL, CKBARL, D< 0> ]
        D1:     DT      [ Q< 1>, , CKL, CKBARL, D< 1> ]
        D2:     DT      [ Q< 2>, , CKL, CKBARL, D< 2> ]
        D3:     DT      [ Q< 3>, , CKL, CKBARL, D< 3> ]
        D4:     DT      [ Q< 4>, , CKL, CKBARL, D< 4> ]
        D5:     DT      [ Q< 5>, , CKL, CKBARL, D< 5> ]
        D6:     DT      [ Q< 6>, , CKL, CKBARL, D< 6> ]
        D7:     DT      [ Q< 7>, , CKL, CKBARL, D< 7> ]

        D8:     DT      [ Q< 8>, , CKU, CKBARU, D< 8> ]
        D9:     DT      [ Q< 9>, , CKU, CKBARU, D< 9> ]
        D10:    DT      [ Q<10>, , CKU, CKBARU, D<10> ]
        D11:    DT      [ Q<11>, , CKU, CKBARU, D<11> ]
        D12:    DT      [ Q<12>, , CKU, CKBARU, D<12> ]
        D13:    DT      [ Q<13>, , CKU, CKBARU, D<13> ]
        D14:    DT      [ Q<14>, , CKU, CKBARU, D<14> ]
        D15:    DT      [ Q<15>, , CKU, CKBARU, D<15> ]
ENDM
;*************************************************************************
;********** TRISTATE BUFFER                                    **********
;*************************************************************

DEFINE
SUBCIR/UCATAL
        TRI   [ OUT<0:15> : IN<0:15>, SELECT ]

        G1:     CKB     [ CK, CKBAR, SELECT ]

        D0:     2TM3    [ OUT<  0>, OUT< 1>, CK, CKBAR, IN<  0> ,IN< 1> ]
        D1:     2TM3    [ OUT<  2>, OUT< 3>, CK, CKBAR, IN<  2> ,IN< 3> ]
        D2:     2TM3    [ OUT<  4>, OUT< 5>, CK, CKBAR, IN<  4> ,IN< 5> ]
        D3:     2TM3    [ OUT<  6>, OUT< 7>, CK, CKBAR, IN<  6> ,IN< 7> ]
        D4:     2TM3    [ OUT<  8>, OUT< 9>, CK, CKBAR, IN<  8> ,IN< 9> ]
        D5:     2TM3    [ OUT< 10>, OUT<11>, CK, CKBAR, IN< 10> ,IN<11> ]
        D6:     2TM3    [ OUT< 12>, OUT<13>, CK, CKBAR, IN< 12> ,IN<13> ]
        D7:     2TM3    [ OUT< 14>, OUT<15>, CK, CKBAR, IN< 14> ,IN<15> ]
ENDM

end
```

Appendix 5.
System CLASSIC files

This appendix consists of CLASSIC files from the final system, complete with library, .SCL files describing the test vectors, and a fuller set of results for the test examined in chapter 5.

```
options
        techlib=cla5000
        userlib=RASLIB
        arraysize=4408
        connect/nets/count
        iolist
        timing/alpha/NETS
        units=10ps
        summ
        netlist
;libdata
;         calls/techlib
;         calls/userlib

;~~~~~~~~~~start of circuit~~~~~~~~~~~~

circuit
;IO declares
SUPPLIES/+VDD,-VSS
INPUTS/  DATAEX<0:7>
INPUTS/  XCLOCK, XRESET, XBG, XMEMR, XBACK

OUTPUTS/ PIXEL<0:31>
OUTPUTS/ XBR, XBREQ, XRDR, XS1, XS9, XS10
OUTPUTS/ XINST<0:5>

;general declares

DECLARE/ OI<0:31>
declare/ qx1<0:15>,qy1<0:15>
DECLARE/ DATAIN<0:7>, DATAINA<0:7>
DECLARE/ DATABAR<0:7>
declare/ CI<0:14>
DECLARE/ SIBAR<0:15>
declare/ BUSA<0:15>, BUSB<0:15>, QABAR<0:15>, QBBAR<0:15>
declare/ QSHBAR<0:15>,    XYBUSA<0:15>, XYBUSB<0:15>
declare/ BUSAa<0:15>,BUSAb<0:15>,BUSAc<0:15>,BUSAd<0:15>
declare/ BUSA11<0:15>, BUSA12<0:15>, BUSA13<0:15>, BUSA14<0:15>
declare/ BUSA21<0:15>, BUSA22<0:15>, BUSA23<0:15>, BUSA24<0:15>
declare/ BUSBa<0:15>,BUSBb<0:15>
declare/ XYBUSA1<0:15>,XYBUSA2<0:15>
declare/ XYBUSB1<0:15>,XYBUSB2<0:15>
DECLARE/ INST<0:5>,NINST<0:5>

;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
;output cells
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

;pixel bus
        op0:    (1,1000)        ops     [ pixel< 0>, ,oi< 0>,    oi< 0>,+,qx1< 0> ]
        op1:    (3,1000)        ops     [ pixel< 1>, ,oi< 1>,    oi< 1>,+,qx1< 1> ]
        op2:    (5,1000)        ops     [ pixel< 2>, ,oi< 2>,    oi< 2>,+,qx1< 2> ]
        op3:    (10,1000)       ops     [ pixel< 3>, ,oi< 3>,    oi< 3>,+,qx1< 3> ]
        op4:    (12,1000)       ops     [ pixel< 4>, ,oi< 4>,    oi< 4>,+,qx1< 4> ]
        op5:    (13,1000)       ops     [ pixel< 5>, ,oi< 5>,    oi< 5>,+,qx1< 5> ]
        op6:    (14,1000)       ops     [ pixel< 6>, ,oi< 6>,    oi< 6>,+,qx1< 6> ]
        op7:    (16,1000)       ops     [ pixel< 7>, ,oi< 7>,    oi< 7>,+,qx1< 7> ]
        op8:    (17,1000)       ops     [ pixel< 8>, ,oi< 8>,    oi< 8>,+,qx1< 8> ]
        op9:    (19,1000)       ops     [ pixel< 9>, ,oi< 9>,    oi< 9>,+,qx1< 9> ]
        op10:   (20,1000)       ops     [ pixel<10>, ,oi<10>,    oi<10>,+,qx1<10> ]
        op11:   (24,1000)       ops     [ pixel<11>, ,oi<11>,    oi<11>,+,qx1<11> ]
        op12:   (26,1000)       ops     [ pixel<12>, ,oi<12>,    oi<12>,+,qx1<12> ]
        op13:   (28,1000)       ops     [ pixel<13>, ,oi<13>,    oi<13>,+,qx1<13> ]
        op14:   (2000,1)        ops     [ pixel<14>, ,oi<14>,    oi<14>,+,qx1<14> ]
        op15:   (2000,3)        ops     [ pixel<15>, ,oi<15>,    oi<15>,+,qx1<15> ]

        op16:   (1,2000)        ops     [ pixel<16>, ,oi<16>,    oi<16>,+,qy1< 0> ]
        op17:   (1000,28)       ops     [ pixel<17>, ,oi<17>,    oi<17>,+,qy1< 1> ]
        op18:   (1000,26)       ops     [ pixel<18>, ,oi<18>,    oi<18>,+,qy1< 2> ]
        op19:   (1000,24)       ops     [ pixel<19>, ,oi<19>,    oi<19>,+,qy1< 3> ]
        op20:   (1000,20)       ops     [ pixel<20>, ,oi<20>,    oi<20>,+,qy1< 4> ]
        op21:   (1000,19)       ops     [ pixel<21>, ,oi<21>,    oi<21>,+,qy1< 5> ]
        op22:   (1000,17)       ops     [ pixel<22>, ,oi<22>,    oi<22>,+,qy1< 6> ]
        op23:   (1000,16)       ops     [ pixel<23>, ,oi<23>,    oi<23>,+,qy1< 7> ]
        op24:   (1000,15)       ops     [ pixel<24>, ,oi<24>,    oi<24>,+,qy1< 8> ]
        op25:   (1000,13)       ops     [ pixel<25>, ,oi<25>,    oi<25>,+,qy1< 9> ]
        op26:   (1000,12)       ops     [ pixel<26>, ,oi<26>,    oi<26>,+,qy1<10> ]
        op27:   (1000,10)       ops     [ pixel<27>, ,oi<27>,    oi<27>,+,qy1<11> ]
        op28:   (1000,9)        ops     [ pixel<28>, ,oi<28>,    oi<28>,+,qy1<12> ]
        op29:   (1000,5)        ops     [ pixel<29>, ,oi<29>,    oi<29>,+,qy1<13> ]
```

```
        op30:    (1000,3)          ops      [ pixel<30>, ,oi<30>,   oi<30>,+,qy1<14> ]
        op31:    (1000,1)          ops      [ pixel<31>, ,oi<31>,   oi<31>,+,qy1<15> ]

;br
        op32:    (2000,5)          ops      [ xbr, ,br , br,+,nbr ] ;external active low
;breq
        op33:    (2000,16)         ops      [ xbreq, ,breq , breq,+,nbreq ] ;EXT LOW
;rdr
        op34:    (2000,10)         ops      [ xrdr, , , rdr,+,+ ] ;external active low
;s1
        op35:    (2000,12)         ops      [ xs1, ,s1,s1,+,ns1 ]
;s9
        op36:    (2000,13)         ops      [ xs9, ,s9,s9,+,ns9 ]
;s10
        op37:    (2000,14)         ops      [ xs10, ,s10,s10,+,ns10 ]

; EXTRA TEST POINTS
        op38:    (20,2000)         ops      [ xinst<0>, ,tst0,tst0,+,inst0 ]
        op39:    (24,2000)         ops      [ xinst<1>, ,tst1,tst1,+,inst1 ]
        op40:    (26,2000)         ops      [ xinst<2>, ,tst2,tst2,+,inst2 ]
        op41:    (28,2000)         ops      [ xinst<3>, ,tst3,tst3,+,inst3 ]
        op42:    (2000,28)         ops      [ xinst<4>, ,tst4,tst4,+,inst4 ]
        op43:    (2000,26)         ops      [ xinst<5>, ,tst5,tst5,+,inst5 ]

;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
;~~~      INPUTS                        ~~~~~~~
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
;


;~~~~~~~~~~~ CLOCK IN, (M)-TYPE DRIVE
        CK0:     (3,2000)          IPS      [ CLOCKU, XCLOCK ]
        CK1:     (37,35 )          INV4     [CLOCK1, CLOCKU ]
        CK2:     (37,36 )          INV4     [CLOCK2, CLOCKU ]
        CK3:     PARAL    [CLOCK, CLOCK1,CLOCK2]


;~~~~~ DATA

        I0:      (5,2000)          IPS      [ DATAINA< 0>, DATAEX< 0> ]
        I1:      (9,2000)          IPS      [ DATAINA< 1>, DATAEX< 1> ]
        I2:      (10,2000)         IPS      [ DATAINA< 2>, DATAEX< 2> ]
        I3:      (12,2000)         IPS      [ DATAINA< 3>, DATAEX< 3> ]
        I4:      (13,2000)         IPS      [ DATAINA< 4>, DATAEX< 4> ]
        I5:      (15,2000)         IPS      [ DATAINA< 5>, DATAEX< 5> ]
        I6:      (16,2000)         IPS      [ DATAINA< 6>, DATAEX< 6> ]
        I7:      (17,2000)         IPS      [ DATAINA< 7>, DATAEX< 7> ]

        ILCK:    (37,37) clkb    [ LCK,LCKB,,     MEMRA ,+]
        IL0:     (39,37) df      [ DATAIN< 0>,,  LCK,LCKB,          DATAINA< 0> ]
        IL1:     (41,37) df      [ DATAIN< 1>,,  LCK,LCKB,          DATAINA< 1> ]
        IL2:     (43,37) df      [ DATAIN< 2>,,  LCK,LCKB,          DATAINA< 2> ]
        IL3:     (45,37) df      [ DATAIN< 3>,,  LCK,LCKB,          DATAINA< 3> ]
        IL4:     (47,37) df      [ DATAIN< 4>,,  LCK,LCKB,          DATAINA< 4> ]
        IL5:     (49,37) df      [ DATAIN< 5>,,  LCK,LCKB,          DATAINA< 5> ]
        IL6:     (51,37) df      [ DATAIN< 6>,,  LCK,LCKB,          DATAINA< 6> ]
        IL7:     (53,37) df      [ DATAIN< 7>,,  LCK,LCKB,          DATAINA< 7> ]

        IV0:     (39,36) INV4    [ DATABAR< 0>, DATAIN< 0> ]
        IV1:     (41,36) INV4    [ DATABAR< 1>, DATAIN< 1> ]
        IV2:     (43,36) INV4    [ DATABAR< 2>, DATAIN< 2> ]
        IV3:     (45,36) INV4    [ DATABAR< 3>, DATAIN< 3> ]
        IV4:     (47,36) INV4    [ DATABAR< 4>, DATAIN< 4> ]
        IV5:     (49,36) INV4    [ DATABAR< 5>, DATAIN< 5> ]
        IV6:     (51,36) INV4    [ DATABAR< 6>, DATAIN< 6> ]
        IV7:     (53,36) INV4    [ DATABAR< 7>, DATAIN< 7> ]
;~~~~~~ control inputs
;reset
        I8:      (2000,24)         IPS      [ RESET, XRESET ]

;bg
        I9:      (2000,19)         IPS      [ nBG,  XBG ];internal active high
        IV8:     (115,38)          INV4     [ BG,   nBG ]
;back
        I10:     (2000,17)         IPS      [ nBACK,      XBACK ];internal active high
        IV9:     (115,37)          INV4     [ BACK, nBACK ]
;memr pos edge catch
        I11:     (19,2000)         IPS      [ MEMRA,        XMEMR ]
```

```
        IMR1:   (55,38) clka-   [ MEMCK,MEMCKB,,MEMRA ,+] ;async edge catch
        IMR2:   (55,35) dfrs2   [ MINT,,         MEMCK,MEMCKB,   +,nmemru, + ]

        IMR3:   (57,38) clka-   [ SCK,  SCKB,    ,CLOCK ,+] ;hold for one cycle
        IMR4:   (57,35) dfrs2   [ MEMRU,NMEMRU, SCKB,SCK,       MINT,   +,      reset ]
; note set memr in state 0 to rset mint

        IV10:   (59,35) INV4    [ NMEMR,        MEMRU ]
        IV11:   (59,36) INV4    [ MEMR, NMEMRU ]


;-------------
;counter
;-------------
        cc1:    (81,1)  counter [ c16, clock, div ]

;-------------------------------------------------
;   optional compare cct
;-------------------------------------------------
        occ1:   (39,1)  NOTSUM  [ INTEQX<0:15>,  QX1BAR<0:15>,QX2BAR<0:15> ]

        occ2:   (71,2)  NAND4   [ INTEQX1,  INTEQX<0:3> ]
        occ3:   (71,1)  NAND4   [ INTEQX2,  INTEQX<4:7> ]
        occ4:   (73,1)  NAND4   [ INTEQX3,  INTEQX<8:11> ]
        occ5:   (73,2)  NAND4   [ INTEQX4,  INTEQX<12:15> ]

        occ6:   (75,2)  NOR4    [ XEQ,  INTEQX1,INTEQX2,INTEQX3,INTEQX4 ]


        occ7:   (39,3)  NOTSUM  [ INTEQY<0:15>,  QY1BAR<0:15>,QY2BAR<0:15> ]

        occ8:   (71,4)  NAND4   [ INTEQY1,  INTEQY<0:3> ]
        occ9:   (71,3)  NAND4   [ INTEQY2,  INTEQY<4:7> ]
        occ10:  (73,3)  NAND4   [ INTEQY3,  INTEQY<8:11> ]
        occ11:  (73,4)  NAND4   [ INTEQY4,  INTEQY<12:15> ]

        occ12:  (75,4)  NOR4    [ YEQ,  INTEQY1,INTEQY2,INTEQY3,INTEQY4 ]

        OCC16:  (75,3)  NOR2    [ NXEQ, NENX,   XEQ ]
        OCC17:  (76,3)  NOR2    [ NYEQ, NENY,   YEQ ]

        occ13:  (77,3)  NOR2    [ EQU,  NYEQ,NXEQ ]
        occ14:  (77,2)  INV4    [ NEQ,  EQU ]
        occ15:  (77,1)  INV4    [ EQ,   NEQ ]

;----------------------------------------
;     SPECIAL MUX FOR DIVISION
;----------------------------------------

; normally DIV = 0 and XRA and ASHA(+) are selected
        M1:     (37,29) 4TM     [ NRA, NASHA,  MCK,MCKBAR,  SLIBAR,NXRA,  SLI ,+ ]
        M3:     (39,29) 2INV    [ RA,   ASHA,   NRA,    NASHA ]
        M2:     (37,30) clka-   [ MCK, MCKBAR, ,DIV ,+]



;-------------------------------------------------------------------------
;-----  adder/subtracter  ------------------------------------------------
;-------------------------------------------------------------------------

;-------------------------------------------------------------------------
;--          Two SUMBLs invert inputs to the adder below if          -----
;--          SUBA1 (for ANI inputs) or SUBA2 (for BNI inputs) is true -----
;-------------------------------------------------------------------------

        BN1:    (37,13) SUMBL   [ AII<0:15>, BUSA1<0:15>, SUBA1, SUBA1, SUBA1,$
                 SUBA1, SUBA1, SUBA1,$
                SUBA1, SUBA1, SUBA1, SUBA1,$
                 SUBA1, SUBA1,$
                 SUBA1, SUBA1, SUBA1, SUBA1  ]

        BN2:    (71,13) SUMBL   [ BII<0:15>, BUSA2<0:15>, SUBA2, SUBA2, SUBA2,$
                 SUBA2, SUBA2, SUBA2,$
                SUBA2, SUBA2, SUBA2, SUBA2,$
                 SUBA2, SUBA2,$
```

```
                SUBA2, SUBA2, SUBA2, SUBA2  ]

        GN1:    (70,14) NOR2    [ CINBAR, SUBA1, SUBA2 ]
        GN2:    (69,13) INV4    [ CIN, CINBAR ]

;-------------------------------------------------------------------------
;-----              adder from here onwards                          ------
;-------------------------------------------------------------------------
;
; carry generate/propagate block

        B1:     (45,15) PGBL    [ PI<0:15>, GIBAR<0:15>, aii<0:15>, bii<0:15>  ]

; first level of carry lookahead

        B2:     (45,19) BCLA    [  CI<0:2>,   PS1, GSB1,  PI<0:3>,   GIBAR<0:3>,   CIN
        B3:     (59,19) BCLA    [  CI<4:6>,   PS2, GSB2,  PI<4:7>,   GIBAR<4:7>,   CI<3>
        B4:     (37,21) BCLA    [  CI<8:10>,  PS3, GSB3,  PI<8:11>,  GIBAR<8:11>,  CI<7>
        B5:     (51,21) BCLA    [  CI<12:14>, PS4, GSB4,  PI<12:15>, GIBAR<12:15>, CI<11>

; second level of carry lookahead

        B6:     (37,23) BCLA    [CI<3>, CI7, CI11, PS, GSB, $
                          PS1, PS2, PS3, PS4, GSB1, GSB2, GSB3, GSB4, CIN ]
        B6X1:   (51,23) 2INV    [NCI7,NCI11,    CI7,CI11]
        B6X2:   (52,23) 2INV    [CI<7>,CI<11>,  NCI7,NCI11]


; cout=gstar + cin.pstar
        G1:     (53,24) NAND2   [PSCIN, CIN, PS ]
        G2:     (54,24) NAND2   [COUT, GSB, PSCIN ]

; get SLI = qa<15> + cout
        RG1:    (53,23) 2INV    [ COUTBAR,SLIBAR,  COUT,SLIU ]
        RG2:    (54,23) NAND2   [ SLIU, COUTBAR, QABAR<15> ]
        RG3:    (55,24) INV4    [SLI,   SLIBAR]
; generate ov = cout, nov, neg = coutbar, nneg
        G3:     (55,23) 2INV    [ ovu, negu, negu, cout ]
        G4:     (57,24) INV4    [ NEG, OVU ]
        G5:     (57,23) INV4    [ NNEG, NEGU ]
        G6:     (59,24) INV4    [ OV, NEGU ]
        G7:     (59,23) INV4    [ NOV, OVU ]

; summer block s = p(i) .exor. c(i-1)
;    NB: output INVERTED to give output active low

        B7:     (45,25) SUMBL   [SI<0:15>, PI<0:15>, CIN, CI<0:14> ]

; invert sum
        sinv0:  (45,27) INV4    [SIBAR< 0 >,    SI< 0 > ]
        sinv1:  (45,28) INV4    [SIBAR< 1 >,    SI< 1 > ]
        sinv2:  (47,27) INV4    [SIBAR< 2 >,    SI< 2 > ]
        sinv3:  (47,28) INV4    [SIBAR< 3 >,    SI< 3 > ]
        sinv4:  (49,27) INV4    [SIBAR< 4 >,    SI< 4 > ]
        sinv5:  (49,28) INV4    [SIBAR< 5 >,    SI< 5> ]
        sinv6:  (51,27) INV4    [SIBAR< 6 >,    SI< 6> ]
        sinv7:  (51,28) INV4    [SIBAR< 7 >,    SI< 7 > ]

        sinv8:  (53,27) INV4    [SIBAR< 8 >,    SI< 8 > ]
        sinv9:  (53,28) INV4    [SIBAR< 9 >,    SI< 9 > ]
        sinv10: (55,27) INV4    [SIBAR< 10>,    SI< 10> ]
        sinv11: (55,28) INV4    [SIBAR< 11>,    SI< 11> ]
        sinv12: (57,27) INV4    [SIBAR< 12>,    SI< 12> ]
        sinv13: (57,28) INV4    [SIBAR< 13>,    SI< 13> ]
        sinv14: (59,27) INV4    [SIBAR< 14>,    SI< 14> ]
        sinv15: (59,28) INV4    [SIBAR< 15>,    SI< 15> ]

; generate a ZERO and NZERO

        ZER1:   (43,28) NAND4   [ INTZ1,  SIBAR<0:3> ]
        ZER2:   (43,27) NAND4   [ INTZ2,  SIBAR<4:7> ]
        ZER3:   (41,28) NAND4   [ INTZ3,  SIBAR<8:11> ]
        ZER4:   (41,27) NAND4   [ INTZ4,  SIBAR<12:15> ]

        ZER5:   (40,28) NOR2    [ ZINT1UB,  INTZ3,INTZ4 ]
        ZER5X:  (40,27) NOR2    [ ZINT2UB,  INTZ1,INTZ2 ]
        ZER6:   (39,28) NAND2   [ NZEROU,       ZINT1UB,ZINT2UB ]
        ZER6X:  (39,27) 2INV    [ ,ZEROU,       +,NZEROU ]
        ZER7:   (37,28) INV4    [ ZERO, NZEROU ]
        ZER8:   (37,27) INV4    [ NZERO, ZEROU ]
```

```
;~~~~~~~~~~~~~~ end of subtracter/adder ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
;~~~~ Register set in this section ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~


;~~~ A-register ~~~~~
       R1:     (1,27)  ABREG    [ QABAR<0:15>, BUSA<0:15>, CLOCK, LA ]

;~~~ B-register ~~~~~
       R2:     (1,21)  BREG     [ QBBAR<0:15>, BUSB<0:15>, CLOCK, LB ,FVB]

;~~~ X1 register ~~~~ qx1 <> outputs used for output
       R3:     (1,1)   XY1REG   [ QX1<0:15>,QX1BAR<0:15>, XYBUSA<0:15>, CLOCK, LX1U, LX1L ]

;~~~ X2 register ~~~~
       R4:     (1,11)  XYREG    [ QX2BAR<0:15>, XYBUSB<0:15>, CLOCK, LX2U, LX2L ]

;~~~ Y1 register ~~~~ qy1 <> outputs used for output
       R5:     (1,5)   XY1REG   [ QY1<0:15>,QY1BAR<0:15>, XYBUSA<0:15>, CLOCK, LY1U, LY1L ]

;~~~ Y2 register ~~~~
       R6:     (1,15)  XYREG    [ QY2BAR<0:15>, XYBUSB<0:15>, CLOCK, LY2U, LY2L ]


;~~~ shift register
       R7:     (1,38)  SHR      [ QSHBAR<0:15>, CLOCK, SLI ]

;~~~~~~~~~~~~~~ end of register set ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
;~~~~~~~~~~  steering tristate buffers and associated busnet pseudo-gates ~~
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

;~~~ busA ~~~~
;declare/ BUSAa<0:15>,BUSAb<0:15>,BUSAc<0:15>,BUSAd<0:15>
       BUS0:   TRIBUS[ BUSA<0>, BUSAa<0>, BUSAb<0>, BUSAc<0>, BUSAd<0> ]
       BUS1:   TRIBUS[ BUSA<1>, BUSAa<1>, BUSAb<1>, BUSAc<1>, BUSAd<1> ]
       BUS2:   TRIBUS[ BUSA<2>, BUSAa<2>, BUSAb<2>, BUSAc<2>, BUSAd<2> ]
       BUS3:   TRIBUS[ BUSA<3>, BUSAa<3>, BUSAb<3>, BUSAc<3>, BUSAd<3> ]
       BUS4:   TRIBUS[ BUSA<4>, BUSAa<4>, BUSAb<4>, BUSAc<4>, BUSAd<4> ]
       BUS5:   TRIBUS[ BUSA<5>, BUSAa<5>, BUSAb<5>, BUSAc<5>, BUSAd<5> ]
       BUS6:   TRIBUS[ BUSA<6>, BUSAa<6>, BUSAb<6>, BUSAc<6>, BUSAd<6> ]
       BUS7:   TRIBUS[ BUSA<7>, BUSAa<7>, BUSAb<7>, BUSAc<7>, BUSAd<7> ]
       BUS8:   TRIBUS[ BUSA<8>, BUSAa<8>, BUSAb<8>, BUSAc<8>, BUSAd<8> ]
       BUS9:   TRIBUS[ BUSA<9>, BUSAa<9>, BUSAb<9>, BUSAc<9>, BUSAd<9> ]
       BUS10:  TRIBUS[ BUSA<10>, BUSAa<10>, BUSAb<10>, BUSAc<10>, BUSAd<10> ]
       BUS11:  TRIBUS[ BUSA<11>, BUSAa<11>, BUSAb<11>, BUSAc<11>, BUSAd<11> ]
       BUS12:  TRIBUS[ BUSA<12>, BUSAa<12>, BUSAb<12>, BUSAc<12>, BUSAd<12> ]
       BUS13:  TRIBUS[ BUSA<13>, BUSAa<13>, BUSAb<13>, BUSAc<13>, BUSAd<13> ]
       BUS14:  TRIBUS[ BUSA<14>, BUSAa<14>, BUSAb<14>, BUSAc<14>, BUSAd<14> ]
       BUS15:  TRIBUS[ BUSA<15>, BUSAa<15>, BUSAb<15>, BUSAc<15>, BUSAd<15> ]

;~~~ A(shifted) to A ~~~~
       T1:     (1,33)  TRI      [ BUSAa<0:15>, +, QABAR<0:14>, ASHA ]
;~~~ SI to A ~~~~
       T2:     (23,33) TRI      [ BUSAb<0:15>, SIBAR<0:15>, RA ]
;~~~ B to A ~~~~
       T3:     (1,31)  TRI      [ BUSAc<0:15>, QBBAR<0:15>, ABBA ]
;~~~ SHIFT REGISTER TO A ~~~~
       T4:     (23,31) TRI      [ BUSAd<0:15>, QSHBAR<0:15>, SRA ]

;~~~ bus A1 ~~~~
;declare/ BUSA11<0:15>, BUSA12<0:15>, BUSA13<0:15>, BUSA14<0:15>
       BUS16:  TRIBUS[ BUSA1<0>, BUSA11<0>, BUSA12<0>, BUSA13<0>, BUSA14<0> ]
       BUS17:  TRIBUS[ BUSA1<1>, BUSA11<1>, BUSA12<1>, BUSA13<1>, BUSA14<1> ]
       BUS18:  TRIBUS[ BUSA1<2>, BUSA11<2>, BUSA12<2>, BUSA13<2>, BUSA14<2> ]
       BUS19:  TRIBUS[ BUSA1<3>, BUSA11<3>, BUSA12<3>, BUSA13<3>, BUSA14<3> ]
       BUS20:  TRIBUS[ BUSA1<4>, BUSA11<4>, BUSA12<4>, BUSA13<4>, BUSA14<4> ]
       BUS21:  TRIBUS[ BUSA1<5>, BUSA11<5>, BUSA12<5>, BUSA13<5>, BUSA14<5> ]
       BUS22:  TRIBUS[ BUSA1<6>, BUSA11<6>, BUSA12<6>, BUSA13<6>, BUSA14<6> ]
       BUS23:  TRIBUS[ BUSA1<7>, BUSA11<7>, BUSA12<7>, BUSA13<7>, BUSA14<7> ]
       BUS24:  TRIBUS[ BUSA1<8>, BUSA11<8>, BUSA12<8>, BUSA13<8>, BUSA14<8> ]
       BUS25:  TRIBUS[ BUSA1<9>, BUSA11<9>, BUSA12<9>, BUSA13<9>, BUSA14<9> ]
       BUS26:  TRIBUS[ BUSA1<10>, BUSA11<10>, BUSA12<10>, BUSA13<10>, BUSA14<10> ]
       BUS27:  TRIBUS[ BUSA1<11>, BUSA11<11>, BUSA12<11>, BUSA13<11>, BUSA14<11> ]
```

```
        BUS28:  TRIBUS[ BUSA1<12>, BUSA11<12>, BUSA12<12>, BUSA13<12>, BUSA14<12> ]
        BUS29:  TRIBUS[ BUSA1<13>, BUSA11<13>, BUSA12<13>, BUSA13<13>, BUSA14<13> ]
        BUS30:  TRIBUS[ BUSA1<14>, BUSA11<14>, BUSA12<14>, BUSA13<14>, BUSA14<14> ]
        BUS31:  TRIBUS[ BUSA1<15>, BUSA11<15>, BUSA12<15>, BUSA13<15>, BUSA14<15> ]

;~~~ A(shifted) to A1 ~~~~
           T5:     (49,11) TRI     [ BUSA11<0:15>, +, QABAR<0:14>, div ]
;~~~ A to A1 ~~~~
           T6:     (49,9)  TRI     [ BUSA12<0:15>, QABAR<0:15>, AA1 ]
;~~~ X1 to A1 ~~~~
           T7:     (49,7)  TRI     [ BUSA13<0:15>, QX1BAR<0:15>, X1A1 ]
;~~~ Y1 TO A1 ~~~~
           T8:     (49,5)  TRI     [ BUSA14<0:15>, QY1BAR<0:15>, Y1A1 ]

;~~~ bus A2 ~~~~
;declare/ BUSA21<0:15>, BUSA22<0:15>, BUSA23<0:15>, BUSA24<0:15>
        BUS32:  TRIBUS[ BUSA2<0>, BUSA21<0>, BUSA22<0>, BUSA23<0>, BUSA24<0> ]
        BUS33:  TRIBUS[ BUSA2<1>, BUSA21<1>, BUSA22<1>, BUSA23<1>, BUSA24<1> ]
        BUS34:  TRIBUS[ BUSA2<2>, BUSA21<2>, BUSA22<2>, BUSA23<2>, BUSA24<2> ]
        BUS35:  TRIBUS[ BUSA2<3>, BUSA21<3>, BUSA22<3>, BUSA23<3>, BUSA24<3> ]
        BUS36:  TRIBUS[ BUSA2<4>, BUSA21<4>, BUSA22<4>, BUSA23<4>, BUSA24<4> ]
        BUS37:  TRIBUS[ BUSA2<5>, BUSA21<5>, BUSA22<5>, BUSA23<5>, BUSA24<5> ]
        BUS38:  TRIBUS[ BUSA2<6>, BUSA21<6>, BUSA22<6>, BUSA23<6>, BUSA24<6> ]
        BUS39:  TRIBUS[ BUSA2<7>, BUSA21<7>, BUSA22<7>, BUSA23<7>, BUSA24<7> ]
        BUS40:  TRIBUS[ BUSA2<8>, BUSA21<8>, BUSA22<8>, BUSA23<8>, BUSA24<8> ]
        BUS41:  TRIBUS[ BUSA2<9>, BUSA21<9>, BUSA22<9>, BUSA23<9>, BUSA24<9> ]
        BUS42:  TRIBUS[ BUSA2<10>, BUSA21<10>, BUSA22<10>, BUSA23<10>, BUSA24<10> ]
        BUS43:  TRIBUS[ BUSA2<11>, BUSA21<11>, BUSA22<11>, BUSA23<11>, BUSA24<11> ]
        BUS44:  TRIBUS[ BUSA2<12>, BUSA21<12>, BUSA22<12>, BUSA23<12>, BUSA24<12> ]
        BUS45:  TRIBUS[ BUSA2<13>, BUSA21<13>, BUSA22<13>, BUSA23<13>, BUSA24<13> ]
        BUS46:  TRIBUS[ BUSA2<14>, BUSA21<14>, BUSA22<14>, BUSA23<14>, BUSA24<14> ]
        BUS47:  TRIBUS[ BUSA2<15>, BUSA21<15>, BUSA22<15>, BUSA23<15>, BUSA24<15> ]

;~~~ HEX 0001 to A2 ~~~~ note: RXY here replaces IA2 (by equivalence)
           T9:     (75,11) TRI     [ BUSA21<0:15>, -,+, +,+, +,+, +,+, +,+, +,+, +,+, +,+, RXY
;~~~ B to A2 ~~~~
           T10:    (75,9)  TRI     [ BUSA22<0:15>, QBBAR<0:15>, BA2 ]
;~~~ X2 to A2 ~~~~
           T11:    (75,7)  TRI     [ BUSA23<0:15>, QX2BAR<0:15>, X2A2 ]
;~~~ Y2 TO A2 ~~~~
           T12:    (75,5)  TRI     [ BUSA24<0:15>, QY2BAR<0:15>, Y2A2 ]

;~~~ busB ~~~~
;declare/ BUSBa<0:15>,BUSBb<0:15>
        BUS48:  TRIBUS[ BUSB<0>, BUSBa<0>, BUSBb<0>]
        BUS49:  TRIBUS[ BUSB<1>, BUSBa<1>, BUSBb<1>]
        BUS50:  TRIBUS[ BUSB<2>, BUSBa<2>, BUSBb<2>]
        BUS51:  TRIBUS[ BUSB<3>, BUSBa<3>, BUSBb<3>]
        BUS52:  TRIBUS[ BUSB<4>, BUSBa<4>, BUSBb<4>]
        BUS53:  TRIBUS[ BUSB<5>, BUSBa<5>, BUSBb<5>]
        BUS54:  TRIBUS[ BUSB<6>, BUSBa<6>, BUSBb<6>]
        BUS55:  TRIBUS[ BUSB<7>, BUSBa<7>, BUSBb<7>]
        BUS56:  TRIBUS[ BUSB<8>, BUSBa<8>, BUSBb<8>]
        BUS57:  TRIBUS[ BUSB<9>, BUSBa<9>, BUSBb<9>]
        BUS58:  TRIBUS[ BUSB<10>, BUSBa<10>, BUSBb<10>]
        BUS59:  TRIBUS[ BUSB<11>, BUSBa<11>, BUSBb<11>]
        BUS60:  TRIBUS[ BUSB<12>, BUSBa<12>, BUSBb<12>]
        BUS61:  TRIBUS[ BUSB<13>, BUSBa<13>, BUSBb<13>]
        BUS62:  TRIBUS[ BUSB<14>, BUSBa<14>, BUSBb<14>]
        BUS63:  TRIBUS[ BUSB<15>, BUSBa<15>, BUSBb<15>]

;~~~ A to B ~~~~
           T13:    (1,25)  TRI     [ BUSBa<0:15>, QABAR<0:15>, ABBA ]
;~~~ SI to B ~~~~
           T14:    (23,25) TRI     [ BUSBb<0:15>, SIBAR<0:15>, RB ]


;~~~ XYBUS ~~~~
;declare/ XYBUSA1<0:15>,XYBUSA2<0:15>
;declare/ XYBUSB1<0:15>,XYBUSB2<0:15>
        BUS64:  TRIBUS[ XYBUSA<0>, XYBUSA1<0>, XYBUSA2<0>]
        BUS65:  TRIBUS[ XYBUSA<1>, XYBUSA1<1>, XYBUSA2<1>]
        BUS66:  TRIBUS[ XYBUSA<2>, XYBUSA1<2>, XYBUSA2<2>]
        BUS67:  TRIBUS[ XYBUSA<3>, XYBUSA1<3>, XYBUSA2<3>]
        BUS68:  TRIBUS[ XYBUSA<4>, XYBUSA1<4>, XYBUSA2<4>]
        BUS69:  TRIBUS[ XYBUSA<5>, XYBUSA1<5>, XYBUSA2<5>]
        BUS70:  TRIBUS[ XYBUSA<6>, XYBUSA1<6>, XYBUSA2<6>]
        BUS71:  TRIBUS[ XYBUSA<7>, XYBUSA1<7>, XYBUSA2<7>]
        BUS72:  TRIBUS[ XYBUSA<8>, XYBUSA1<8>, XYBUSA2<8>]
```

```
        BUS73:  TRIBUS[ XYBUSA<9>, XYBUSA1<9>, XYBUSA2<9>]
        BUS74:  TRIBUS[ XYBUSA<10>, XYBUSA1<10>, XYBUSA2<10>]
        BUS75:  TRIBUS[ XYBUSA<11>, XYBUSA1<11>, XYBUSA2<11>]
        BUS76:  TRIBUS[ XYBUSA<12>, XYBUSA1<12>, XYBUSA2<12>]
        BUS77:  TRIBUS[ XYBUSA<13>, XYBUSA1<13>, XYBUSA2<13>]
        BUS78:  TRIBUS[ XYBUSA<14>, XYBUSA1<14>, XYBUSA2<14>]
        BUS79:  TRIBUS[ XYBUSA<15>, XYBUSA1<15>, XYBUSA2<15>]

        BUS164: TRIBUS[ XYBUSB<0>, XYBUSB1<0>, XYBUSB2<0>]
        BUS165: TRIBUS[ XYBUSB<1>, XYBUSB1<1>, XYBUSB2<1>]
        BUS166: TRIBUS[ XYBUSB<2>, XYBUSB1<2>, XYBUSB2<2>]
        BUS167: TRIBUS[ XYBUSB<3>, XYBUSB1<3>, XYBUSB2<3>]
        BUS168: TRIBUS[ XYBUSB<4>, XYBUSB1<4>, XYBUSB2<4>]
        BUS169: TRIBUS[ XYBUSB<5>, XYBUSB1<5>, XYBUSB2<5>]
        BUS170: TRIBUS[ XYBUSB<6>, XYBUSB1<6>, XYBUSB2<6>]
        BUS171: TRIBUS[ XYBUSB<7>, XYBUSB1<7>, XYBUSB2<7>]
        BUS172: TRIBUS[ XYBUSB<8>, XYBUSB1<8>, XYBUSB2<8>]
        BUS173: TRIBUS[ XYBUSB<9>, XYBUSB1<9>, XYBUSB2<9>]
        BUS174: TRIBUS[ XYBUSB<10>, XYBUSB1<10>, XYBUSB2<10>]
        BUS175: TRIBUS[ XYBUSB<11>, XYBUSB1<11>, XYBUSB2<11>]
        BUS176: TRIBUS[ XYBUSB<12>, XYBUSB1<12>, XYBUSB2<12>]
        BUS177: TRIBUS[ XYBUSB<13>, XYBUSB1<13>, XYBUSB2<13>]
        BUS178: TRIBUS[ XYBUSB<14>, XYBUSB1<14>, XYBUSB2<14>]
        BUS179: TRIBUS[ XYBUSB<15>, XYBUSB1<15>, XYBUSB2<15>]

;--- DATA to XY ----
        T15:    (1,9)   TRI     [ XYBUSA1<0:15>, DATABAR<0:7>, DATABAR<0:7>, dxy ]
        T115:   (23,9)  TRI     [ XYBUSB1<0:15>, DATABAR<0:7>, DATABAR<0:7>, dxy ]
;--- SI to XY ----
        T16:    (1,19)  TRI     [ XYBUSA2<0:15>, SIBAR<0:15>, RXY ]
        T116:   (23,19) TRI     [ XYBUSB2<0:15>, SIBAR<0:15>, RXY ]

;-------------------------------------------------------------------------------
;---  fsm follows                    -----
;---  $$$$$$$$$$$                    -----
;-------------------------------------------------------------------------------
;
; register set

        FSMCKV: (115,36)        2INV    [ NCLOCK,,CLOCK,+]
        FSMCK:  (113,38)        clkB    [ FCK,  FCKB,   ,NCLOCK ,+]
        FSMO:   (73,35) dfrs2   [ INST<0>, NINST<0>, FCK, FCKB, OUTST0, RESET, + ]
        FS0:    (71,37) drvb    [ INST0,        INST<0> ]
        FS1:    (77,37) drvb    [ NINST0,       NINST<0> ]

        FSM1:   (83,38) dfrs2   [INST<1>, NINST<1>, FCK, FCKB, OUTST1, RESET, + ]
        FS2:    (81,36) drvb    [ INST1,        INST<1> ]
        FS3:    (75,36) drvb    [ NINST1,       NINST<1> ]

        FSM2:   (87,35) dfrs2   [INST<2>, NINST<2>, FCK, FCKB, OUTST2, RESET, + ]
        FS4:    (85,37) drvb    [ INST2,        INST<2> ]
        FS5:    (91,37) drvb    [ NINST2,       NINST<2> ]

        FSM3:   (97,38) dfrs2   [INST<3>, NINST<3>, FCK, FCKB, OUTST3, RESET, + ]
        FS6:    (95,36) drvb    [ INST3,        INST<3> ]
        FS7:    (89,36) drvb    [ NINST3,       NINST<3> ]

        FSM4:   (101,35)        dfrs2   [INST<4>, NINST<4>, FCK, FCKB, OUTST4, RESET, + ]
        FS8:    (99,37) drvb    [ INST4,        INST<4> ]
        FS9:    (105,37)        drvb    [ NINST4,       NINST<4> ]

        FSM5:   (111,38)        dfrs2   [INST<5>, NINST<5>, FCK, FCKB, OUTST5, RESET, + ]
        FS10:   (109,36)        drvb    [ INST5,        INST<5> ]
        FS11:   (103,36)        drvb    [ NINST5,       NINST<5> ]


; next state logic as at 18 aug

;product terms and partial product terms prefixed with P
;sum and partial sum terms prefixed with S
;
;outst0-------------------------------------------------------------
;P1, P2
        OA1:    (60,38) NAND2   [ POA1, MAN,    NENX ]
        OA2:    (61,38) NAND3   [ POA2, ,       BG,     NEQ,    ENY,    + ]
        OA3:    (63,38) NAND2   [ POA3, INSTO,  NINST1 ]
        OA4:    (65,38) NAND4   [ POA4, NINST2, NINST3, INST4,  NINST5 ]
        OA5:    (64,38) NAND2   [ POA5, ENY,    NBG ]
```

```
        OB1:    (59,37) NOR4    [ POB1, POA1,   POA2,   POA3,   POA4 ]
        OB2:    (61,37) NOR3    [ POB2, ,       POA3,   POA4,   POA5,   + ]

;P3, P4
        OA6:    (63,37) NAND2   [ POA6, MAN,    NENY ]
        OA7:    (65,37) NAND3   [ POA7, ,       ENX,    NBG,    NEQ,    + ]
        OA8:    (64,37) NAND2   [ POA8, INSTO,  NINST1 ]
        OA9:    (67,37) NAND4   [ POA9, NINST2, NINST3, INST4,  NINST5 ]
        OA10:   (69,37) NAND2   [ POA10,        NBG,    EQ ]

        OB3:    (67,38) NOR4    [ POB3, POA6,   POA7,   POA8,   POA9 ]
        OB4:    (69,38) NOR4    [ POB4, POA6,   POA8,   POA9,   POA10 ]

;P5, P6
;killed OA11:   ()      NAND4   [ POA11,        MAN,    NENX,   NENY,   NEQ ]
        OA12:   (61,35) NAND2   [ POA12,        INSTO,  NINST1 ]
        OA13:   (61,36) NAND4   [ POA13,        NINST2, NINST3, INST4,  NINST5 ]
        OA14:   (63,36) NAND3   [ POA14,        ,       NMAN,   NENY,   NBG,    + ]

;killed OB5:    ()      NOR3    [ POB5, ,       POA11,  POA12,  POA13,  + ]
        OB6:    (63,35) NOR3    [ POB6, ,       POA12,  POA13,  POA14,  + ]

;P7
        OA15:   (65,36) NAND3   [ POA15,        ,       INSTO,  NINST1, NINST2, + ]
        OA16:   (65,35) NAND2   [ POA16,        NINST3, NINST4 ]

        OB7:    (66,35) NOR2    [ POB7, POA15,  POA16 ]

;P8
        OB8:    (67,36) NAND4   [ POB8, NINSTO, INST1,  INST2,  INST3 ]
        obn8:   (68,35) 2inv    [ npOb8,,       pOb8,+ ]

;P9
        OA17:   (69,36) NAND4   [ POA17,        NENX,   NOV,    NINSTO, INST1 ]
        OA18:   (69,35) NAND4   [ POA18,        INST2,  NINST3, NINST4, INST5 ]

        OB9:    (71,36) NOR2    [ POB9, POA17,  POA18 ]

;P10, P11
        OA19:   (53,34) NAND4   [ POA19,        NINST2, INST3,  NINST4, NINST5 ]
        OA20:   (53,33) NAND2   [ POA20,        NINSTO, INST1 ]
        OA21:   (55,34) NAND4   [ POA21,        MAN,    NINST2, NINST3, INST4 ]

        OB10:   (55,33) NOR2    [ POB10,        POA19,  POA20 ]
        OB11:   (56,33) NOR2    [ POB11,        POA20,  POA21 ]

;p12 new
        0a22:   (57,34) nand4   [ pOa22,        NINSTO, INST1,  INST2,  NINST3 ]
        0a23:   (57,33) nand2   [ pOa23,        NINST4, NINST5 ]

        0b12:   (58,33) nor2    [ pOb12,        pOa22,  pOa23 ]

;SUMMATION
        OC1:    (59,34) NOR4    [ SOC1, POB1,   POB2,   POB3,   POB4 ]
        OC2:    (59,33) NOR3    [ SOC2, ,       POB6,   POB7,   NPOB8 ,+ ];fixed(see kill)
        OC3:    (61,34) NOR4    [ SOC3, pOb12,  POB9,   POB10,  POB11 ];fixed

        OD1:    (61,33) NAND3   [ OUTSTO,       ,       SOC1,   SOC2,   SOC3,   + ]

;outst1^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

;P1 , P2
        1a1:    (115,34)        NAND4   [ P1a1,   NINST2, NINST3, INST4, NINST5 ]
        1a2:    (115,33)        NAND4   [ P1a2,   BG,     NEQ,    INSTO, NINST1 ]
        1a3:    (112,34)        NAND2   [ P1a3,   MAN,    ENX ]

        1b1:    (113,33)        NOR3    [ P1b1,   ,       P1a1,   P1a2,   P1a3,   +]
        1b2:    (113,34)        NOR3    [ P1b2,   ,       P1a1,   P1a2,   MAN,    + ]

;P3 , P4
;scrubbed       1a4:
        1a5:    (111,33)        NAND4   [ P1a5,   NINSTO, INST1,  INST2,  NINST3 ]

        1b3:    (111,34)        NOR2    [ P1b3,   ninst4,   P1a5 ]
;scrubbed       1b4:

;P5 , P6
        1a6:    (110,34)        NAND4   [ P1a6,   ENX,    NOV,    NINSTO, INST1 ]
        1a7:    (110,33)        NAND4   [ P1a7,   INST2,  NINST3, NINST4, INST5 ]
```

```
        1a8:    (107,34)        NAND3   [ P1a8,    ,        OV,     NINSTO, INST1, + ]

        1b5:    (107,33)        NOR2    [ P1b5,    P1a6,    P1a7 ]
        1b6:    (108,33)        NOR2    [ P1b6,    P1a7,    P1a8 ]

;P8 , P7
        1a9:    (106,34)        NAND2   [ P1a9,    NINST5, INST4 ]
        1a10:   (106,33)        NAND4   [ P1a10,   NINSTO, INST1,  NINST2, INST3 ]

        1b7:    (104,34)        NOR2    [ P1b7,    P1a10,   NINST5 ]
        1b8:    (104,33)        NOR2    [ P1b8,    P1a9,    P1a10 ]

;P9 , P10  scrubbed

;P11 , P12
        1a14:   (108,32)        nand2   [ p1a14,         INST4,  nman ]
        1a15:   (108,31)        NAND4   [ P1a15,   NINSTO, INST1,  NINST2, NINST3 ]
        1a16:   (109,31)        NAND3   [ P1a16,   ,       NINST4, INST5,  NZERO, +]

        1b11:   (109,32)        nor2    [ p1b11,   p1a14,          p1a15 ]
        1b12:   (110,32)        NOR2    [ P1b12,   P1a15,   P1a16 ]

;P13 , P14
        1a17:   (109,30)        NAND4   [ P1a17,   ZERO,   NNEG,   NINST4, INST5 ]
        1a18:   (109,29)        NAND4   [ P1a18,   NINSTO, INST1,  NINST2, NINST3 ]
        1a19:   (111,29)        NAND2   [ P1a19,   NINST4, NINST5 ]

        1b13:   (111,30)        NOR2    [ P1b13,   P1a17,   P1a18 ]
        1b14:   (112,29)        NOR2    [ P1b14,   P1a18,   P1a19 ]

;P15 , P16
        1a20:   (114,29)        NAND2   [ P1a20,   INST4,  INST5 ]
        1a21:   (116,29)        NAND4   [ P1a21,   NINSTO, NINST1, INST2,  INST3 ]
        1a22:   (116,30)        NAND3   [ P1a22,   ,       NNEG,   INST4,  NINST5, +]

        1b15:   (113,30)        NOR2    [ P1b15,   P1a20,   P1a21 ]
        1b16:   (113,29)        NOR2    [ P1b16,   P1a21,   P1a22 ]

;SUMMATION
        1C1:    (115,32)        NOR3    [ S1C1, ,   P1b1,   P1b2,   P1b3,    + ]
        1C2:    (115,31)        NOR4    [ S1C2,    P1b5,   P1b6,   P1b7,   P1b8 ]
        1C3:    (114,32)        nor2    [ S1C3,    p1b11,         P1b12 ];fixed
        1C4:    (113,31)        NOR4    [ S1C4,    P1b13,  P1b14,  P1b15,  P1b16 ]

        1D1:    (111,31)        NAND4   [ OUTST1U, S1C1,    S1C2,    S1C3,    S1C4 ]
        1D1X:   (113,32)        2INV    [NOUTST1,,        OUTST1U,+]
        1D1X1:  (111,32)        INV4    [OUTST1,NOUTST1]

;outst2^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

;P1, P2
        2A1:    (104,30)        NAND2   [ P2A1, MAN,    ENX ]
        2A2:    (105,30)        NAND4   [ P2A2, BG,     NEQ,    INSTO,  NINST1 ]
        2A3:    (105,29)        NAND4   [ P2A3, NINST2, NINST3, INST4,  NINST5 ]

        2B1:    (107,30)        NOR3    [ P2B1, ,       P2A1,   P2A2,   P2A3,    + ]
        2B2:    (107,29)        NOR3    [ P2B2, ,       P2A2,   P2A3,   MAN,     + ]

;P3, P4
        2A4:    (103,30)        NAND2   [ P2A4, INST1,  INST4 ];fixed
        2A5:    (103,29)        NAND3   [ P2A5, ,       NINSTO, INST2,  NINST3, +  ];refixe
        2a5x:   (102,30)        nand2   [ p2a5x,        NINST1, NINST5 ]

        2B3:    (102,29)        NOR2    [ P2B3, P2A4,   P2A5 ]
        2B4:    (101,29)        NOR2    [ P2B4, P2A5,   p2a5x ]

;P5, P6
        2A6:    (97,30) NAND2   [ P2A6, ENX,    NOV ]
        2A7:    (98,30) NAND2   [ P2A7, NINSTO, INST1 ]
        2A8:    (97,29) NAND4   [ P2A8, INST2,  NINST3, NINST4, INST5 ]

        2B5:    (99,30) NOR3    [ P2B5, ,       P2A6,   P2A7,   P2A8,    + ]
        2B6:    (99,29) NOR3    [ P2B6, ,       P2A7,   P2A8,   NOV,     + ]

;P7, P8
        2A9:    (95,30) NAND2   [ P2A9, INST4,  INST5 ]
        2A10:   (95,29) NAND4   [ P2A10,NINSTO, INST1,  NINST2, INST3 ]
;sccrubbed      2A11:

        2B7:    (96,30) NOR2    [ P2B7, P2A9,   P2A10 ]
```

```
;scrubbed       2B8:

;P9, P10
;scrubbed       2A12:
       2A13:   (105,32)        NAND2   [ P2A13,NINST0, INST4 ]
       2A14:   (106,32)        NAND2   [ P2A14,NEG,    NZERO ]
       2A15:   (105,31)        NAND4   [ P2A15,NINST1, INST2,  INST3,  NINST5 ]

;scrubbed       2B9:
       2B10:   (103,31)        NOR3    [ P2B10,,       P2A13,  P2A14,  P2A15,  + ]

;P11, P12
       2A16:   (98,34) NAND2   [ P2A16,INST3,  NINST4 ]
       2A17:   (99,34) NAND3   [ P2A17,,       NINST0, NINST1, INST2,  + ]
       2A18:   (99,33) NAND4   [ P2A18,NINST3, INST4,  INST5,  NZERO ]

       2B11:   (97,34) NOR2    [ P2B11,P2A16,  P2A17 ]
       2B12:   (97,33) NOR2    [ P2B12,P2A17,  P2A18 ]

;P13, P14
       2A19:   (93,32) NAND4   [ P2A19,ZERO,   NNEG,   NINST0, INST4 ]
       2A20:   (93,31) NAND4   [ P2A20,NINST1, INST2,  NINST3, INST5 ]
       2A21:   (95,32) NAND2   [ P2A21,NINST4, NINST0 ]

       2B13:   (96,32) NOR2    [ P2B13,P2A19,  P2A20 ]
       2B14:   (96,31) NOR2    [ P2B14,P2A20,  P2A21 ]

;P15
       2A22:   (101,34)        NAND3   [ P2A22,,       MEMR,   NINST0, NINST1, + ]
       2A23:   (101,33)        NAND4   [ P2A23,NINST2, INST3,  INST4,  INST5 ]

       2B15:   (103,33)        NOR2    [ P2B15,P2A22,  P2A23 ]

;SUMMATION
       2C1:    (97,32) NOR4    [ S2C1, P2B1,   P2B2,   P2B3,   P2B4 ]
       2C2:    (97,31) NOR3    [ S2C2, ,       P2B5,   P2B6,   P2B7,   + ];fixed
       2C3:    (99,31) NOR3    [ S2C3, ,       P2B10,  P2B11,  P2B12,  + ];fixed
       2C4:    (99,32) NOR3    [ S2C4, ,       P2B13,  P2B14,  P2B15,  + ]

       2D1:    (101,31)        NAND4   [ OUTST2U,      S2C1,   S2C2,   S2C3,   S2C4 ]
       2D1X:   (103,32)        2INV    [NOUTST2,,      OUTST2U,+]
       2D1X1:  (101,32)        INV4    [OUTST2,NOUTST2]


;OUTST3^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
;P1, P2
       3A1:    (115,28)        NAND4   [ P3A1, MAN,    ENX,    BG,     NEQ ]
       3A2:    (115,27)        NAND4   [ P3A2, INST0,  NINST1, NINST2, NINST5 ]
       3A3:    (113,27)        NAND2   [ P3A3, NINST3, INST4 ]
       3A4:    (113,28)        NAND3   [ P3A4, ,       NINST0, INST1,  INST2,  + ]

       3B1:    (111,28)        NOR3    [ P3B1, ,       P3A1,   P3A2,   P3A3,   + ]
       3B2:    (112,27)        NOR2    [ P3B2, P3A3,   P3A4 ]

;P3, P4
       3A5:    (108,28)        NAND2   [ P3A5, ENX,    NOV ]
       3A6:    (108,27)        NAND2   [ P3A6, NINST0, INST1 ]
       3A7:    (109,28)        NAND4   [ P3A7, INST2,  NINST3, NINST4, INST5 ]
       R3A7:   (109,27)        NAND4   [ PR3A7,        NINST2, INST3,  INST4,  NINST5 ]

       3B3:    (105,28)        NOR3    [ P3B3, ,       P3A5,   P3A6,   P3A7,   + ]
       3B4:    (106,27)        NOR2    [ P3B4, P3A6,   PR3A7 ]

;P5, P6 scrubb p6
       3A8:    (103,28)        NAND4   [ P3A8, NINST0, NINST2, INST3,  NINST4 ]
;scrubbed       3A9:

       3B5:    (104,27)        NOR2    [ P3B5, NINST5, P3A8 ]
;scrubbed       3B6:

;P7, P8
;scrubbed       3A10:
       3A11:   (101,28)        nand4   [p3a11, NINST0, INST1,  NINST2, NINST3 ]
       3A12:   (102,27)        nand2   [p3a12, INST4,  nman ]

;scrubbed       3B7:
       3B8:    (101,27)        nor2    [p3b8,  p3a11,  p3a12 ]

;P9, P10
       3A13:   (115,26)        NAND3   [ P3A13,,ZERO,  NNEG,   INST1,  + ]
```

```
        R3A13:  (115,25)        NAND3   [ PR3A13,,NZERO,         NEG,    NINST1, + ]
        3A14:   (113,26)        NAND4   [ P3A14,NINST2, NINST3, NINST4, INST5 ]
        R3A14:  (113,25)        NAND4   [ PR3A14,INST2, INST3,  INST4,  NINST5 ]

        3B9:    (111,26)        NOR3    [ P3B9, ,       P3A13,  P3A14,  INSTO,  + ]
        3B10:   (111,25)        NOR3    [ P3B10,        ,       INSTO,  PR3A13, PR3A14, +

;P11, P12
        3A15:   (108,25)        NAND2   [ P3A15,INST3,  NINST4 ]
        3A16:   (109,26)        NAND3   [ P3A16,,       NINSTO, NINST1, INST2,  + ]
        3A17:   (109,25)        NAND4   [ P3A17,NZERO,  NINST3, INST4,  INST5 ]

        3B11:   (107,26)        NOR2    [ P3B11,P3A15,  P3A16 ]
        3B12:   (107,25)        NOR2    [ P3B12,P3A16,  P3A17 ]

;P13, P14
        3A18:   (105,25)        NAND4   [ P3A18,NNEG,   INST2,  NINST3, INST5 ]
        3A19:   (105,26)        NAND3   [ P3A19,,       NINSTO, NINST1, INST4,  + ]
        3A20:   (103,25)        NAND3   [ P3A20,,       NINST2, INST3,  NMEMR,  + ]

        3B13:   (101,25)        NOR3    [ P3B13,,       NZERO,  P3A18,  P3A19,  + ]
        3B14:   (102,26)        NOR2    [ P3B14,P3A19,  P3A20 ]

;P15, P16
        3A21:   (101,26)        NAND2   [ P3A21,MEMR,   NINSTO ]
        3A22:   (99,25) NAND4   [ P3A22,NINST1, NINST2, INST3,  NINST5 ]
        3A23:   (99,26) NAND3   [ P3A23,,       NMEMR,  NINSTO, NINST4, + ]

        3B15:   (98,25) NOR2    [ P3B15,P3A21,  P3A22 ]
        3B16:   (98,26) NOR2    [ P3B16,P3A22,  P3A23 ]

;P17
        3A24:   (95,25) NAND3   [ P3A24,,       MEMR,   NINSTO, NINST1, + ]
        3A25:   (95,26) NAND4   [ P3A25,NINST2, NINST3, INST4,  INST5 ]

        3B17:   (97,26) NOR2    [ P3B17,P3A24,  P3A25 ]

;SUMMATION

        3C1:    (99,28) NOR4    [ S3C1, P3B1,   P3B2,   P3B3,   P3B4 ]
        3C2:    (99,27) NOR3    [ S3C2, ,       P3B5,   P3B9,   p3b8,   + ]   ;altered
        3C3:    (97,28) NOR2    [ S3C3, P3B10,  P3B11 ]  ;altered
        3C4:    (97,27) NOR4    [ S3C4, P3B12,  P3B13,  P3B14,  P3B15 ]
        3C5:    (98,28) NOR2    [ S3C5, P3B16,  P3B17 ]

        3D1:    (95,28) NAND2   [ S3D1, S3C1,   S3C2 ]
        3D2:    (95,27) NAND3   [ S3D2, OUTST3, S3C3,   S3C4,   S3C5,   S3E1 ]

        3E1:    (96,28) NOR2    [ S3E1, S3D1,   S3D2 ]


;OUTST4^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

;P1, P2
        4A1:    (81,34) NAND2   [ P4A1, NBG,    ENY ]
        4A2:    (81,33) NAND2   [ P4A2, INSTO,  NINST1 ]
        4A3:    (79,34) NAND4   [ P4A3, NINST2, NINST3, INST4,  NINST5 ]
        4A4:    (79,33) NAND4   [ P4A4, MAN,    ENX,    BG,     NEQ ]

        4B1:    (77,34) NOR3    [ P4B1, ,       P4A1,   P4A2,   P4A3,   + ]
        4B2:    (77,33) NOR3    [ P4B2, ,       P4A2,   P4A3,   P4A4,   + ]

;P3, P4
        4A5:    (75,34) NAND3   [ P4A5, ,       ENX,    NEQ,    NINST1, + ]
        4A6:    (75,33) NAND4   [ P4A6, NINST2, NINST3, INST4,  NINST5 ]
        4A7:    (73,34) NAND4   [ P4A7, MAN,    NENY,   NBG,    INSTO ]
        4A8:    (73,33) NAND2   [ P4A8, EQ,     NINST1 ]

        4B3:    (71,34) NOR3    [ P4B3, ,       P4A5,   P4A6,   P4A7,   + ]
        4B4:    (71,33) NOR3    [ P4B4, ,       P4A6,   P4A7,   P4A8,   + ]

;P5, P6
;killed 4A9:    NAND3   [ P4A9, ,       MAN,    NENX,   NEQ,    + ]
        4A10:   (73,29) NAND3   [ P4A10,        ,       NENY,   INSTO,  NINST1, + ]
        4A11:   (75,29) NAND4   [ P4A11,        NINST2, NINST3, INST4,  NINST5 ]
        4A12:   (77,29) NAND2   [ P4A12,        NMAN,   NBG ]

;killed 4B5:    NOR3    [ P4B5, ,       P4A9,   P4A10,  P4A11,  + ]
        4B6:    (75,30) NOR3    [ P4B6, ,       P4A10,  P4A11,  P4A12,  + ]
```

```
;P7, P8
        4A13:   (67,29) NAND4   [ P4A13,        NINST1, NINST2, NINST3, NINST4 ]
        R4A13:  (69,29) NAND4   [ PR4A13,       INST1,  INST2,  INST3,  INST4 ]
        4A14:   (71,29) NAND2   [ P4A14,        NINSTO, NENY ]

        4B7:    (69,30) NOR2    [ P4B7, NINSTO, P4A13 ]
        4B8:    (71,30) NOR2    [ P4B8, PR4A13, P4A14 ]

;P9
        4B9:    (65,29) NAND4   [ P4B9, NINSTO, INST2,  INST3,  NINST4 ]
        4b9x:   (66,30) 2inv    [ np4b9,,       p4b9,+]

;P9, P10
        4A15:   (62,29) NAND2   [ P4A15,        ENX,    NOV ]
        4A16:   (63,29) NAND4   [ P4A16,        NINSTO, INST1,  INST2,  NINST3 ]
        4A17:   (61,29) NAND2   [ P4A17,        NINST4, INST5 ]

        4B10:   (61,30) NOR3    [ P4B10,        ,       P4A15,  P4A16,  P4A17,  + ]
        4B11:   (63,30) NOR3    [ P4B11,        ,       P4A16,  P4A17,  NOV,    + ]

;P12, P13
;killed 4A18:   ()       NAND2  [ P4A19,        NINSTO, INST1 ]
        4A19:   (79,28) NAND2   [P4A19, NINSTO, INST1 ]
;killed 4A20:   ()       NAND4  [ P4A20,        INST2,  NINST3, NINST4, INST5 ]
        R4A20:  (79,27) NAND4   [ PR4A20,       NINST2, INST3,  INST4,  NINST5 ]

;killed 4B12:   ()       NOR3   [ P4B12,        ,       P4A18,  P4A19,  P4A20,  + ]
        4B13:   (80,28) NOR2    [ P4B13,        PR4A20, P4A19 ]

;P14, P15
        4A21:   (77,28) NAND2   [ P4A21,        NINST4, INST5 ]
        4A22:   (77,27) NAND4   [ P4A22,        NINSTO, INST1,  NINST2, INST3 ]
;scrubbed       4A23:

        4B14:   (78,28) NOR2    [ P4B14,        P4A21,  P4A22 ]
;scrubbed       4B15:

;P16
        4A24:   (75,28) NAND2   [ P4A24,        NINSTO, INST1 ];enx,eny removed
        4A25:   (75,27) NAND4   [ P4A25,        NINST2, INST3,  NINST4, NINST5 ]

        4B16:   (76,28) NOR2    [ P4B16,        P4A24,  P4A25 ]

;P17, P18
        4A26:   (73,28) NAND2   [ P4A26,        ninst3, MAN ];enx replaced ($error site$)
        4A27:   (73,27) NAND4   [ P4A27,        NINSTO, INST1,  NINST2, INST4 ]
;scrubbed       4A28:

        4B17:   (74,28) NOR2    [ P4B17,        P4A26,  P4A27 ]
;scrubbed       4B18:

;P19, P20
        4A29:   (70,28) NAND2   [ P4A29,        NZERO,  NINSTO ]
        4A30:   (71,28) NAND4   [ P4A30,        INST1,  NINST2, NINST3, NINST4 ]
        R4A30:  (71,27) NAND4   [ PR4A30,       NINST1, INST2,  INST3,  INST4 ]
        4A31:   (70,27) NAND2   [ P4A31,        NEG,    NINST5 ]

        4B19:   (67,28) NOR3    [ P4B19,        ,       NINST5, P4A29,  P4A30,  + ]
        4B20:   (67,27) NOR3    [ P4B20,        ,       P4A29,  PR4A30, P4A31,  + ]

;P21, P22
        4A32:   (69,34) NAND4   [ P4A32,        NZERO,  NNEG,   NINSTO, INST5 ]
        4A33:   (69,33) NAND4   [ P4A33,        NINST1, INST2,  NINST3, INST4 ]
        4A34:   (68,34) NAND2   [ P4A34,        NINST5, NINSTO ]

        4B21:   (67,34) NOR2    [ P4B21,        P4A32,  P4A33 ]
        4B22:   (67,33) NOR2    [ P4B22,        P4A33,  P4A34 ]

;P23, P24
        4A35:   (65,32) NAND4   [ P4A35,        MEMR,   NINST3, NINST4, INST5 ]
        4A36:   (64,32) NAND2   [ P4A36,        NINSTO, NINST1 ]
        4A37:   (65,31) NAND3   [ P4A37,        ,       NMEMR,  NINST2, INST4,  + ]

        4B23:   (63,32) NOR2    [ P4B23,        P4A35,  P4A36 ]
        4B24:   (63,31) NOR2    [ P4B24,        P4A36,  P4A37 ]

;P25, P26
        4A38:   (68,32) NAND2   [ P4A38,        INST4,  NINST5 ]
        4A39:   (69,32) NAND4   [ P4A39,        MEMR,   NINSTO, NINST1, NINST2 ]
```

```
        4A40:   (69,31) NAND3   [ P4A40,         ,         INST3,  NINST4, INST5,  + ]

        4B25:   (67,32) NOR2    [ P4B25,        P4A38,  P4A39 ]
        4B26:   (67,31) NOR2    [ P4B26,        P4A39,  P4A40 ]
;p27 new
        4a41:   (71,31) nand2   [ p4a41,        NINSTO, INST1 ]
        4a42:   (71,32) nand4   [ p4a42,        INST2,  NINST3, NINST4, NINST5 ]

        4b27:   (72,31) nor2    [ p4b27,        p4a41,  p4a42 ]
;SUMMATION

        4C1:    (82,32) NOR2    [ S4C1, P4B1,   P4B2 ]
        4C2:    (82,31) NOR2    [ S4C2, P4B3,   P4B4 ];fixed (see kill)
        4C3:    (79,32) NOR4    [ S4C3, P4B6,   P4B7,   P4B8,   NP4B9 ]
        4C4:    (79,31) NOR3    [ S4C4, ,       P4B10,  P4B11,  P4B13,  + ];fixed(see kill
        4C5:    (81,32) nor2    [ S4C5, P4B14,  p4b27 ]
        4C6:    (81,31) NOR2    [ S4C6, P4B16,  P4B17 ] ;p4b18 scrubbed
        4C7:    (77,32) NOR4    [ S4C7, P4B19,  P4B20,  P4B21,  P4B22 ]
        4C8:    (77,31) NOR4    [ S4C8, P4B23,  P4B24,  P4B25,  P4B26 ]

        4D1:    (75,32) NAND4   [ S4D1, S4C1,   S4C2,   S4C3,   S4C4 ]
        4D2:    (75,31) NAND4   [ S4D2, S4C5,   S4C6,   S4C7,   S4C8 ]

        4E1:    (73,32) NOR2    [ S4E1, S4D1,   S4D2 ]
        4F1:    (73,31) 2INV    [ OUTST4,,      S4E1,+ ]


;OUTST5^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
;P1, P2
        5A1:    (95,34) NAND4   [ P5A1, MAN,    INY,    NENX,   ENY ]
        5A2:    (93,34) NAND4   [ P5A2, NINST2, NINST3, INST4,  NINST5 ]
        5A3:    (91,34) NAND4   [ P5A3, BG,     NEQ,    INSTO,  NINST1 ]
        5A4:    (91,33) NAND3   [ P5A4, ,       MAN,    ENX,    INX,    + ]

        5B1:    (93,33) NOR3    [ P5B1, ,       P5A1,   P5A2,   P5A3,   + ]
        5B2:    (95,33) NOR3    [ P5B2, ,       P5A2,   P5A3,   P5A4,   + ]

;P3, P4
        5A5:    (89,33) NAND4   [ P5A5, NEQ,    BG,     INST4,  NMAN ]
        5A6:    (89,34) NAND4   [ P5A6, INSTO,  NINST1, NINST2, NINST3 ]
        R5A6:   (87,34) NAND4   [ PR5A6,        NINSTO, INST1,  INST2,  INST3 ]
        5A7:    (85,34) NAND3   [ P5A7, ,       ENY,    INY,    INST4,  + ]

        5B3:    (87,33) NOR3    [ P5B3, ,       INST5,  P5A5,   P5A6,   + ]
        5B4:    (86,33) NOR2    [ P5B4, PR5A6,  P5A7 ]

;P5, P6
        5A8:    (89,29) NAND2   [ P5A8, INY,    INST4 ]
        5A9:    (91,29) NAND4   [ P5A9, NINSTO, INST1,  INST2,  NINST3 ]
        5A10:   (90,29) NAND2   [ P5A10,        NENX,   ENY ]
        5A11:   (93,29) NAND4   [ P5A11,        INY,    NOV,    NINST4, INST5 ]

        5B5:    (90,30) NOR2    [ P5B5, P5A8,   P5A9 ]
        5B6:    (91,30) NOR3    [ P5B6, ,       P5A9,   P5A10,  P5A11,  + ]

;P7, P8
        5A12:   (81,29) NAND3   [ P5A12,        ,       ENX,    INX,    NOV,    + ]
        5A13:   (83,29) NAND2   [ P5A13,        NINSTO, INST1 ]
        5A14:   (85,29) NAND4   [ P5A14,        INST2,  NINST3, NINST4, INST5 ]
        5A15:   (87,29) NAND2   [ P5A15,        OV,     INX ]

        5B7:    (83,30) NOR3    [ P5B7, ,       P5A12,  P5A13,  P5A14,  + ]
        5B8:    (87,30) NOR3    [ P5B8, ,       P5A13,  P5A14,  P5A15,  + ]

;P9
        5A16:   (78,29) NAND2   [ P5A16,        NINSTO, INST1 ]
;inserted ninst2
        5A17:   (79,29) NAND4   [ P5A17,        NINST2, NINST3, NINST4, NINST5 ]

        5B9:    (79,30) NOR2    [ P5B9, P5A16,  P5A17 ]

;P10, P11 (p11 scrubbed )
        5A18:   (83,34) NAND3   [ P5A18,        ,       C16,    INST4,  NINST5, + ]
        5A19:   (83,33) NAND4   [ P5A19,        NINSTO, INST1,  NINST2, INST3 ]

        5B10:   (85,33) NOR2    [ P5B10,        P5A18,  P5A19 ]

;P12 scrubbed

;P13, P14 scrubbed
```

```
;P15, P16 (p15 scrubbed )
        5a28:    (92,28) nand2   [ p5a28,       INST4,  nman ]
        5A29:    (93,28) NAND4   [ P5A29,       NINST0, INST1,  NINST2, NINST3 ]
        5A30:    (93,27) NAND4   [ P5A30,       NINST4, INST5,  NZERO,  NEG ]

        5b15:    (91,28) nor2    [ p5b15,       p5a28,  p5a29 ]
        5B16:    (91,27) NOR2    [ P5B16,       P5A29,  P5A30 ]

;P17, P18
        5A31:    (88,28) NAND2   [ P5A31,       INST4,  INST5 ]
        5A32:    (89,28) NAND4   [ P5A32,       NINST0, NINST1, INST2,  INST3 ]
        5A33:    (89,27) NAND3   [ P5A33,       ,       NZERO,  INST4,  NINST5, + ]

        5B17:    (87,28) NOR2    [ P5B17,       P5A31,  P5A32 ]
        5B18:    (87,27) NOR2    [ P5B18,       P5A32,  P5A33 ]

;P19, P20
        5A34:    (85,28) NAND4   [ P5A34,       ZERO,   NNEG,   INST4,  INST5 ]
        5A35:    (85,27) NAND4   [ P5A35,       NINST0, NINST1, INST2,  NINST3]
        5A36:    (84,28) NAND2   [ P5A36,       MEMR,   NINST5 ]

        5B19:    (83,28) NOR2    [ P5B19,       P5A34,  P5A35 ]
        5B20:    (83,27) NOR2    [ P5B20,       P5A35,  P5A36 ]

;P21, P22
        5A37:    (93,25) NAND2   [ P5A37,       NINST3, NINST4 ]
        5A38:    (93,26) NAND4   [ P5A38,       NMEMR,  NINST0, NINST1, INST5 ]
        5A39:    (94,25) NAND2   [ P5A39,       NINST2, INST3 ]

        5B21:    (92,26) NOR2    [ P5B21,       P5A37,  P5A38 ]
        5B22:    (92,25) NOR2    [ P5B22,       P5A38,  P5A39 ]

;P23, P24
        5A40:    (89,26) NAND3   [ P5A40,       ,       MEMR,   INST3,  NINST5, + ]
        5A41:    (89,25) NAND3   [ P5A41,       ,       NINST0, NINST1, NINST2, + ]
        5A42:    (87,25) NAND4   [ P5A42,       NMEMR,  NINST3, INST4,  INST5 ]

        5B23:    (88,26) NOR2    [ P5B23,       P5A40,  P5A41 ]
        5B24:    (87,26) NOR2    [ P5B24,       P5A41,  P5A42 ]

;P25, P26
        5A43:    (85,26) NAND3   [ P5A43,       ,       MEMR,   INST4,  NINST5, + ]
        5A44:    (85,25) NAND4   [ P5A44,       NINST0, NINST1, NINST2, NINST3 ]
        5A45:    (83,25) NAND3   [ P5A45,       ,       BACK,   NINST4, NINST5, + ]

        5B25:    (83,26) NOR2    [ P5B25,       P5A43,  P5A44 ]
        5B26:    (84,26) NOR2    [ P5B26,       P5A44,  P5A45 ]

;SUMMATION
        5C1:     (92,32) NOR2    [ P5C1, P5B1,  P5B2 ]
        5C2:     (91,31) NOR3    [ P5C2, ,      P5B3,   P5B4,   P5B5,   + ]
        5C3:     (89,31) NOR4    [ P5C3, P5B6,  P5B7,   P5B8,   P5B9 ]
        5C4:     (91,32) 2inv    [ P5C4, ,      P5B10,  + ] ;p5b11/12/13 scrubbed
;       5C5:     scrubbed with p5b14/15
        5C6:     (89,32) NOR4    [ P5C6, p5b15, P5B16,  P5B17,  P5B18 ];fixed
        5C7:     (87,32) NOR4    [ P5C7, P5B19, P5B20,  P5B21,  P5B22 ]
        5C8:     (87,31) NOR4    [ P5C8, P5B23, P5B24,  P5B25,  P5B26 ]

        5D1:     (85,31) NAND4   [ P5D1, P5C1,  P5C2,   P5C3,   P5C4 ]
;altered
        5D2:     (85,32) NAND3   [ P5D2, outst5, P5C6,  P5C7,   P5C8,   p5e1 ]

        5E1:     (84,32) NOR2    [ P5E1, P5D1,  P5D2 ]

;the end of new nsl



;output forming logic ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

;BR ^^^^^

        AA1:     (99,2)  NAND3   [PAA1,  ,      INST0,  NINST1, NINST2, + ]
        AA2:     (101,2) NAND3   [PAA2,  ,      NINST3, INST4,  NINST5, + ]

        AB1:     (102,1) NOR2    [BRu,   PAA1,  PAA2 ]
        ac1:     (99,1)  inv4    [nbr,   bru ]
```

```
;RXY  and  IA2 ^^^^
        BA1:    (113,23)        NAND2   [PBA1,  INSTO,  NINST1 ]
        BA2:    (115,24)        NAND3   [PBA2,  NPBA3,  NINST2, NINST3, NINST4, PBA3 ]
        BA3:    (115,23)        NAND4   [PBA3,  NINSTO, INST1,  INST2,  INST3 ]
        BA4:    (113,24)        NAND3   [PBA4,  ,       NINSTO, INST1,  INST2,  + ]
        BA5:    (114,23)        NAND2   [PBA5,  NINST3, INST4 ]

        BB1:    (112,23)        NOR2    [PBB1,  PBA1,   PBA2 ]
        BB2:    (111,23)        NOR2    [PBB2,  PBA4,   PBA5 ]

        BC1:    (111,24)        NOR3    [SBC1,  ,       PBB1,   NPBA3,  PBB2,   + ]
        BD1:    (109,24)        INV4    [RXY,   SBC1 ]

;FIVEB
        CA1:    (107,24)        NAND3   [PCA1,  ,       NINSTO, INST1,  INST2,  + ]
        CA2:    (107,23)        NAND3   [PCA2,  ,       NINST3, NINST4, NINST5, + ]

        CB1:    (106,24)        NOR2    [FIVEBU,        PCA1,   PCA2 ]
        CB1X:   (105,24)        2INV    [NFIVEB,,       FIVEBU,+]
        CB1X1:  (105,23)        INV4    [FIVEB,NFIVEB]

;SRA ^^^^^^^
;       DA1:    ()      NAND3   [PDA1,  ,       NINSTO, INST1,  NINST2, + ]
;       DA2:    (105,34)        NAND3   [PDA2,  ,       INST3,  INST4,  INST5,  + ]
;
;       DB1:    (105,33)        NOR2    [SRAU,  PDA1,   PDA2 ]
;       DB1X:   2INV    [NSRA,, SRAU,+]
;       DB1X1:  INV4    [SRA,NSRA]


;DIV and ASHA1
        EA1:    (103,24)        NAND3   [PEA1,  ,       NINSTO, INST1,  NINST2, + ]
        EA2:    (103,23)        NAND3   [PEA2,  NDIV,   INST3,  INST4,  NINST5, DIVU ]

        EB1:    (102,24)        NOR2    [DIVU,  PEA1,   PEA2 ]
        EC1:    (101,23)        INV4    [DIV,   NDIV ]
;clearc was here

;BA or AB
        GA1:    (100,24)        NAND2   [PGA1,  NINSTO, INST1 ]
        GA2:    (99,23) NAND4   [PGA2,  NINST2, NINST3, INST4,  INST5 ]

        GB1:    (99,24) NOR2    [ABBAU, PGA1,   PGA2 ]

        GC1:    (98,24) 2INV    [,      NABBA,  +,      ABBAU ]
        gd1:    (97,23) inv4    [abba,  nabba]
;BA2
;       HA1:    ()      NAND2   [PHA1,  NINSTO, INST1 ]
;       HA2:    ()      NAND3   [PHA2,  BA2,    NINST3, NINST4, INST5,  SHC1 ]
;       HA3:    ()      NAND2   [PHA3,  NINSTO, INST1 ]
;       HA4:    ()      NAND4   [PHA4,  NINST2, INST3,  INST4,  NINST5 ]
;
;       HB1:    ()      NOR2    [PHB1,  PHA1,   PHA2 ]
;       HB2:    ()      NOR2    [PHB2,  PHA3,   PHA4 ]
;
;       HC1:    ()      NOR2    [SHC1,  PHB1,   PHB2 ]

;AA1
        IA1:    (96,24) NAND2   [PIA1,  NINSTO, INST1 ]
        IA2:    (95,23) NAND3   [PIA2,  ,       NINST3, NINST4, INST5,  + ]

        IB1:    (94,24) NOR2    [AA1u,  PIA1,   PIA2 ]
        ic1:    (93,24) 2inv    [naa1,  ,       aa1u,+ ]
        id1:    (93,23) inv4    [aa1,   naa1]

;RB
;       JA1:    ()      NAND2   [PJA1,  NINSTO, INST1 ]
;       JA2:    ()      NAND4   [PJA2,  INST2,  NINST3, NINST4, INST5 ]
;       JA3:    ()      NAND3   [PJA3,  RB,     NINSTO, NINST1, INST2,  SJC1 ]
;       JA4:    ()      NAND2   [PJA4,  INST3,  INST4 ]
;
;       JB1:    ()      NOR2    [PJB1,  PJA1,   PJA2 ]
;       JB2:    ()      NOR2    [PJB2,  PJA3,   PJA4 ]
;
;       JC1:    ()      NOR2    [SJC1,  PJB1,   PJB2 ]

;Y2A2
        KA1:    (92,23) NAND2   [PKA1,  NINSTO, NINST1 ]
        KA2:    (91,24) NAND3   [PKA2,  ,       INST2,  INST3,  INST4,  + ]
```

```
        KB1:    (90,23) NOR2    [Y2A2U, PKA1,   PKA2 ]
        KB1X:   (89,23) 2INV    [NY2A2,,        Y2A2U,+]
        KB1X1:  (89,24) INV4    [Y2A2,NY2A2]

;Y1A1
;       LA1:    ()      NAND2   [PLA1,  INSTO,  NINST1 ]
;       LA2:    ()      NAND3   [PLA2,  ,       NINST2, NINST3, NINST4, + ]
;       LA3:    ()      NAND2   [PLA3,  INST1,  NINST4 ]
;       LA4:    ()      NAND3   [PLA4,  Y1A1,   NINSTO, INST2,  INST3,  SLC1 ]
;       LA5:    ()      NAND2   [PLA5,  NINST1, INST4 ]
;
;       LB1:    ()      NOR2    [PLB1,  PLA1,   PLA2 ]
;       LB2:    ()      NOR2    [PLB2,  PLA3,   PLA4 ]
;       LB3:    ()      NOR2    [PLB3,  PLA4,   PLA5 ]
;
;       LC1:    ()      NOR3    [SLC1,  ,       PLB1,   PLB2,   PLB3,   + ]

;SINY   and  SINX
        MA1:    (116,22)        NAND2   [PMA1,  INST4,  INST5 ]
        MA2:    (115,21)        NAND4   [PMA2,  NINSTO, NINST1, INST2,  INST3 ]
        MA3:    (115,22)        NAND2   [PMA3,  NINST4, NINST5 ]

        MB1:    (113,22)        NOR2    [SINYu, PMA1,   PMA2 ]
        MB2:    (114,22)        NOR2    [SINXu, PMA2,   PMA3 ]
        mc1:    (113,21)        2inv    [nsinx, nsiny,  sinxu,  sinyu]
        md1:    (111,21)        inv4    [sinx,  nsinx]
        md2:    (111,22)        inv4    [siny,  nsiny]

;SENY
        NA1:    (108,22)        NAND2   [PNA1,  INST3,  NINST4 ]
        NA2:    (109,22)        NAND4   [PNA2,  NINSTO, INST1,  NINST2, NINST5 ]
        NA3:    (108,21)        NAND2   [PNA3,  NINST3, INST4 ]
        NA4:    (109,21)        NAND4   [PNA4,  NINSTO, NINST1, INST2,  INST5 ]

        NB1:    (107,22)        NOR2    [PNB1,  PNA1,   PNA2 ]
        NB2:    (107,21)        NOR2    [PNB2,  PNA2,   PNA3 ]
        NB3:    (106,21)        NOR2    [PNB3,  PNA1,   PNA4 ]

        NC1:    (105,22)        NOR3    [SNC1,  ,       PNB1,   PNB2,   PNB3,   + ]
        nc1x:   (103,22)        inv4    [seny,  snc1]

;SENX
        OA1:    (100,21)        NAND2   [POA1,  NINSTO, INST1 ]
        OA2:    (101,22)        NAND3   [POA2,  ,       NINST2, NINST4, NINST5, + ]
        OA3:    (101,21)        NAND4   [POA3,  NINST2, NINST3, INST4,  INST5 ]

        OB1:    (99,22) NOR2    [POB1,  POA1,   POA2 ]
        OB2:    (99,21) NOR2    [POB2,  POA1,   POA3 ]

        OC1:    (98,21) NOR2    [SOC1,  POB1,   POB2 ]
        oc1x:   (97,22) inv4    [senx,  soc1]

;SMAN
        PA1:    (94,22) NAND2   [PPA1,  NINSTO, NINST4 ]
        PA2:    (95,22) NAND3   [PPA2,  ,       INST1,  NINST2, NINST5, + ]
        PA3:    (95,21) NAND4   [PPA3,  NINST1, INST2,  INST3,  INST5 ]

        PB1:    (93,22) NOR2    [PPB1,  PPA1,   PPA2 ]
        PB2:    (93,21) NOR2    [PPB2,  PPA1,   PPA3 ]

        PC1:    (92,21) NOR2    [SPC1,  PPB1,   PPB2 ]
        pc1x:   (91,22) inv4    [sman,  spc1]
;LB
        QA1:    (88,21) NAND2   [PQA1,  NINSTO, INST2 ]
        QA2:    (89,22) NAND4   [PQA2,  INST1,  NINST3, NINST4, inst5 ];refixed
        QA3:    (89,21) NAND3   [PQA3,  ,       NINST1, INST3,  INST4,  + ]
        QA4:    (87,22) NAND4   [PQA4,  ninst3, NINST2, INST4,  INST5 ];refixed
        QA5:    (87,21) NAND2   [PQA5,  NINSTO, INST1 ]

        QB1:    (86,21) NOR2    [PQB1,  PQA1,   PQA2 ]
        QB2:    (86,22) NOR2    [PQB2,  PQA1,   PQA3 ]
        QB3:    (85,22) NOR2    [PQB3,  PQA4,   PQA5 ]

        QC1:    (83,21) NOR3    [SQC1,  ,       PQB1,   PQB2,   PQB3,   + ]
        qc1x:   (83,22) inv4    [lb,    sqc1]

;SUBA1
        RA1:    (114,20)        NAND2   [PRA1,  NINSTO, INST1 ]
        RA2:    (115,20)        NAND4   [PRA2,  NINST2, NINST3, NINST4, INST5 ]
```

```
        RA3:    (113,20)        NAND2   [PRA3,  INST4,  INST5 ]
        RA4:    (115,19)        NAND4   [PRA4,  NINSTO, NINST1, INST2,  INST3 ]
        RA5:    (113,19)        NAND2   [PRA5,  NINST5, NINST4 ]

        RB1:    (112,20)        NOR2    [PRB1,  PRA1,   PRA2 ]
        RB2:    (111,20)        NOR2    [PRB2,  PRA3,   PRA4 ]
        RB3:    (111,19)        NOR2    [PRB3,  PRA4,   PRA5 ]

        RC1:    (109,20)        NOR3    [SRC1,  ,       PRB1,   PRB2,   PRB3,   + ]
        RD1:    (109,19)        INV4    [SUBA1, SRC1 ]

;SUBA2
        SA1:    (106,20)        NAND2   [PSA1,  INSTO,  NINST1 ]
        SA2:    (107,20)        NAND4   [PSA2,  NINST2, NINST3, NINST4, NINST5 ]
        SA3:    (105,20)        NAND2   [PSA3,  NINSTO, INST1 ]
        SA4:    (107,19)        NAND3   [PSA4,  ,       INST2,  INST3,  NINST5, + ]
        SA5:    (104,20)        NAND2   [PSA5,  INST2,  NINST3 ]
        SA6:    (105,19)        NAND4   [PSA6,  NINSTO, INST1,  INST4,  NINST5 ]
        SA7:    (103,20)        NAND2   [PSA7,  NINST2, INST3 ]
        SA8:    (102,20)        NAND2   [PSA8,  INST3,  NINST5 ]
        SA9:    (103,19)        NAND4   [PSA9,  NINSTO, NINST1, INST2,  INST4 ]
        SA10:   (101,20)        NAND2   [PSA10, NINST3, INST5 ]

        SB1:    (100,20)        NOR2    [PSB1,  PSA1,   PSA2 ]
        SB2:    (99,20) NOR2    [PSB2,  PSA3,   PSA4 ]
        SB3:    (99,19) NOR2    [PSB3,  PSA5,   PSA6 ]
        SB4:    (98,20) NOR2    [PSB4,  PSA6,   PSA7 ]
        SB5:    (97,20) NOR2    [PSB5,  PSA8,   PSA9 ]
        SB6:    (97,19) NOR2    [PSB6,  PSA9,   PSA10 ]

        SC1:    (95,20) NOR3    [SSC1,  ,       PSB1,   PSB2,   PSB3,   + ]
        SC2:    (95,19) NOR3    [SSC2,  NSUBA2, PSB4,   PSB5,   PSB6,   SUBA2U ]

        SD1:    (94,20) NAND2   [SUBA2U,        SSC1,   SSC2 ]
        SE1:    (93,19) INV4    [SUBA2, NSUBA2 ]

;LA
        TA1:    (91,20) NAND3   [PTA1,  ,       NINST2, INST4,  INST5,  + ]
        TA2:    (91,19) NAND2   [PTA2,  NINSTO, INST1 ]
        TA3:    (89,20) NAND3   [PTA3,  ,       NINST2, INST3,  INST4,  + ]
        TA4:    (89,19) NAND3   [PTA4,  ,       NINSTO, NINST1, INST2,  + ]
        TA5:    (87,20) NAND3   [PTA5,  ,       INST3,  NINST4, NINST5, + ]
        TA6:    (87,19) NAND3   [PTA6,  ,       NINST3, INST4,  INST5,  + ]

        TB1:    (86,20) NOR2    [PTB1,  PTA1,   PTA2 ]
        TB2:    (85,20) NOR2    [PTB2,  PTA2,   PTA3 ]
        TB3:    (86,19) NOR2    [PTB3,  PTA4,   PTA5 ]
        TB4:    (85,19) NOR2    [PTB4,  PTA4,   PTA6 ]

        TC1:    (83,20) NOR4    [STC1,  PTB1,   PTB2,   PTB3,   PTB4 ]
        tc1x:   (83,19) inv4    [1a,    stc1]

;RA  or X2A2  get terms from above
        UC1:    (86,18) NOR2    [NXRAU, PTB3,   PTB4 ]
        UD1:    (85,18) 2INV    [X2A2,  XRAU,   NXRAU,  NXRAU ]
        UE1:    (85,17) INV4    [NXRA,XRAU ]

;X1A1
        VA1:    (115,18)        NAND4   [PVA1,  NINSTO, INST1,  INST2,  INST4 ]
        VA2:    (115,17)        NAND3   [PVA2,  ,       NINSTO, NINST1, INST2,  + ]
        VA3:    (113,18)        NAND3   [PVA3,  ,       INST3,  NINST4, NINST5, + ]
        VA4:    (113,17)        NAND3   [PVA4,  NPVA1,  NINST3, INST4,  INST5,  PVA1 ]

        VB1:    (112,18)        NOR2    [PVB1,  PVA2,   PVA3 ]
        VB2:    (112,17)        NOR2    [PVB2,  PVA2,   PVA4 ]

        VC1:    (109,18)        NOR3    [SVC1,  ,       NPVA1,  PVB1,   PVB2,   +]
        vc1x:   (109,17)        inv4    [x1a1,  svc1]

;CLEARFF
        WA1:    (87,23) NAND2   [PWA1,  NINSTO, NINST1 ]
        WA2:    (87,24) NAND4   [PWA2,  INST2,  NINST3, INST4,  INST5 ]

        WB1:    (86,23) NOR2    [CLEARFFu,      PWA1,   PWA2 ]
        wc1:    (85,23) 2inv    [nclff, ,       clearffu,+ ]
        wd1:    (85,24) inv4    [clearff,nclff]

;LY2L and LY2U
        XA1:    (108,18)        NAND2   [PXA1,  INST4,  NINST5 ]
        XA2:    (107,17)        NAND4   [PXA2,  NINSTO, NINST1, INST2,  NINST3 ]
```

```
        XA3:    (107,18)        NAND2   [PXA3,  NINST4, INST5 ]

        XB1:    (106,18)        NOR2    [LY2UU, PXA1,   PXA2 ]
        XB2:    (106,17)        NOR2    [LY2LU, PXA2,   PXA3 ]

        XB1X:   (105,18)        2INV    [ NLY2U,        NLY2L,  LY2UU,  LY2LU ]
        XB2X:   (103,18)        INV4    [ LY2U, NLY2U ]
        XB3X:   (103,17)        INV4    [ LY2L, NLY2L ]


;LX2L and LX2U
        YA1:    (100,18)        NAND2   [PYA1,  NINSTO, NINST1 ]
        YA2:    (101,18)        NAND4   [PYA2,  NINST2, INST3,  INST4,  INST5 ]
        YA3:    (101,17)        NAND4   [PYA3,  INST2,  NINST3, NINST4, NINST5 ]

        YB1:    (99,18) NOR2    [LX2LU, PYA1,   PYA2 ]
        YB2:    (99,17) NOR2    [LX2UU, PYA1,   PYA3 ]

        YB1X:   (98,18) 2INV    [ NLX2U,        NLX2L,  LX2UU,  LX2LU ]
        YB2X:   (97,17) INV4    [ LX2U, NLX2U ]
        YB3X:   (95,17) INV4    [ LX2L, NLX2L ]

;LY1U
        ZA1:    (114,16)        NAND2   [PZA1,  NINST3, NINST4 ]
        ZA2:    (115,16)        NAND3   [PZA2,  ,       INSTO,  NINST1, NINST2, + ]
        ZA3:    (115,15)        NAND3   [PZA3,  ,       NINSTO, INST1,  INST2,  + ]
        ZA4:    (113,16)        NAND2   [PZA4,  INST3,  NINST4 ]
        ZA5:    (113,15)        NAND2   [PZA5,  NINSTO, NINST1 ]
        ZA6:    (111,16)        NAND4   [PZA6,  NINST2, INST3,  INST4,  NINST5 ]

        ZB1:    (110,16)        NOR2    [PZB1,  PZA1,   PZA2 ]
        ZB2:    (109,16)        NOR2    [PZB2,  PZA3,   PZA4 ]
        ZB3:    (109,15)        NOR2    [PZB3,  PZA5,   PZA6 ]

        ZC1:    (107,16)        NOR3    [SZC1,  ,       PZB1,   PZB2,   PZB3,   +]
        ZC1X:   (107,15)        INV4    [LY1U,  SZC1 ]

;LY1L
        AAA1:   (93,17) NAND2   [PAAA1, NINSTO, NINST1 ]
        AAA2:   (93,18) NAND4   [PAAA2, NINST2, INST3,  NINST4, INST5 ]

        AAB1:   (92,18) NOR2    [PAAB1, PAAA1,  PAAA2 ]

        AAC1:   (91,17) NOR3    [SAAC1, ,       PZB1,   PZB2,   PAAB1,  +]
        AAC1X:  (89,18) INV4    [LY1L,  SAAC1 ]

;LX1U
        abA1:   (105,16)        NAND4   [PabA1, NINSTO, INST1,  INST2,  INST4 ]
        abA2:   (104,16)        NAND2   [PabA2, NINSTO, NINST1 ]
        abA3:   (105,15)        NAND4   [PabA3, NINST2, INST3,  NINST4, NINST5 ]

        abB1:   (104,15)        NOR2    [PabB1, PabA2,  PabA3 ]

        abB2:   (103,15)        2INV    [NPabA1,        ,       PabA1,  +]

        abC1:   (102,15)        NOR2    [SabC1, NPabA1, PabB1 ]
        abC1X:  (101,16)        INV4    [LX1U,  SABC1 ]

;LX1L   two terms swiped from LX1U
        acA1:   (103,14)        NAND4   [PacA1, NINST2, NINST3, INST4,  INST5 ]

        acB1:   (105,14)        NOR2    [PacB1, PabA2,  PacA1 ]

        acB2:   (106,14)        2INV    [NabA1, ,       PabA1,  +]

        acC1:   (106,13)        NOR2    [SacC1, NabA1,  PacB1 ]
        ACC1X:  (103,13)        INV4    [LX1L,  SACC1 ]

;BREQ
        adA1:   (111,3) NAND2   [PadA1, NINSTO, NINST1 ]
        adA2:   (111,4) NAND4   [PadA2, NINST2, NINST3, NINST4, NINST5 ]

        adB1:   (112,3) NOR2    [BREQu, PadA1,  PadA2 ]
        adC1:   (113,3) INV4    [NBREQ, BREQu ]

;RDR
        AEA1:   (103,2) NAND3   [PAEA1, ,       NINSTO, NINST1, NINST2, + ]
        AEA2:   (103,1) NAND4   [PAEA2, NINSTO, NINST1, INST2,  NINST3 ]

        AEB1:   (105,2) NOR2    [PAEB1, PAEA1,  NINST3 ]
```

```
        AEB2:   (106,2) NOR2    [PAEB2, PAEA1,  NINST4 ]
        AEB3:   (107,2) NOR2    [PAEB3, PAEA1,  NINST5 ]

        AEB4:   (108,2) NOR2    [PAEB4, PAEA2,  INST4 ]
        AEB5:   (109,2) NOR2    [PAEB5, PAEA2,  INST5 ]

        AEC1:   (105,1) NOR3    [SAEC1, NRDRU,  PAEB1,  PAEB2,  PAEB3,  RDRU ]
        AEC2:   (107,1) NOR2    [SAEC2, PAEB4,  PAEB5 ]

        AED1:   (108,1) NAND2   [RDRU,  SAEC1,  SAEC2 ]
        AEE1:   (109,1) INV4    [RDR,   NRDRU ]

;s1
        afA1:   (111,2) NAND4   [PafA1, nINST0, nINST1, nINST2, nINST3 ]
        afA2:   (111,1) NAND2   [PafA2, nINST4, INST5 ]

        afB1:   (112,1) NOR2    [S1u,   PafA1,  PafA2 ]
        afC1:   (113,1) INV4    [NS1,   S1u ]
;s9
        agA1:   (113,2) NAND4   [PagA1, nINST0, nINST1, INST2,  nINST3 ]
        agA2:   (115,2) NAND2   [PagA2, nINST4, INST5 ]

        agB1:   (116,2) NOR2    [S9u,   PagA1,  PagA2 ]
        agC1:   (115,1) INV4    [NS9,   S9u ]

;s10
        ahA1:   (113,4) NAND4   [PahA1, nINST0, nINST1, INST2,  nINST3 ]
        ahA2:   (115,4) NAND2   [PahA2, INST4,  nINST5 ]

        ahB1:   (116,4) NOR2    [S10u,  PahA1,  PahA2 ]
        ahC1:   (115,3) INV4    [NS10,  S10u ]

;extra bits and clearb,fiveb sync

        ex0:    (46,29) clka    [exckb,exck,    clock]

        ex1:    (53,29) df      [,      sinxb,  exck,exckb,     sinx ]
        ex2:    (53,31) rsnand  [,ninxu,        clf,    sinxb]
        exbu1:  (53,32) inv4    [inx,ninxu]


        ex3:    (55,29) df      [,      sinyb,  exck,exckb,     siny ]
        ex4:    (55,31) rsnand  [,ninyu,        clf,    sinyb]
        exbu3:  (55,32) inv4    [iny,ninyu]

        ex5:    (47,29) df      [,      senxb,  exck,exckb,     senx ]
        ex6:    (47,31) rsnand  [enxu,nenxu,    clf,    senxb]
        exbu5:  (47,32) inv4    [enx,nenxu]
        exbu6:  (47,33) inv4    [nenx,enxu]

        ex7:    (49,29) df      [,      senyb,  exck,exckb,     seny ]
        ex8:    (49,31) rsnand  [enyu,nenyu,    clf,    senyb]
        exbu7:  (49,32) inv4    [eny,nenyu]
        exbu8:  (49,33) inv4    [neny,enyu]

        ex9:    (51,29) df      [,      smanb,  exck,exckb,     sman ]
        ex10:   (51,31) rsnand  [manu,nmanu,    clf,    smanb]
        exbu9:  (51,32) inv4    [man,nmanu]
        exbu10: (51,33) inv4    [nman,manu]

        ex11:   (43,29) df      [,      clf,    exck,exckb,     clearff]
        ex12:   (57,29) df      [,      NFVB,   exck,exckb,     FIVEB]
        EXBU12: (57,31) INV4    [FVB,   NFVB]

; new logic to ensure busses are defined all the time :8 sept 86

;sra
        nsra0:  (81,24) nor3    [srau,  ,ra,asha,abba,  +]
        nsra1:  (81,23) 2inv    [nsra,, srau,+]
        nsra2:  (80,24) inv4    [sra,   nsra]

;dxy
        ndxy:   (93,16) 2inv    [dxy,,  rxy,+]

;rb
        nrb:    (91,16) 2inv    [rb,,   abba,+]

;y1a1   (DIV IS ASHA1)
        ny1a10: (89,16) nor3    [y1a1u, ,       div,    aa1,    x1a1,   +]
        ny1a11: (89,15) 2inv    [ny1a1,,        y1a1u,+]
```

```
        ny1a12: (87,16) inv4    [y1a1,  ny1a1]

;ba2  ( rxy is IA2 )
        nba20:  (85,16) nor3    [ba2u,  ,       x2a2,y2a2,rxy,  +]
        nba21:  (85,15) 2inv    [nba2,  ,ba2u,+]
        nba22:  (83,16) inv4    [ba2,   nba2]


;WHEW!!!!

END
```

```
options
        techlib=cla5000


        arraysize=6000
;       connect/nets/count
;       iolist
;       timing/alpha/all
        units=10ps
        summ
;       netlist


define

;@~~~~~~

subcir/ucatal
        drvb    [o160 : i4]

        dv1:    (2,1)           inv4    [i11,i4]

        dv2:    (0,0)           inv8    [i21,i11]
        dv3:    (4,0)           inv8    [i22,i11]

        dv4:    paral   [o160,i21,i22]
endm
;@ ~~~~~~~~~~~~~~
subcir/ucatal
COUNTER [SXT : EXCK,EN ]
declare/c<0:2>,d<0:3>,q<0:3>
        cc0:    (10,3)          2inv    [ , enb,        +,en ]
        cc0x:   (14,3)          inv4    [ eni, enb ]

        cc1:    (2,0)           dfrs2   [ q<0>,, ck,ckb, d<0>,eni,+ ]
        cc5:    (4,2)           hadd    [ d<0>,, c<0>,  ,    +,q<0>,  + ]

        cc2:    (6,0)           dfrs2   [ q<1>,, ck,ckb, d<1>,eni,+ ]
        cc6:    (8,2)           hadd    [ d<1>,, c<1>,  , c<0>,q<1>,  + ]


        cc3:    (10,0)          dfrs2   [ q<2>,, ck,ckb, d<2>,eni,+ ]
        cc7:    (12,2)          hadd    [ d<2>,, c<2>,  , c<1>,q<2>,  + ]


        cc4:    (14,0)          dfrs2   [ q<3>,, ck,ckb, d<3>,eni,+ ]
        cc8:    (16,2)          hadd    [ d<3>,,      ,  , c<2>,q<3>,  + ]

        cc9:    (0,3)           nand2   [ inck, exck,eni ]
        cx9:    (1,3)           2inv    [ ickb, ,inck,+]
        cx9x:   (0,2)           inv4    [ ick,ickb]
        cc10:   (0,0)           clkb    [ ck, ckb, ,ick ,+]

        cc11:   (16,1)          nand4   [ nsxt, q<0:3> ]
        cx11:   (17,0)          2inv    [ sxt,, nsxt,+ ]

endm

;@~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
;   pg cell with asserted low gi output      ~~~~~~~~~~~
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
;

subcir/UCATAL
pgB [pi,giB:ai,bi]
        G1:     NAND2   [giB,AI,BI]
        G2:     NAND2   [AABI,AI,giB]
        G3:     NAND2   [ABBI,giB,BI]
        G5:     NAND2   [PI,AABI,ABBI]

endm

;@~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
; SINGLE PG CELL                                          ~
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
;

subcir/UCATAL

pg [pi,gi:ai,bi]
        G1:     NAND2   [ABI,AI,BI]
        G2:     NAND2   [AABI,AI,ABI]
        G3:     NAND2   [ABBI,ABI,BI]
        G4:     2INV    [,GI,+,ABI]
        G5:     NAND2   [PI,AABI,ABBI]

endm
```

```
;Q~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
;16 BIT PG BLOCK using asserted high gi ~~~~~~~~
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

;subcir/UCATAL
;
;        PGBLAH  [PI<0:15>,GI<0:15>:  AI<0:15>,BI<0:15>]
;
;  ;using the pg block from library create a 16 bit pg block
;
;  SUPPLIES/+VDD
;
;;       c0 :      PG        [PI<0>,GI<0>,  AI<0>,BI<0>]
; C0__G1   : NAND2[C0__ABI,AI<0>,BI<0>]
; C0__G2   : NAND2[C0__AABI,AI<0>,C0__ABI]
; C0__G3   : NAND2[C0__ABBI,C0__ABI,BI<0>]
; C0__G4   : 2INV[,GI<0>,VDD,C0__ABI]
; C0__G5   : NAND2[PI<0>,C0__AABI,C0__ABBI]
;;       c1 :      PG        [PI<1>,GI<1>,  AI<1>,BI<1>]
; C1__G1   : NAND2[C1__ABI,AI<1>,BI<1>]
; C1__G2   : NAND2[C1__AABI,AI<1>,C1__ABI]
; C1__G3   : NAND2[C1__ABBI,C1__ABI,BI<1>]
; C1__G4   : 2INV[,GI<1>,VDD,C1__ABI]
; C1__G5   : NAND2[PI<1>,C1__AABI,C1__ABBI]
;;       c2 :      PG        [PI<2>,GI<2>,  AI<2>,BI<2>]
; C2__G1   : NAND2[C2__ABI,AI<2>,BI<2>]
; C2__G2   : NAND2[C2__AABI,AI<2>,C2__ABI]
; C2__G3   : NAND2[C2__ABBI,C2__ABI,BI<2>]
; C2__G4   : 2INV[,GI<2>,VDD,C2__ABI]
; C2__G5   : NAND2[PI<2>,C2__AABI,C2__ABBI]
;;       c3 :      PG        [PI<3>,GI<3>,  AI<3>,BI<3>]
; C3__G1   : NAND2[C3__ABI,AI<3>,BI<3>]
; C3__G2   : NAND2[C3__AABI,AI<3>,C3__ABI]
; C3__G3   : NAND2[C3__ABBI,C3__ABI,BI<3>]
; C3__G4   : 2INV[,GI<3>,VDD,C3__ABI]
; C3__G5   : NAND2[PI<3>,C3__AABI,C3__ABBI]
;;       c4 :      PG        [PI<4>,GI<4>,  AI<4>,BI<4>]
; C4__G1   : NAND2[C4__ABI,AI<4>,BI<4>]
; C4__G2   : NAND2[C4__AABI,AI<4>,C4__ABI]
; C4__G3   : NAND2[C4__ABBI,C4__ABI,BI<4>]
; C4__G4   : 2INV[,GI<4>,VDD,C4__ABI]
; C4__G5   : NAND2[PI<4>,C4__AABI,C4__ABBI]
;;       c5 :      PG        [PI<5>,GI<5>,  AI<5>,BI<5>]
; C5__G1   : NAND2[C5__ABI,AI<5>,BI<5>]
; C5__G2   : NAND2[C5__AABI,AI<5>,C5__ABI]
; C5__G3   : NAND2[C5__ABBI,C5__ABI,BI<5>]
; C5__G4   : 2INV[,GI<5>,VDD,C5__ABI]
; C5__G5   : NAND2[PI<5>,C5__AABI,C5__ABBI]
;;       c6 :      PG        [PI<6>,GI<6>,  AI<6>,BI<6>]
; C6__G1   : NAND2[C6__ABI,AI<6>,BI<6>]
; C6__G2   : NAND2[C6__AABI,AI<6>,C6__ABI]
; C6__G3   : NAND2[C6__ABBI,C6__ABI,BI<6>]
; C6__G4   : 2INV[,GI<6>,VDD,C6__ABI]
; C6__G5   : NAND2[PI<6>,C6__AABI,C6__ABBI]
;;       c7 :      PG        [PI<7>,GI<7>,  AI<7>,BI<7>]
; C7__G1   : NAND2[C7__ABI,AI<7>,BI<7>]
; C7__G2   : NAND2[C7__AABI,AI<7>,C7__ABI]
; C7__G3   : NAND2[C7__ABBI,C7__ABI,BI<7>]
; C7__G4   : 2INV[,GI<7>,VDD,C7__ABI]
; C7__G5   : NAND2[PI<7>,C7__AABI,C7__ABBI]
;;       c8 :      PG        [PI<8>,GI<8>,  AI<8>,BI<8>]
; C8__G1   : NAND2[C8__ABI,AI<8>,BI<8>]
; C8__G2   : NAND2[C8__AABI,AI<8>,C8__ABI]
; C8__G3   : NAND2[C8__ABBI,C8__ABI,BI<8>]
; C8__G4   : 2INV[,GI<8>,VDD,C8__ABI]
; C8__G5   : NAND2[PI<8>,C8__AABI,C8__ABBI]
;;       c9 :      PG        [PI<9>,GI<9>,  AI<9>,BI<9>]
; C9__G1   : NAND2[C9__ABI,AI<9>,BI<9>]
; C9__G2   : NAND2[C9__AABI,AI<9>,C9__ABI]
; C9__G3   : NAND2[C9__ABBI,C9__ABI,BI<9>]
; C9__G4   : 2INV[,GI<9>,VDD,C9__ABI]
; C9__G5   : NAND2[PI<9>,C9__AABI,C9__ABBI]
;;       c10:      PG        [PI<10>,GI<10>,   AI<10>,BI<10>]
; C10_G1   : NAND2[C10_ABI,AI<10>,BI<10>]
; C10_G2   : NAND2[C10_AABI,AI<10>,C10_ABI]
; C10_G3   : NAND2[C10_ABBI,C10_ABI,BI<10>]
; C10_G4   : 2INV[,GI<10>,VDD,C10_ABI]
; C10_G5   : NAND2[PI<10>,C10_AABI,C10_ABBI]
;;       c11:      PG        [PI<11>,GI<11>,   AI<11>,BI<11>]
; C11_G1   : NAND2[C11_ABI,AI<11>,BI<11>]
; C11_G2   : NAND2[C11_AABI,AI<11>,C11_ABI]
; C11_G3   : NAND2[C11_ABBI,C11_ABI,BI<11>]
; C11_G4   : 2INV[,GI<11>,VDD,C11_ABI]
; C11_G5   : NAND2[PI<11>,C11_AABI,C11_ABBI]
```

```
;;      c12:    PG      [PI<12>,GI<12>,  AI<12>,BI<12>]
; C12_G1   : NAND2[C12_ABI,AI<12>,BI<12>]
; C12_G2   : NAND2[C12_AABI,AI<12>,C12_ABI]
; C12_G3   : NAND2[C12_ABBI,C12_ABI,BI<12>]
; C12_G4   : 2INV[,GI<12>,VDD,C12_ABI]
; C12_G5   : NAND2[PI<12>,C12_AABI,C12_ABBI]
;;      c13:    PG      [PI<13>,GI<13>,  AI<13>,BI<13>]
; C13_G1   : NAND2[C13_ABI,AI<13>,BI<13>]
; C13_G2   : NAND2[C13_AABI,AI<13>,C13_ABI]
; C13_G3   : NAND2[C13_ABBI,C13_ABI,BI<13>]
; C13_G4   : 2INV[,GI<13>,VDD,C13_ABI]
; C13_G5   : NAND2[PI<13>,C13_AABI,C13_ABBI]
;;      c14:    PG      [PI<14>,GI<14>,  AI<14>,BI<14>]
; C14_G1   : NAND2[C14_ABI,AI<14>,BI<14>]
;; C14_G2   : NAND2[C14_AABI,AI<14>,C14_ABI]
; C14_G3   : NAND2[C14_ABBI,C14_ABI,BI<14>]
; C14_G4   : 2INV[,GI<14>,VDD,C14_ABI]
; C14_G5   : NAND2[PI<14>,C14_AABI,C14_ABBI]
;;      c15:    PG      [PI<15>,GI<15>,  AI<15>,BI<15>]
; C15_G1   : NAND2[C15_ABI,AI<15>,BI<15>]
; C15_G2   : NAND2[C15_AABI,AI<15>,C15_ABI]
; C15_G3   : NAND2[C15_ABBI,C15_ABI,BI<15>]
; C15_G4   : 2INV[,GI<15>,VDD,C15_ABI]
; C15_G5   : NAND2[PI<15>,C15_AABI,C15_ABBI]
;ENDM

;@-------------------------------------------------------------------------
;   pg block using asserted low G     ------------------------------
;-------------------------------------------------------------------------
;

subcir/UCATAL
        PGBL    [PI<0:15>,GB<0:15>:  A<0:15>,BI<0:15>]

  ;using the pg block from adderlib create a 16 bit pg block

;       c0 :    ()              pgB     [PI<0>,GB<0>,  A<0>,BI<0>]
 C01   : (0,1)    NAND2[GB<0>,A<0>,BI<0>]
 C02   : (0,0)    NAND2[C0_L,A<0>,GB<0>]
 C03   : (1,0)    NAND2[C0_M,GB<0>,BI<0>]
 C05   : (1,1)    NAND2[P0,C0_L,C0_M]
 c06   : (1,2)    2inv [np0,,p0,+]
 c07   : (0,3)    inv4 [pi<0>,np0]
;       c1 :    ()              pgB     [PI<1>,GB<1>,  A<1>,BI<1>]
 C11   : (2,1)    NAND2[GB<1>,A<1>,BI<1>]
 C12   : (2,0)    NAND2[C1_L,A<1>,GB<1>]
 C13   : (3,0)    NAND2[C1_M,GB<1>,BI<1>]
 C15   : (3,1)    NAND2[P1,C1_L,C1_M]
 c16   : (3,2)    2inv [np1,,p1,+]
 c17   : (2,3)    inv4 [pi<1>,np1]
;       c2 :    ()              pgB     [PI<2>,GB<2>,  A<2>,BI<2>]
 C21   : (4,1)    NAND2[GB<2>,A<2>,BI<2>]
 C22   : (4,0)    NAND2[C2_L,A<2>,GB<2>]
 C23   : (5,0)    NAND2[C2_M,GB<2>,BI<2>]
 C25   : (5,1)    NAND2[P2,C2_L,C2_M]
 c26   : (5,2)    2inv [np2,,p2,+]
 c27   : (4,3)    inv4 [pi<2>,np2]
;       c3 :    ()              pgB     [PI<3>,GB<3>,  A<3>,BI<3>]
 C31   : (6,1)    NAND2[GB<3>,A<3>,BI<3>]
 C32   : (6,0)    NAND2[C3_L,A<3>,GB<3>]
 C33   : (7,0)    NAND2[C3_M,GB<3>,BI<3>]
 C35   : (7,1)    NAND2[P3,C3_L,C3_M]
 c36   : (7,2)    2inv [np3,,p3,+]
 c37   : (6,3)    inv4 [pi<3>,np3]
;       c4 :    ()              pgB     [PI<4>,GB<4>,  A<4>,BI<4>]
 C41   : (8,1)    NAND2[GB<4>,A<4>,BI<4>]
 C42   : (8,0)    NAND2[C4_L,A<4>,GB<4>]
 C43   : (9,0)    NAND2[C4_M,GB<4>,BI<4>]
 C45   : (9,1)    NAND2[P4,C4_L,C4_M]
 c46   : (9,2)    2inv [np4,,p4,+]
 c47   : (8,3)    inv4 [pi<4>,np4]
;       c5 :    ()              pgB     [PI<5>,GB<5>,  A<5>,BI<5>]
 C51   : (10,1)   NAND2[GB<5>,A<5>,BI<5>]
 C52   : (10,0)   NAND2[C5_L,A<5>,GB<5>]
 C53   : (11,0)   NAND2[C5_M,GB<5>,BI<5>]
 C55   : (11,1)   NAND2[P5,C5_L,C5_M]
 c56   : (11,2)   2inv [np5,,p5,+]
 c57   : (10,3)   inv4 [pi<5>,np5]
;       c6 :    ()              pgB     [PI<6>,GB<6>,  A<6>,BI<6>]
 C61   : (12,1)   NAND2[GB<6>,A<6>,BI<6>]
 C62   : (12,0)   NAND2[C6_L,A<6>,GB<6>]
 C63   : (13,0)   NAND2[C6_M,GB<6>,BI<6>]
 C65   : (13,1)   NAND2[P6,C6_L,C6_M]
 c66   : (13,2)   2inv [np6,,p6,+]
 c67   : (12,3)   inv4 [pi<6>,np6]
```

```
;       c7 :    ()              pgB     [PI<7>,GB<7>,  A<7>,BI<7>]
 C71   : (14,1)   NAND2[GB<7>,A<7>,BI<7>]
 C72   : (14,0)   NAND2[C7_L,A<7>,GB<7>]
 C73   : (15,0)   NAND2[C7_M,GB<7>,BI<7>]
 C75   : (15,1)   NAND2[P7,C7_L,C7_M]
 c76   : (15,2)   2inv [np7,,p7,+]
 c77   : (14,3)   inv4 [pi<7>,np7]
;       c8 :    ()              pgB     [PI<8>,GB<8>,  A<8>,BI<8>]
 C81   : (16,1)   NAND2[GB<8>,A<8>,BI<8>]
 C82   : (16,0)   NAND2[C8_L,A<8>,GB<8>]
 C83   : (17,0)   NAND2[C8_M,GB<8>,BI<8>]
 C85   : (17,1)   NAND2[P8,C8_L,C8_M]
 c86   : (17,2)   2inv [np8,,p8,+]
 c87   : (16,3)   inv4 [pi<8>,np8]
;       c9 :    ()              pgB     [PI<9>,GB<9>,  A<9>,BI<9>]
 C91   : (18,1)   NAND2[GB<9>,A<9>,BI<9>]
 C92   : (18,0)   NAND2[C9_L,A<9>,GB<9>]
 C93   : (19,0)   NAND2[C9_M,GB<9>,BI<9>]
 C95   : (19,1)   NAND2[P9,C9_L,C9_M]
 c96   : (19,2)   2inv [np9,,p9,+]
 c97   : (18,3)   inv4 [pi<9>,np9]
;       c10:    ()              pgB     [PI<10>,GB<10>,  A<10>,BI<10>]
 C101  :         (20,1)   NAND2[GB<10>,A<10>,BI<10>]
 C102  :         (20,0)   NAND2[C10L,A<10>,GB<10>]
 C103  :         (21,0)   NAND2[C10M,GB<10>,BI<10>]
 C105  :         (21,1)   NAND2[P10,C10L,C10M]
 c106  :         (21,2)   2inv [np10,,p10,+]
 c107  :         (20,3)   inv4 [pi<10>,np10]
;       c11:    ()              pgB     [PI<11>,GB<11>,  A<11>,BI<11>]
 C111  :         (22,1)   NAND2[GB<11>,A<11>,BI<11>]
 C112  :         (22,0)   NAND2[C11L,A<11>,GB<11>]
 C113  :         (23,0)   NAND2[C11M,GB<11>,BI<11>]
 C115  :         (23,1)   NAND2[P11,C11L,C11M]
 c116  :         (23,2)   2inv [np11,,p11,+]
 c117  :         (22,3)   inv4 [pi<11>,np11]
;       c12:    ()              pgB     [PI<12>,GB<12>,  A<12>,BI<12>]
 C121  :         (24,1)   NAND2[GB<12>,A<12>,BI<12>]
 C122  :         (24,0)   NAND2[C12L,A<12>,GB<12>]
 C123  :         (25,0)   NAND2[C12M,GB<12>,BI<12>]
 C125  :         (25,1)   NAND2[P12,C12L,C12M]
 c126  :         (25,2)   2inv [np12,,p12,+]
 c127  :         (24,3)   inv4 [pi<12>,np12]
;       c13:    ()              pgB     [PI<13>,GB<13>,  A<13>,BI<13>]
 C131  :         (26,1)   NAND2[GB<13>,A<13>,BI<13>]
 C132  :         (26,0)   NAND2[C13L,A<13>,GB<13>]
 C133  :         (27,0)   NAND2[C13M,GB<13>,BI<13>]
 C135  :         (27,1)   NAND2[P13,C13L,C13M]
 c136  :         (27,2)   2inv [np13,,p13,+]
 c137  :         (26,3)   inv4 [pi<13>,np13]
;       c14:    ()              pgB     [PI<14>,GB<14>,  A<14>,BI<14>]
 C141  :         (28,1)   NAND2[GB<14>,A<14>,BI<14>]
 C142  :         (28,0)   NAND2[C14L,A<14>,GB<14>]
 C143  :         (29,0)   NAND2[C14M,GB<14>,BI<14>]
 C145  :         (29,1)   NAND2[P14,C14L,C14M]
 c146  :         (29,2)   2inv [np14,,p14,+]
 c147  :         (28,3)   inv4 [pi<14>,np14]
;       c15:    ()              pgB     [PI<15>,GB<15>,  A<15>,BI<15>]
 C151  :         (30,1)   NAND2[GB<15>,A<15>,BI<15>]
 C152  :         (30,0)   NAND2[C15L,A<15>,GB<15>]
 C153  :         (31,0)   NAND2[C15M,GB<15>,BI<15>]
 C155  :         (31,1)   NAND2[P15,C15L,C15M]
 c156  :         (31,2)   2inv [np15,,p15,+]
 c157  :         (30,3)   inv4 [pi<15>,np15]
ENDM


;@-------------------------------------------------------------------------
;16 BIT SUMMER BLOCK (just a string of xor gates)                 -------
;-------------------------------------------------------------------------
subcir/UCATAL
SUMBL   [SI<0:15>: PI<0:15>,CIN,CI<0:14>]
        G0:     (0,0)           EXOR[ SI<0> ,,,,PI<0>, CIN ,-,-,-]
        G1:     (2,0)           EXOR[ SI<1> ,,,,PI<1>, CI<0>,-,-,-]
        G2:     (4,0)           EXOR[ SI<2> ,,,,PI<2>, CI<1>,-,-,-]
        G3:     (6,0)           EXOR[ SI<3> ,,,,PI<3>, CI<2>,-,-,-]
        G4:     (8,0)           EXOR[ SI<4> ,,,,PI<4>, CI<3>,-,-,-]
        G5:     (10,0)          EXOR[ SI<5> ,,,,PI<5>, CI<4>,-,-,-]
        G6:     (12,0)          EXOR[ SI<6> ,,,,PI<6>, CI<5>,-,-,-]
        G7:     (14,0)          EXOR[ SI<7> ,,,,PI<7>, CI<6>,-,-,-]
        G8:     (16,0)          EXOR[ SI<8> ,,,,PI<8>, CI<7>,-,-,-]
        G9:     (18,0)          EXOR[ SI<9> ,,,,PI<9>, CI<8>,-,-,-]
        G10:    (20,0)          EXOR[ SI<10> ,,,,PI<10>, CI<9>,-,-,-]
        G11:    (22,0)          EXOR[ SI<11> ,,,,PI<11>, CI<10>,-,-,-]
```

```
        G12:    (24,0)          EXOR[ SI<12> ,,,,PI<12>, CI<11>,-,-,-]
        G13:    (26,0)          EXOR[ SI<13> ,,,,PI<13>, CI<12>,-,-,-]
        G14:    (28,0)          EXOR[ SI<14> ,,,,PI<14>, CI<13>,-,-,-]
        G15:    (30,0)          EXOR[ SI<15> ,,,,PI<15>, CI<14>,-,-,-]

endm

;@-------------------------------------------------------------------
;16 BIT SUMMER BLOCK WITH ACTIVE LOW OUTPUTS (just a string of xnor gates)
;-------------------------------------------------------------------
;
subcir/ucatal
NOTSUM  [SIB<0:15>: PI<0:15>,CIN,CI<0:14>]
        G0:     (0,0)           EXNOR[ SIB<0> ,,,,PI<0>, CIN ,-,-,-]
        G1:     (2,0)           EXNOR[ SIB<1> ,,,,PI<1>, CI<0>,-,-,-]
        G2:     (4,0)           EXNOR[ SIB<2> ,,,,PI<2>, CI<1>,-,-,-]
        G3:     (6,0)           EXNOR[ SIB<3> ,,,,PI<3>, CI<2>,-,-,-]
        G4:     (8,0)           EXNOR[ SIB<4> ,,,,PI<4>, CI<3>,-,-,-]
        G5:     (10,0)          EXNOR[ SIB<5> ,,,,PI<5>, CI<4>,-,-,-]
        G6:     (12,0)          EXNOR[ SIB<6> ,,,,PI<6>, CI<5>,-,-,-]
        G7:     (14,0)          EXNOR[ SIB<7> ,,,,PI<7>, CI<6>,-,-,-]
        G8:     (16,0)          EXNOR[ SIB<8> ,,,,PI<8>, CI<7>,-,-,-]
        G9:     (18,0)          EXNOR[ SIB<9> ,,,,PI<9>, CI<8>,-,-,-]
        G10:    (20,0)          EXNOR[ SIB<10> ,,,,PI<10>, CI<9>,-,-,-]
        G11:    (22,0)          EXNOR[ SIB<11> ,,,,PI<11>, CI<10>,-,-,-]
        G12:    (24,0)          EXNOR[ SIB<12> ,,,,PI<12>, CI<11>,-,-,-]
        G13:    (26,0)          EXNOR[ SIB<13> ,,,,PI<13>, CI<12>,-,-,-]
        G14:    (28,0)          EXNOR[ SIB<14> ,,,,PI<14>, CI<13>,-,-,-]
        G15:    (30,0)          EXNOR[ SIB<15> ,,,,PI<15>, CI<14>,-,-,-]

endm


;@-------------------------------------------------------------
; 4 bit bcla unit                          ------------------
;-------------------------------------------------------------

subcir/UCATAL
        bcla [ci<0:2>, pstr, gstB :  pi<0:3>, giB<0:3>,cin]

        G1:     (8,1)           NAND2   [ci<0>,giB<0>,pc]
        G2:     (7,0)           NAND2   [pc,cin,pi<0>]
        G3:     (6,0)           NAND2   [pg0,g0,pi<1>]
        G4:     (12,0)          NAND2   [pg1,g1,pi<2>]
        G5:     (2,0)           NAND2   [pg2,g2,pi<3>]

; use extra inverters from the first
;three 3nands to produce high asserted g0 g1 g2

        G6:     (6,1)           NAND3   [ci<1>, g0 , giB<1>, pg0, ppc, giB<0>  ]
        G7:     (4,1)           NAND3   [ppc, g1 , pi<1>, pi<0>, cin, giB<1>   ]
        G8:     (12,1)          NAND3   [ppg0, g2 , pi<2>, pi<1>, g0, giB<2>   ]
        G9:     (0,1)           NAND3   [ppg1, pstr, pi<3>, pi<2>, g1, pstB    ]

        G10:    (10,1)          NAND4   [pppc, pi<2>, pi<1>, pi<0>, cin]
        G11:    (10,0)          NAND4   [ci<2>, giB<2>, pg1, ppg0, pppc]
        G12:    (2,1)           NAND4   [3pg0, g0, pi<1>, pi<2>, pi<3>]
        G13:    (4,0)           NAND4   [gstI, giB<3>, pg2, ppg1, 3pg0]
        G14:    (0,0)           NAND4   [pstB, pi<0>, pi<1>, pi<2>, pi<3>]


        G15:    (9,1)           2INV    [,      gstb,   +,      GSTI]
endm
;-------@-------------------------------------------------------
;-------    REGISTERS VARIOUS                  ----------------------
;-------------------------------------------------------------------

;@------
SUBCIR/ucatal

        SHR [ QB<0:15>  :  EXCK, SLI ]
;declare/ Q<0:14>,QD<0:14>,QI<0:14>
declare/ QD<0:15>
        g0x:    (17,2)          2inv    [ xck,icku, icku,exck ]
        g0:     (18,2)          inv4    [ inck , xck ]
        G1:     (16,0)          clkb    [ CKl, CKlb, ,inCK ,+]
        G1x:    (18,0)          clkb    [ CKu, CKub, ,inCK ,+]

        D0:     (0,0)           df      [ QD< 0>, , ckl, cklB, SLI  ]
        D1:     (2,0)           df      [ QD< 1>, , ckl, cklB, QD< 0> ]
        D2:     (4,0)           df      [ QD< 2>, , ckl, cklB, QD< 1> ]
        D3:     (6,0)           df      [ QD< 3>, , ckl, cklB, QD< 2> ]
        D4:     (8,0)           df      [ QD< 4>, , ckl, cklB, QD< 3> ]
        D5:     (10,0)          df      [ QD< 5>, , ckl, cklB, QD< 4> ]
        D6:     (12,0)          df      [ QD< 6>, , ckl, cklB, QD< 5> ]
        D7:     (14,0)          df      [ QD< 7>, , ckl, cklB, QD< 6> ]
```

```
        D8:     (20,0)          df      [ QD< 8>, , cku, ckuB, QD< 7> ]
        D9:     (22,0)          df      [ QD< 9>, , cku, ckuB, QD< 8> ]
        D10:    (24,0)          df      [ QD<10>, , cku, ckuB, QD< 9> ]
        D11:    (26,0)          df      [ QD<11>, , cku, ckuB, QD<10> ]
        D12:    (28,0)          df      [ QD<12>, , cku, ckuB, QD<11> ]
        D13:    (30,0)          df      [ QD<13>, , cku, ckuB, QD<12> ]
        D14:    (32,0)          df      [ QD<14>, , cku, ckuB, QD<13> ]
        D15:    (34,0)          df      [ QD<15>, , cku, ckuB, QD<14> ]

        I0:     (0,2)           INV4    [QB< 0>,        QD< 0>]
        I1:     (2,2)           INV4    [QB< 1>,        QD< 1>]
        I2:     (4,2)           INV4    [QB< 2>,        QD< 2>]
        I3:     (6,2)           INV4    [QB< 3>,        QD< 3>]
        I4:     (8,2)           INV4    [QB< 4>,        QD< 4>]
        I5:     (10,2)          INV4    [QB< 5>,        QD< 5>]
        I6:     (12,2)          INV4    [QB< 6>,        QD< 6>]
        I7:     (14,2)          INV4    [QB< 7>,        QD< 7>]

        I8:     (20,2)          INV4    [QB< 8>,        QD< 8>]
        I9:     (22,2)          INV4    [QB< 9>,        QD< 9>]
        I10:    (24,2)          INV4    [QB<10>,        QD<10>]
        I11:    (26,2)          INV4    [QB<11>,        QD<11>]
        I12:    (28,2)          INV4    [QB<12>,        QD<12>]
        I13:    (30,2)          INV4    [QB<13>,        QD<13>]
        I14:    (32,2)          INV4    [QB<14>,        QD<14>]
        I15:    (34,2)          INV4    [QB<15>,        QD<15>]

;       D0:     (0,0)           df      [ Q< 0>,QB< 0>, ckl, cklB, SLI  ]
;       D1:     (2,0)           df      [ Q< 1>,QB< 1>, ckl, cklB, QD< 0> ]
;       D2:     (4,0)           df      [ Q< 2>,QB< 2>, ckl, cklB, QD< 1> ]
;       D3:     (6,0)           df      [ Q< 3>,QB< 3>, ckl, cklB, QD< 2> ]
;       D4:     (8,0)           df      [ Q< 4>,QB< 4>, ckl, cklB, QD< 3> ]
;       D5:     (10,0)          df      [ Q< 5>,QB< 5>, ckl, cklB, QD< 4> ]
;       D6:     (12,0)          df      [ Q< 6>,QB< 6>, ckl, cklB, QD< 5> ]
;       D7:     (14,0)          df      [ Q< 7>,QB< 7>, ckl, cklB, QD< 6> ]
;
;       D8:     (22,0)          df      [ Q< 8>,QB< 8>, cku, ckuB, QD< 7> ]
;       D9:     (24,0)          df      [ Q< 9>,QB< 9>, cku, ckuB, QD< 8> ]
;       D10:    (26,0)          df      [ Q<10>,QB<10>, cku, ckuB, QD< 9> ]
;       D11:    (28,0)          df      [ Q<11>,QB<11>, cku, ckuB, QD<10> ]
;       D12:    (30,0)          df      [ Q<12>,QB<12>, cku, ckuB, QD<11> ]
;       D13:    (32,0)          df      [ Q<13>,QB<13>, cku, ckuB, QD<12> ]
;       D14:    (34,0)          df      [ Q<14>,QB<14>, cku, ckuB, QD<13> ]
;       D15:    (36,0)          df      [       ,QB<15>, cku, ckuB, QD<14> ]

;       dl1:    (0,2)           2inv    [ qd< 0>, qi< 0>, qi< 0>,q< 0> ]
;       dl2:    (1,2)           2inv    [ qd< 1>, qi< 1>, qi< 1>,q< 1> ]
;       dl3:    (2,2)           2inv    [ qd< 2>, qi< 2>, qi< 2>,q< 2> ]
;       dl4:    (3,2)           2inv    [ qd< 3>, qi< 3>, qi< 3>,q< 3> ]
;       dl5:    (4,2)           2inv    [ qd< 4>, qi< 4>, qi< 4>,q< 4> ]
;       dl6:    (5,2)           2inv    [ qd< 5>, qi< 5>, qi< 5>,q< 5> ]
;       dl7:    (6,2)           2inv    [ qd< 6>, qi< 6>, qi< 6>,q< 6> ]
;       dl8:    (7,2)           2inv    [ qd< 7>, qi< 7>, qi< 7>,q< 7> ]
;       dl9:    (8,2)           2inv    [ qd< 8>, qi< 8>, qi< 8>,q< 8> ]
;       dl19:   (9,2)           2inv    [ qd< 9>, qi< 9>, qi< 9>,q< 9> ]
;       dl11:   (10,2)          2inv    [ qd<10>, qi<10>, qi<10>,q<10> ]
;       dl12:   (11,2)          2inv    [ qd<11>, qi<11>, qi<11>,q<11> ]
;       dl13:   (12,2)          2inv    [ qd<12>, qi<12>, qi<12>,q<12> ]
;       dl14:   (13,2)          2inv    [ qd<13>, qi<13>, qi<13>,q<13> ]
;       dl15:   (14,2)          2inv    [ qd<14>, qi<14>, qi<14>,q<14> ]



ENDM

;@~~~~~
SUBCIR/UCATAL

        ABREG [ QB<0:15>   :  D<0:15>, EXCK, LOAD ]
DECLARE/Q<0:15>

        G1:     (16,2)          NAND2   [ INCK, EXCK, LOAD ]
        g1x1:   (17,2)          2inv    [ ickb,, inck,+]
        g1x2:   (18,2)          inv4    [ ick,  ickb ]

        G3:     (16,0)          clkb    [ CKl, CKlB, ,ICK ,+]
        G3x:    (18,0)          clkb    [ CKu, CKuB, ,ICK ,+]

        D0:     (0,0)           df      [ Q< 0>,, ckl, cklB, D< 0>]
        D1:     (2,0)           df      [ Q< 1>,, ckl, cklB, D< 1>]
        D2:     (4,0)           df      [ Q< 2>,, ckl, cklB, D< 2>]
```

```
        D3:     (8,0)           df      [ Q< 3>,, ckl, cklB, D< 3>]
        D4:     (8,0)           df      [ Q< 4>,, ckl, cklB, D< 4>]
        D5:     (10,0)          df      [ Q< 5>,, ckl, cklB, D< 5>]
        D6:     (12,0)          df      [ Q< 6>,, ckl, cklB, D< 6>]
        D7:     (14,0)          df      [ Q< 7>,, ckl, cklB, D< 7>]

        D8:     (20,0)          df      [ Q< 8>,, cku, ckuB, D< 8>]
        D9:     (22,0)          df      [ Q< 9>,, cku, ckuB, D< 9>]
        D10:    (24,0)          df      [ Q<10>,, cku, ckuB, D<10>]
        D11:    (26,0)          df      [ Q<11>,, cku, ckuB, D<11>]
        D12:    (28,0)          df      [ Q<12>,, cku, ckuB, D<12>]
        D13:    (30,0)          df      [ Q<13>,, cku, ckuB, D<13>]
        D14:    (32,0)          df      [ Q<14>,, cku, ckuB, D<14>]
        D15:    (34,0)          df      [ Q<15>,, cku, ckuB, D<15>]

        I0:     (0,2)           INV4    [QB< 0>,        Q< 0>]
        I1:     (2,2)           INV4    [QB< 1>,        Q< 1>]
        I2:     (4,2)           INV4    [QB< 2>,        Q< 2>]
        I3:     (6,2)           INV4    [QB< 3>,        Q< 3>]
        I4:     (8,2)           INV4    [QB< 4>,        Q< 4>]
        I5:     (10,2)          INV4    [QB< 5>,        Q< 5>]
        I6:     (12,2)          INV4    [QB< 6>,        Q< 6>]
        I7:     (14,2)          INV4    [QB< 7>,        Q< 7>]

        I8:     (20,2)          INV4    [QB< 8>,        Q< 8>]
        I9:     (22,2)          INV4    [QB< 9>,        Q< 9>]
        I10:    (24,2)          INV4    [QB<10>,        Q<10>]
        I11:    (26,2)          INV4    [QB<11>,        Q<11>]
        I12:    (28,2)          INV4    [QB<12>,        Q<12>]
        I13:    (30,2)          INV4    [QB<13>,        Q<13>]
        I14:    (32,2)          INV4    [QB<14>,        Q<14>]
        I15:    (34,2)          INV4    [QB<15>,        Q<15>]

ENDM

;@~~~~~ THIS REGISTER IS SET TO 8000 HEX BY S15
SUBCIR/UCATAL

        BREG [ QB<0:15>  :  D<0:15>, EXCK, LOAD,S15 ]

DECLARE/Q<0:15>

        G1:     (16,3)          NAND2   [ INCK, EXCK, LOAD ]
        g1x1:   (17,3)          2inv    [ ickb,, inck,+]
        g1x2:   (16,2)          inv4    [ ick,  ickb ]

        G3:     (16,0)          clkb    [ CKl, CKlB, ,ICK ,+]
        G3x:    (18,0)          clkb    [ CKu, CKuB, ,ICK ,+]

        G2:     (18,2)          INV8    [ NS15, S15 ]

        D0:     (0,0)           dfrs2   [ Q< 0>,, CKl, cklB, D< 0>,     NS15, + ]
        D1:     (2,0)           dfrs2   [ Q< 1>,, CKl, cklB, D< 1>,     NS15, + ]
        D2:     (4,0)           dfrs2   [ Q< 2>,, CKl, cklB, D< 2>,     NS15, + ]
        D3:     (6,0)           dfrs2   [ Q< 3>,, CKl, cklB, D< 3>,     NS15, + ]
        D4:     (8,0)           dfrs2   [ Q< 4>,, CKl, cklB, D< 4>,     NS15, + ]
        D5:     (10,0)          dfrs2   [ Q< 5>,, CKl, cklB, D< 5>,     NS15, + ]
        D6:     (12,0)          dfrs2   [ Q< 6>,, CKl, cklB, D< 6>,     NS15, + ]
        D7:     (14,0)          dfrs2   [ Q< 7>,, CKl, cklB, D< 7>,     NS15, + ]

        D8:     (20,0)          dfrs2   [ Q< 8>,, CKu, ckuB, D< 8>,     NS15, + ]
        D9:     (22,0)          dfrs2   [ Q< 9>,, CKu, ckuB, D< 9>,     NS15, + ]
        D10:    (24,0)          dfrs2   [ Q<10>,, CKu, ckuB, D<10>,     NS15, + ]
        D11:    (26,0)          dfrs2   [ Q<11>,, CKu, ckuB, D<11>,     NS15, + ]
        D12:    (28,0)          dfrs2   [ Q<12>,, CKu, ckuB, D<12>,     NS15, + ]
        D13:    (30,0)          dfrs2   [ Q<13>,, CKu, ckuB, D<13>,     NS15, + ]
        D14:    (32,0)          dfrs2   [ Q<14>,, CKu, ckuB, D<14>,     NS15, + ]
        D15:    (34,0)          dfrs2   [ Q<15>,, CKu, ckuB, D<15>,     +, NS15 ]

        I0:     (0,3)           INV4    [QB< 0>,        Q< 0>]
        I1:     (2,3)           INV4    [QB< 1>,        Q< 1>]
        I2:     (4,3)           INV4    [QB< 2>,        Q< 2>]
        I3:     (6,3)           INV4    [QB< 3>,        Q< 3>]
        I4:     (8,3)           INV4    [QB< 4>,        Q< 4>]
        I5:     (10,3)          INV4    [QB< 5>,        Q< 5>]
        I6:     (12,3)          INV4    [QB< 6>,        Q< 6>]
        I7:     (14,3)          INV4    [QB< 7>,        Q< 7>]

        I8:     (20,3)          INV4    [QB< 8>,        Q< 8>]
        I9:     (22,3)          INV4    [QB< 9>,        Q< 9>]
        I10:    (24,3)          INV4    [QB<10>,        Q<10>]
        I11:    (26,3)          INV4    [QB<11>,        Q<11>]
        I12:    (28,3)          INV4    [QB<12>,        Q<12>]
        I13:    (30,3)          INV4    [QB<13>,        Q<13>]
        I14:    (32,3)          INV4    [QB<14>,        Q<14>]
```

```
        I15:    (34,3)          INV4    [QB<15>,        Q<15>]

ENDM

;@~~~~
SUBCIR/UCATAL

        XYREG [ QB<0:15>  :  D<0:15>, EXCK, LUPP, LLOW ]

DECLARE/Q<0:15>

        G1:     (18,2)          NAND2   [ icku, EXCK, LUPP  ]
        G2:     (16,2)          NAND2   [ ickl, EXCK, LLOW  ]
        g1x:    (17,2)          2inv    [ kl,iclb,      iclb,ickl ]
        g2x:    (19,2)          2inv    [ ku,icub,      icub,icku ]

        G3:     (18,0)          clkb    [ CKU, CKBU, ,ku ,+]
        G4:     (16,0)          clkb    [ CKL, CKBL, ,kl ,+]

        D0:     (0,0)           df      [ Q< 0>,, CKL, CKBL, D< 0> ]
        D1:     (2,0)           df      [ Q< 1>,, CKL, CKBL, D< 1> ]
        D2:     (4,0)           df      [ Q< 2>,, CKL, CKBL, D< 2> ]
        D3:     (6,0)           df      [ Q< 3>,, CKL, CKBL, D< 3> ]
        D4:     (8,0)           df      [ Q< 4>,, CKL, CKBL, D< 4> ]
        D5:     (10,0)          df      [ Q< 5>,, CKL, CKBL, D< 5> ]
        D6:     (12,0)          df      [ Q< 6>,, CKL, CKBL, D< 6> ]
        D7:     (14,0)          df      [ Q< 7>,, CKL, CKBL, D< 7> ]

        D8:     (20,0)          df      [ Q< 8>,, CKU, CKBU, D< 8> ]
        D9:     (22,0)          df      [ Q< 9>,, CKU, CKBU, D< 9> ]
        D10:    (24,0)          df      [ Q<10>,, CKU, CKBU, D<10> ]
        D11:    (26,0)          df      [ Q<11>,, CKU, CKBU, D<11> ]
        D12:    (28,0)          df      [ Q<12>,, CKU, CKBU, D<12> ]
        D13:    (30,0)          df      [ Q<13>,, CKU, CKBU, D<13> ]
        D14:    (32,0)          df      [ Q<14>,, CKU, CKBU, D<14> ]
        D15:    (34,0)          df      [ Q<15>,, CKU, CKBU, D<15> ]

        I0:     (0,2)           INV4    [QB< 0>,        Q< 0>]
        I1:     (2,2)           INV4    [QB< 1>,        Q< 1>]
        I2:     (4,2)           INV4    [QB< 2>,        Q< 2>]
        I3:     (6,2)           INV4    [QB< 3>,        Q< 3>]
        I4:     (8,2)           INV4    [QB< 4>,        Q< 4>]
        I5:     (10,2)          INV4    [QB< 5>,        Q< 5>]
        I6:     (12,2)          INV4    [QB< 6>,        Q< 6>]
        I7:     (14,2)          INV4    [QB< 7>,        Q< 7>]

        I8:     (20,2)          INV4    [QB< 8>,        Q< 8>]
        I9:     (22,2)          INV4    [QB< 9>,        Q< 9>]
        I10:    (24,2)          INV4    [QB<10>,        Q<10>]
        I11:    (26,2)          INV4    [QB<11>,        Q<11>]
        I12:    (28,2)          INV4    [QB<12>,        Q<12>]
        I13:    (30,2)          INV4    [QB<13>,        Q<13>]
        I14:    (32,2)          INV4    [QB<14>,        Q<14>]
        I15:    (34,2)          INV4    [QB<15>,        Q<15>]

ENDM
;@~~~~~~
SUBCIR/UCATAL

        XY1REG [ Q<0:15>,QB<0:15>  :  D<0:15>, EXCK, LUPP, LLOW ]

DECLARE/QI<0:15>,QIB<0:15>

        G1:     (18,2)          NAND2   [ icku, EXCK, LUPP  ]
        G2:     (16,2)          NAND2   [ ickl, EXCK, LLOW  ]
        g1x:    (17,2)          2inv    [ kl,iclb,      iclb,ickl ]
        g2x:    (19,2)          2inv    [ ku,icub,      icub,icku ]

        G3:     (18,0)          clkb    [ CKU, CKBU, ,ku ,+]
        G4:     (16,0)          clkb    [ CKL, CKBL, ,kl ,+]

        D0:     (0,0)           df      [ QI< 0>,QIB< 0>, CKL, CKBL, D< 0> ]
        D1:     (2,0)           df      [ QI< 1>,QIB< 1>, CKL, CKBL, D< 1> ]
        D2:     (4,0)           df      [ QI< 2>,QIB< 2>, CKL, CKBL, D< 2> ]
        D3:     (6,0)           df      [ QI< 3>,QIB< 3>, CKL, CKBL, D< 3> ]
        D4:     (8,0)           df      [ QI< 4>,QIB< 4>, CKL, CKBL, D< 4> ]
        D5:     (10,0)          df      [ QI< 5>,QIB< 5>, CKL, CKBL, D< 5> ]
        D6:     (12,0)          df      [ QI< 6>,QIB< 6>, CKL, CKBL, D< 6> ]
        D7:     (14,0)          df      [ QI< 7>,QIB< 7>, CKL, CKBL, D< 7> ]

        D8:     (20,0)          df      [ QI< 8>,QIB< 8>, CKU, CKBU, D< 8> ]
        D9:     (22,0)          df      [ QI< 9>,QIB< 9>, CKU, CKBU, D< 9> ]
        D10:    (24,0)          df      [ QI<10>,QIB<10>, CKU, CKBU, D<10> ]
        D11:    (26,0)          df      [ QI<11>,QIB<11>, CKU, CKBU, D<11> ]
```

```
        D12:    (28,0)          df      [ QI<12>,QIB<12>, CKU, CKBU, D<12> ]
        D13:    (30,0)          df      [ QI<13>,QIB<13>, CKU, CKBU, D<13> ]
        D14:    (32,0)          df      [ QI<14>,QIB<14>, CKU, CKBU, D<14> ]
        D15:    (34,0)          df      [ QI<15>,QIB<15>, CKU, CKBU, D<15> ]

        I0:     (0,2)           INV4    [QB< 0>,        QI< 0>]
        I1:     (2,2)           INV4    [QB< 1>,        QI< 1>]
        I2:     (4,2)           INV4    [QB< 2>,        QI< 2>]
        I3:     (6,2)           INV4    [QB< 3>,        QI< 3>]
        I4:     (8,2)           INV4    [QB< 4>,        QI< 4>]
        I5:     (10,2)          INV4    [QB< 5>,        QI< 5>]
        I6:     (12,2)          INV4    [QB< 6>,        QI< 6>]
        I7:     (14,2)          INV4    [QB< 7>,        QI< 7>]

        I8:     (20,2)          INV4    [QB< 8>,        QI< 8>]
        I9:     (22,2)          INV4    [QB< 9>,        QI< 9>]
        I10:    (24,2)          INV4    [QB<10>,        QI<10>]
        I11:    (26,2)          INV4    [QB<11>,        QI<11>]
        I12:    (28,2)          INV4    [QB<12>,        QI<12>]
        I13:    (30,2)          INV4    [QB<13>,        QI<13>]
        I14:    (32,2)          INV4    [QB<14>,        QI<14>]
        I15:    (34,2)          INV4    [QB<15>,        QI<15>]

        I16:    (0,3)           INV4    [Q< 0>, QIB< 0>]
        I17:    (2,3)           INV4    [Q< 1>, QIB< 1>]
        I18:    (4,3)           INV4    [Q< 2>, QIB< 2>]
        I19:    (6,3)           INV4    [Q< 3>, QIB< 3>]
        I20:    (8,3)           INV4    [Q< 4>, QIB< 4>]
        I21:    (10,3)          INV4    [Q< 5>, QIB< 5>]
        I22:    (12,3)          INV4    [Q< 6>, QIB< 6>]
        I23:    (14,3)          INV4    [Q< 7>, QIB< 7>]

        I24:    (20,3)          INV4    [Q< 8>, QIB< 8>]
        I25:    (22,3)          INV4    [Q< 9>, QIB< 9>]
        I26:    (24,3)          INV4    [Q<10>, QIB<10>]
        I27:    (26,3)          INV4    [Q<11>, QIB<11>]
        I28:    (28,3)          INV4    [Q<12>, QIB<12>]
        I29:    (30,3)          INV4    [Q<13>, QIB<13>]
        I30:    (32,3)          INV4    [Q<14>, QIB<14>]
        I31:    (34,3)          INV4    [Q<15>, QIB<15>]

ENDM
;@-------------------------------------------------------------------------------
;---------- TRISTATE BUFFER                                          ----------
;-------------------------------------------------------------------------------


SUBCIR/UCATAL
        TRI  [ OB<0:15> : IN<0:15>, SELECT ]

        g1x1:   (10,1)          2inv    [ ickb,, SELECT,+]
        g1x2:   (10,0)          inv4    [ ick,  ickb ]

        G3:     (12,0)          clkb    [ CK1, CK1B, ,ICK ,+]
        G3x:    (8,0)           clkb    [ CKu, CKuB, ,ICK ,+]

DECLARE/OB1<0:15>,OB2<0:15>

        D0:     (0,0)           tm      [ OB1<  0>, cku, ckuB, IN<  0> ]
        D0X:    (0,1)           tm      [ OB2<  0>, cku, ckuB, IN<  0> ]
        P0:     PARAL   [ OB< 0>,       OB1< 0>,OB2< 0> ]

        D1:     (1,0)           tm      [ OB1<  1>, cku, ckuB, IN<  1> ]
        D1X:    (1,1)           tm      [ OB2<  1>, cku, ckuB, IN<  1> ]
        P1:     PARAL   [ OB< 1>,       OB1< 1>,OB2< 1> ]

        D2:     (2,0)           tm      [ OB1<  2>, cku, ckuB, IN<  2> ]
        D2X:    (2,1)           tm      [ OB2<  2>, cku, ckuB, IN<  2> ]
        P2:     PARAL   [ OB< 2>,       OB1< 2>,OB2< 2> ]

        D3:     (3,0)           tm      [ OB1<  3>, cku, ckuB, IN<  3> ]
        D3X:    (3,1)           tm      [ OB2<  3>, cku, ckuB, IN<  3> ]
        P3:     PARAL   [ OB< 3>,       OB1< 3>,OB2< 3> ]

        D4:     (4,0)           tm      [ OB1<  4>, cku, ckuB, IN<  4> ]
        D4X:    (4,1)           tm      [ OB2<  4>, cku, ckuB, IN<  4> ]
        P4:     PARAL   [ OB< 4>,       OB1< 4>,OB2< 4> ]

        D5:     (5,0)           tm      [ OB1<  5>, cku, ckuB, IN<  5> ]
        D5X:    (5,1)           tm      [ OB2<  5>, cku, ckuB, IN<  5> ]
        P5:     PARAL   [ OB< 5>,       OB1< 5>,OB2< 5> ]

        D6:     (6,0)           tm      [ OB1<  6>, cku, ckuB, IN<  6> ]
        D6X:    (6,1)           tm      [ OB2<  6>, cku, ckuB, IN<  6> ]
```

```
P6:     PARAL   [ OB< 6>,      OB1< 6>,OB2< 6> ]

D7:     (7,0)       tm      [ OB1<  7>, cku, ckuB, IN<  7> ]
D7X:    (7,1)       tm      [ OB2<  7>, cku, ckuB, IN<  7> ]
P7:     PARAL   [ OB< 7>,      OB1< 7>,OB2< 7> ]


D8:     (14,0)      tm      [ OB1<  8>, ckl, cklB, IN<  8> ]
D8X:    (14,1)      tm      [ OB2<  8>, ckl, cklB, IN<  8> ]
P8:     PARAL   [ OB< 8>,      OB1< 8>,OB2< 8> ]

D9:     (15,0)      tm      [ OB1<  9>, ckl, cklB, IN<  9> ]
D9X:    (15,1)      tm      [ OB2<  9>, ckl, cklB, IN<  9> ]
P9:     PARAL   [ OB< 9>,      OB1< 9>,OB2< 9> ]

D10:    (16,0)      tm      [ OB1< 10>, ckl, cklB, IN< 10> ]
D10X:   (16,1)      tm      [ OB2< 10>, ckl, cklB, IN< 10> ]
P10:    PARAL   [ OB<10>,      OB1<10>,OB2<10> ]

D11:    (17,0)      tm      [ OB1< 11>, ckl, cklB, IN< 11> ]
D11X:   (17,1)      tm      [ OB2< 11>, ckl, cklB, IN< 11> ]
P11:    PARAL   [ OB<11>,      OB1<11>,OB2<11> ]

D12:    (18,0)      tm      [ OB1< 12>, ckl, cklB, IN< 12> ]
D12X:   (18,1)      tm      [ OB2< 12>, ckl, cklB, IN< 12> ]
P12:    PARAL   [ OB<12>,      OB1<12>,OB2<12> ]

D13:    (19,0)      tm      [ OB1< 13>, ckl, cklB, IN< 13> ]
D13X:   (19,1)      tm      [ OB2< 13>, ckl, cklB, IN< 13> ]
P13:    PARAL   [ OB<13>,      OB1<13>,OB2<13> ]

D14:    (20,0)      tm      [ OB1< 14>, ckl, cklB, IN< 14> ]
D14X:   (20,1)      tm      [ OB2< 14>, ckl, cklB, IN< 14> ]
P14:    PARAL   [ OB<14>,      OB1<14>,OB2<14> ]

D15:    (21,0)      tm      [ OB1< 15>, ckl, cklB, IN< 15> ]
D15X:   (21,1)      tm      [ OB2< 15>, ckl, cklB, IN< 15> ]
P15:    PARAL   [ OB<15>,      OB1<15>,OB2<15> ]
ENDM

end
```

```
options
        ALL
        PERIOD=200ns
        XPROP
        TMAX=2000
        NPARAL=2
!        NPARAL=1
        SIM(1)=MAX
        SIM(2)=MIN
!faults
!  clasp
!  AUTO
!        STROBE

inputs

XCLOCK=Q
XRESET=0>1,1>800


;1
XBACK=1>3,0>5,$
   1>200


;1
XBG=  1>50,$
   0>200

XMEMR=  1>7,  0>8,      1>9,  0>10,$
        1>11, 0>12,     1>13, 0>14,     1>15, 0>16,     1>17, 0>18, $
        1>19, 0>20,     1>21, 0>22,     1>23, 0>24,     1>25, 0>26, $
     1>200

;1
DATAEX<7:0>/HEX=00>11,  AA >13, AA >15,  CC >17, CC >19,  CA >21, AA >23,  EE >25, CC >27,
;
;  co-ord          (   x1l     x1u   ,  y1l     y1u )  ( x2l     x2u   ,   y2l     y2u   )
      00>200


end
```

C L A S S I C D I S P L A Y E D I T O R

ECLASSIC E2.6 17-JUL-86 16:37 04-00

```
Circuit compiled using CCLASSIC E2.6 17-JUL-86 13:03 04-00 on 18-SEP-86 at 17:11:39
Circuit file spec DISK$PSCUK_TS:[IZZARD_M.CLASSIC.ROUT2]Z.CCL;30
Circuit name Z
Compilation time units    10PS
Technology library CLA5000  version V1R5


Circuit simulated using SCLASSIC E2.6 17-JUL-86 16:34 04-00 on 18-SEP-86 at 18:12:42
Simulation file spec DISK$PSCUK_TS:[IZZARD_M.CLASSIC.ROUT2]Z.SCL;1
Simulation name Z
Simulation period   200NS
Min_mask(01) Typ_mask(00) Max_mask(10)
Simulation to spec : Supply =   5V  10%  Temp =    0->  70
Error nodes have been simulated if present
Circuit simulated with X propagation ON
```

NODE	INST	PIXEL	PIXEL	QX2BAR	QY2BAR	BUSA1	BUSA2	SI	SLI	BUSA	BUSB	QABAR	QBBAR	NEG	OV	SLI	NXRA
INDX	HEX	HEX	HEX	HEX	HEX	HEX	HEX	HEX	HEX	HEX	HEX	HEX	HEX	HEX	HEX	HEX	HEX
SIM	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
I/O																	
10000	00	XXXX	XXXX	XXXX	XXXX	XXXX	XXXX	XXXX	X	XXXX	XXXX	XXXX	XXXX	X	X	X	1
90000	20	XXXX	XXXX	XXXX	XXXX	XXXX	XXXX	XXXX	X	XXXX	XXXX	XXXX	XXXX	X	X	X	1
210000	10	XXXX	XXXX	XXXX	XXXX	XXXX	XXXX	XXXX	X	XXXX	XXXX	XXXX	XXXX	X	X	X	1
250000	30	XXXX	XXXX	XXXX	XXXX	XXXX	XXXX	XXXX	X	XXXX	XXXX	XXXX	XXXX	X	X	X	1
270000	30	XXAA	XXXX	XXXX	XXXX	XXXX	XXXX	XXXX	X	XXXX	XXXX	XXXX	XXXX	X	X	X	1
290000	08	XXAA	XXXX	XXXX	XXXX	XXXX	XXXX	XXXX	X	XXXX	XXXX	XXXX	XXXX	X	X	X	1
310000	08	AAAA	XXXX	XXXX	XXXX	XXXX	XXXX	XXXX	X	XXXX	XXXX	XXXX	XXXX	X	X	X	1
330000	28	AAAA	XXXX	XXXX	XXXX	XXXX	XXXX	XXXX	X	XXXX	XXXX	XXXX	XXXX	X	X	X	1
350000	28	AAAA	XXCC	XXXX	XXXX	XXCC	XXXX	XXXX	X	XXXX	XXXX	XXXX	XXXX	X	X	X	1
370000	18	AAAA	XXCC	XXXX	XXXX	XXCC	XXXX	XXXX	X	XXXX	XXXX	XXXX	XXXX	X	X	X	1
390000	18	AAAA	CCCC	XXXX	XXXX	CCCC	XXXX	XXXX	X	XXXX	XXXX	XXXX	XXXX	X	X	X	1
410000	38	AAAA	CCCC	XXXX	XXXX	CCCC	XXXX	XXXX	X	XXXX	XXXX	XXXX	XXXX	X	X	X	1
430000	38	AAAA	CCCC	XX35	XXXX	CCCC	XXXX	XXXX	X	XXXX	XXXX	XXXX	XXXX	X	X	X	1
450000	04	AAAA	CCCC	XX35	XXXX	CCCC	XXXX	XXXX	X	XXXX	XXXX	XXXX	XXXX	X	X	X	1
470000	04	AAAA	CCCC	5535	XXXX	CCCC	XXXX	XXXX	X	XXXX	XXXX	XXXX	XXXX	X	X	X	1
490000	24	AAAA	CCCC	5535	XXXX	CCCC	XXXX	XXXX	X	XXXX	XXXX	XXXX	XXXX	X	X	X	1
510000	24	AAAA	CCCC	5535	XX11	CCCC	XXXX	XXXX	X	XXXX	XXXX	XXXX	XXXX	X	X	X	1
530000	14	AAAA	CCCC	5535	XX11	CCCC	XXXX	XXXX	X	XXXX	XXXX	XXXX	XXXX	X	X	X	1
550000	14	AAAA	CCCC	5535	3311	CCCC	XXXX	XXXX	X	XXXX	XXXX	XXXX	XXXX	X	X	X	1
570000	34	AAAA	CCCC	5535	3311	AAAA	AACA	FFE0	X	FFE0	FFE0	XXXX	XXXX	1	0	X	0
590000	0C	AAAA	CCCC	5535	3311	AAAA	AACA	0020	1	0020	0020	001F	XXXX	0	1	1	0
610000	1C	AAAA	CCCC	5535	3311	CCCC	CCEE	FFDE	0	XXX?	FFDE	FFDF	XXXX	1	0	0	1
630000	3C	AAAA	CCCC	5535	3311	CCCC	CCEE	0022	1	XXX?	0022	FFDF	0021	0	1	1	1
650000	22	AAAA	CCCC	5535	3311	0020	0022	0002	1	XXX?	0002	FFDF	FFDD	0	1	1	1
670000	12	AAAA	CCCC	5535	3311	CCCC	0022	CCEE	0	XXXB	CCEE	FFDF	FFDD	1	0	0	1
690000	2A	AAAA	CCCC	5535	3311	CCCC	0022	CCEE	0	XX?6	CCEE	FFDF	FFDD	1	0	0	1
710000	1A	AAAA	CCCC	5535	3311	0040	0022	001E	1	001E	001E	FFDF	FFDD	0	1	1	1
730000	1A	AAAA	CCCC	5535	3311	003C	0022	001A	1	001A	001A	FFE1	FFDD	0	1	1	1
750000	1A	AAAA	CCCC	5535	3311	0034	0022	0012	1	0012	0012	FFE5	FFDD	0	1	1	1
770000	1A	AAAA	CCCC	5535	3311	0024	0022	0002	1	0002	0002	FFED	FFDD	0	1	1	1
790000	1A	AAAA	CCCC	5535	3311	0004	0022	FFE2	0	0004	FFE2	FFFD	FFDD	1	0	0	1
810000	1A	AAAA	CCCC	5535	3311	0008	0022	FFE6	0	0008	FFE6	FFFB	FFDD	1	0	0	1
830000	1A	AAAA	CCCC	5535	3311	0010	0022	FFEE	0	0010	FFEE	FFF7	FFDD	1	0	0	1
850000	1A	AAAA	CCCC	5535	3311	0020	0022	FFFE	0	0020	FFFE	FFEF	FFDD	1	0	0	1
870000	1A	AAAA	CCCC	5535	3311	0040	0022	001E	1	001E	001E	FFDF	FFDD	0	1	1	1
890000	1A	AAAA	CCCC	5535	3311	003C	0022	001A	1	001A	001A	FFE1	FFDD	0	1	1	1
910000	1A	AAAA	CCCC	5535	3311	0034	0022	0012	1	0012	0012	FFE5	FFDD	0	1	1	1
930000	1A	AAAA	CCCC	5535	3311	0024	0022	0002	1	0002	0002	FFED	FFDD	0	1	1	1

950000	1A	AAAA	CCCC	5535	3311	0004	0022	FFE2	0	0004	FFE2	FFFD	FFDD	1	0	0	1
970000	1A	AAAA	CCCC	5535	3311	0008	0022	FFE6	0	0008	FFE6	FFFB	FFDD	1	0	0	1
990000	1A	AAAA	CCCC	5535	3311	0010	0022	FFEE	0	0010	FFEE	FFF7	FFDD	1	0	0	1
1010000	1A	AAAA	CCCC	5535	3311	0020	0022	FFFE	0	0020	FFFE	FFEF	FFDD	1	0	0	1
1030000	3A	AAAA	CCCC	5535	3311	CCCC	0022	CCEE	0	F0F0	CCEE	FFDF	FFDD	1	0	0	1
1050000	06	AAAA	CCCC	5535	3311	CCCC	0022	CCEE	1	E1E0	CCEE	0F0F	FFDD	1	0	1	1
1070000	11	AAAA	CCCC	5535	3311	CCCC	8000	4CCC	1	C3C1	4CCC	0F0F	7FFF	0	1	1	1
1090000	26	AAAA	CCCC	5535	3311	F0F0	8000	70F0	1	8783	70F0	0F0F	7FFF	0	1	1	1
1110000	36	AAAA	CCCC	5535	3311	AAAA	0001	AAAB	1	0F07	AAAB	0F0F	8F0F	1	0	1	1
1130000	2E	AAAB	CCCC	5535	3311	CCCC	0001	CCCD	1	1E0F	CCCD	0F0F	8F0F	1	0	1	1
1150000	11	AAAB	CCCD	5535	3311	CCCD	70F0	3DBD	1	3C1F	3DBD	0F0F	8F0F	0	1	1	1
1170000	26	AAAB	CCCD	5535	3311	F0F0	70F0	61E0	1	783F	61E0	0F0F	8F0F	0	1	1	1
1190000	36	AAAB	CCCD	5535	3311	AAAB	0001	AAAC	1	F07F	AAAC	0F0F	9E1F	1	0	1	1
1210000	2E	AAAC	CCCD	5535	3311	CCCD	0001	CCCE	1	E0FF	CCCE	0F0F	9E1F	1	0	1	1
1230000	11	AAAC	CCCE	5535	3311	CCCE	61E0	2EAE	1	C1FF	2EAE	0F0F	9E1F	0	1	1	1
1250000	26	AAAC	CCCE	5535	3311	F0F0	61E0	52D0	1	83FF	52D0	0F0F	9E1F	0	1	1	1
1270000	36	AAAC	CCCE	5535	3311	AAAC	0001	AAAD	1	07FF	AAAD	0F0F	AD2F	1	0	1	1
1290000	2E	AAAD	CCCE	5535	3311	CCCE	0001	CCCF	1	0FFF	CCCF	0F0F	AD2F	1	0	1	1
1310000	11	AAAD	CCCF	5535	3311	CCCF	52D0	1F9F	1	1FFF	1F9F	0F0F	AD2F	0	1	1	1
1330000	26	AAAD	CCCF	5535	3311	F0F0	52D0	43C0	1	3FFF	43C0	0F0F	AD2F	0	1	1	1
1350000	36	AAAD	CCCF	5535	3311	AAAD	0001	AAAE	1	7FFF	AAAE	0F0F	BC3F	1	0	1	1
1370000	2E	AAAE	CCCF	5535	3311	CCCF	0001	CCD0	1	FFFF	CCD0	0F0F	BC3F	1	0	1	1
1390000	11	AAAE	CCD0	5535	3311	CCD0	43C0	1090	1	FFFF	1090	0F0F	BC3F	0	1	1	1
1410000	26	AAAE	CCD0	5535	3311	F0F0	43C0	34B0	1	FFFF	34B0	0F0F	BC3F	0	1	1	1
1430000	36	AAAE	CCD0	5535	3311	AAAE	0001	AAAF	1	FFFF	AAAF	0F0F	CB4F	1	0	1	1
1450000	2E	AAAF	CCD0	5535	3311	CCD0	0001	CCD1	1	FFFF	CCD1	0F0F	CB4F	1	0	1	1
1470000	11	AAAF	CCD1	5535	3311	CCD1	34B0	0181	1	FFFF	0181	0F0F	CB4F	0	1	1	1
1490000	26	AAAF	CCD1	5535	3311	F0F0	34B0	25A0	1	FFFF	25A0	0F0F	CB4F	0	1	1	1
1510000	36	AAAF	CCD1	5535	3311	AAAF	0001	AAB0	1	FFFF	AAB0	0F0F	DA5F	1	0	1	1
1530000	2E	AAB0	CCD1	5535	3311	CCD1	0001	CCD2	1	FFFF	CCD2	0F0F	DA5F	1	0	1	1
1550000	11	AAB0	CCD2	5535	3311	CCD2	25A0	F272	1	FFFF	F272	0F0F	DA5F	1	0	1	1
1570000	26	AAB0	CCD2	5535	3311	F0F0	25A0	1690	1	FFFF	1690	0F0F	DA5F	0	1	1	1
1590000	36	AAB0	CCD2	5535	3311	AAB0	0001	AAB1	1	FFFF	AAB1	0F0F	E96F	1	0	1	1
1610000	2E	AAB1	CCD2	5535	3311	CCD2	0001	CCD3	1	FFFF	CCD3	0F0F	E96F	1	0	1	1
1630000	11	AAB1	CCD3	5535	3311	CCD3	1690	E363	1	FFFF	E363	0F0F	E96F	1	0	1	1
1650000	26	AAB1	CCD3	5535	3311	F0F0	1690	0780	1	FFFF	0780	0F0F	E96F	0	1	1	1
1670000	36	AAB1	CCD3	5535	3311	AAB1	0001	AAB2	1	FFFF	AAB2	0F0F	F87F	1	0	1	1
1690000	2E	AAB2	CCD3	5535	3311	CCD3	0001	CCD4	1	FFFF	CCD4	0F0F	F87F	1	0	1	1
1710000	11	AAB2	CCD4	5535	3311	CCD4	0780	D454	1	FFFF	D454	0F0F	F87F	1	0	1	1
1730000	26	AAB2	CCD4	5535	3311	F0F0	0780	F870	1	FFFF	F870	0F0F	F87F	1	0	1	1
1750000	21	AAB2	CCD4	5535	3311	CCD4	0001	CCD5	1	FFFF	CCD5	0F0F	078F	1	0	1	1
1770000	11	AAB2	CCD5	5535	3311	CCD5	F870	C545	1	FFFF	C545	0F0F	078F	0	1	1	1
1790000	26	AAB2	CCD5	5535	3311	F0F0	F870	E960	1	FFFF	E960	0F0F	078F	0	1	1	1
1810000	36	AAB2	CCD5	5535	3311	AAB2	0001	AAB3	1	FFFF	AAB3	0F0F	169F	1	0	1	1
1830000	2E	AAB3	CCD5	5535	3311	CCD5	0001	CCD6	1	FFFF	CCD6	0F0F	169F	1	0	1	1
1850000	11	AAB3	CCD6	5535	3311	CCD6	E960	B636	1	FFFF	B636	0F0F	169F	0	1	1	1
1870000	26	AAB3	CCD6	5535	3311	F0F0	E960	DA50	1	FFFF	DA50	0F0F	169F	0	1	1	1
1890000	36	AAB3	CCD6	5535	3311	AAB3	0001	AAB4	1	FFFF	AAB4	0F0F	25AF	1	0	1	1
1910000	2E	AAB4	CCD6	5535	3311	CCD6	0001	CCD7	1	FFFF	CCD7	0F0F	25AF	1	0	1	1
1930000	11	AAB4	CCD7	5535	3311	CCD7	DA50	A727	1	FFFF	A727	0F0F	25AF	0	1	1	1
1950000	26	AAB4	CCD7	5535	3311	F0F0	DA50	CB40	1	FFFF	CB40	0F0F	25AF	0	1	1	1
1970000	36	AAB4	CCD7	5535	3311	AAB4	0001	AAB5	1	FFFF	AAB5	0F0F	34BF	1	0	1	1
1990000	2E	AAB5	CCD7	5535	3311	CCD7	0001	CCD8	1	FFFF	CCD8	0F0F	34BF	1	0	1	1
2010000	11	AAB5	CCD8	5535	3311	CCD8	CB40	9818	1	FFFF	9818	0F0F	34BF	0	1	1	1
2030000	26	AAB5	CCD8	5535	3311	F0F0	CB40	BC30	1	FFFF	BC30	0F0F	34BF	0	1	1	1
2050000	36	AAB5	CCD8	5535	3311	AAB5	0001	AAB6	1	FFFF	AAB6	0F0F	43CF	1	0	1	1
2070000	2E	AAB6	CCD8	5535	3311	CCD8	0001	CCD9	1	FFFF	CCD9	0F0F	43CF	1	0	1	1
2090000	11	AAB6	CCD9	5535	3311	CCD9	BC30	8909	1	FFFF	8909	0F0F	43CF	0	1	1	1
2110000	26	AAB6	CCD9	5535	3311	F0F0	BC30	AD20	1	FFFF	AD20	0F0F	43CF	0	1	1	1
2130000	36	AAB6	CCD9	5535	3311	AAB6	0001	AAB7	1	FFFF	AAB7	0F0F	52DF	1	0	1	1

2150000	2E	AAB7	CCD9	5535	3311	CCD9	0001	CCDA	1	FFFF	CCDA	OFOF	52DF	1	0	1	1
2170000	11	AAB7	CCDA	5535	3311	CCDA	AD20	79FA	1	FFFF	79FA	OFOF	52DF	0	1	1	1
2190000	26	AAB7	CCDA	5535	3311	FOFO	AD20	9E10	1	FFFF	9E10	OFOF	52DF	0	1	1	1
2210000	36	AAB7	CCDA	5535	3311	AAB7	0001	AAB8	1	FFFF	AAB8	OFOF	61EF	1	0	1	1
2230000	2E	AAB8	CCDA	5535	3311	CCDA	0001	CCDB	1	FFFF	CCDB	OFOF	61EF	1	0	1	1
2250000	11	AAB8	CCDB	5535	3311	CCDB	9E10	6AEB	1	FFFF	6AEB	OFOF	61EF	0	1	1	1
2270000	26	AAB8	CCDB	5535	3311	FOFO	9E10	8F00	1	FFFF	8F00	OFOF	61EF	0	1	1	1
2290000	36	AAB8	CCDB	5535	3311	AAB8	0001	AAB9	1	FFFF	AAB9	OFOF	70FF	1	0	1	1
2310000	2E	AAB9	CCDB	5535	3311	CCDB	0001	CCDC	1	FFFF	CCDC	OFOF	70FF	1	0	1	1
2330000	11	AAB9	CCDC	5535	3311	CCDC	8F00	5BDC	1	FFFF	5BDC	OFOF	70FF	0	1	1	1
2350000	26	AAB9	CCDC	5535	3311	FOFO	8F00	7FF0	1	FFFF	7FF0	OFOF	70FF	0	1	1	1
2370000	36	AAB9	CCDC	5535	3311	AAB9	0001	AABA	1	FFFF	AABA	OFOF	800F	1	0	1	1
2390000	2E	AABA	CCDC	5535	3311	CCDC	0001	CCDD	1	FFFF	CCDD	OFOF	800F	1	0	1	1
2410000	11	AABA	CCDD	5535	3311	CCDD	7FF0	4CCD	1	FFFF	4CCD	OFOF	800F	0	1	1	1
2430000	26	AABA	CCDD	5535	3311	FOFO	7FF0	70E0	1	FFFF	70E0	OFOF	800F	0	1	1	1
2450000	36	AABA	CCDD	5535	3311	AABA	0001	AABB	1	FFFF	AABB	OFOF	8F1F	1	0	1	1
2470000	2E	AABB	CCDD	5535	3311	CCDD	0001	CCDE	1	FFFF	CCDE	OFOF	8F1F	1	0	1	1
2490000	11	AABB	CCDE	5535	3311	CCDE	70E0	3DBE	1	FFFF	3DBE	OFOF	8F1F	0	1	1	1
2510000	26	AABB	CCDE	5535	3311	FOFO	70E0	61D0	1	FFFF	61D0	OFOF	8F1F	0	1	1	1
2530000	36	AABB	CCDE	5535	3311	AABB	0001	AABC	1	FFFF	AABC	OFOF	9E2F	1	0	1	1
2550000	2E	AABC	CCDE	5535	3311	CCDE	0001	CCDF	1	FFFF	CCDF	OFOF	9E2F	1	0	1	1
2570000	11	AABC	CCDF	5535	3311	CCDF	61D0	2EAF	1	FFFF	2EAF	OFOF	9E2F	0	1	1	1
2590000	26	AABC	CCDF	5535	3311	FOFO	61D0	52C0	1	FFFF	52C0	OFOF	9E2F	0	1	1	1
2610000	36	AABC	CCDF	5535	3311	AABC	0001	AABD	1	FFFF	AABD	OFOF	AD3F	1	0	1	1
2630000	2E	AABD	CCDF	5535	3311	CCDF	0001	CCE0	1	FFFF	CCE0	OFOF	AD3F	1	0	1	1
2650000	11	AABD	CCE0	5535	3311	CCE0	52C0	1FA0	1	FFFF	1FA0	OFOF	AD3F	0	1	1	1
2670000	26	AABD	CCE0	5535	3311	FOFO	52C0	43B0	1	FFFF	43B0	OFOF	AD3F	0	1	1	1
2690000	36	AABD	CCE0	5535	3311	AABD	0001	AABE	1	FFFF	AABE	OFOF	BC4F	1	0	1	1
2710000	2E	AABE	CCE0	5535	3311	CCE0	0001	CCE1	1	FFFF	CCE1	OFOF	BC4F	1	0	1	1
2730000	11	AABE	CCE1	5535	3311	CCE1	43B0	1091	1	FFFF	1091	OFOF	BC4F	0	1	1	1
2750000	26	AABE	CCE1	5535	3311	FOFO	43B0	34A0	1	FFFF	34A0	OFOF	BC4F	0	1	1	1
2770000	36	AABE	CCE1	5535	3311	AABE	0001	AABF	1	FFFF	AABF	OFOF	CB5F	1	0	1	1
2790000	2E	AABF	CCE1	5535	3311	CCE1	0001	CCE2	1	FFFF	CCE2	OFOF	CB5F	1	0	1	1
2810000	11	AABF	CCE2	5535	3311	CCE2	34A0	0182	1	FFFF	0182	OFOF	CB5F	0	1	1	1
2830000	26	AABF	CCE2	5535	3311	FOFO	34A0	2590	1	FFFF	2590	OFOF	CB5F	0	1	1	1
2850000	36	AABF	CCE2	5535	3311	AABF	0001	AAC0	1	FFFF	AAC0	OFOF	DA6F	1	0	1	1
2870000	2E	AAC0	CCE2	5535	3311	CCE2	0001	CCE3	1	FFFF	CCE3	OFOF	DA6F	1	0	1	1
2890000	11	AAC0	CCE3	5535	3311	CCE3	2590	F273	1	FFFF	F273	OFOF	DA6F	1	0	1	1
2910000	26	AAC0	CCE3	5535	3311	FOFO	2590	1680	1	FFFF	1680	OFOF	DA6F	0	1	1	1
2930000	36	AAC0	CCE3	5535	3311	AAC0	0001	AAC1	1	FFFF	AAC1	OFOF	E97F	1	0	1	1
2950000	2E	AAC1	CCE3	5535	3311	CCE3	0001	CCE4	1	FFFF	CCE4	OFOF	E97F	1	0	1	1
2970000	11	AAC1	CCE4	5535	3311	CCE4	1680	E364	1	FFFF	E364	OFOF	E97F	1	0	1	1
2990000	26	AAC1	CCE4	5535	3311	FOFO	1680	0770	1	FFFF	0770	OFOF	E97F	0	1	1	1
3010000	36	AAC1	CCE4	5535	3311	AAC1	0001	AAC2	1	FFFF	AAC2	OFOF	F88F	1	0	1	1
3030000	2E	AAC2	CCE4	5535	3311	CCE4	0001	CCE5	1	FFFF	CCE5	OFOF	F88F	1	0	1	1
3050000	11	AAC2	CCE5	5535	3311	CCE5	0770	D455	1	FFFF	D455	OFOF	F88F	1	0	1	1
3070000	26	AAC2	CCE5	5535	3311	FOFO	0770	F860	1	FFFF	F860	OFOF	F88F	1	0	1	1
3090000	21	AAC2	CCE5	5535	3311	CCE5	0001	CCE6	1	FFFF	CCE6	OFOF	079F	1	0	1	1
3110000	11	AAC2	CCE6	5535	3311	CCE6	F860	C546	1	FFFF	C546	OFOF	079F	0	1	1	1
3130000	26	AAC2	CCE6	5535	3311	FOFO	F860	E950	1	FFFF	E950	OFOF	079F	0	1	1	1
3150000	36	AAC2	CCE6	5535	3311	AAC2	0001	AAC3	1	FFFF	AAC3	OFOF	16AF	1	0	1	1
3170000	2E	AAC3	CCE6	5535	3311	CCE6	0001	CCE7	1	FFFF	CCE7	OFOF	16AF	1	0	1	1
3190000	11	AAC3	CCE7	5535	3311	CCE7	E950	B637	1	FFFF	B637	OFOF	16AF	0	1	1	1
3210000	26	AAC3	CCE7	5535	3311	FOFO	E950	DA40	1	FFFF	DA40	OFOF	16AF	0	1	1	1
3230000	36	AAC3	CCE7	5535	3311	AAC3	0001	AAC4	1	FFFF	AAC4	OFOF	25BF	1	0	1	1
3250000	2E	AAC4	CCE7	5535	3311	CCE7	0001	CCE8	1	FFFF	CCE8	OFOF	25BF	1	0	1	1
3270000	11	AAC4	CCE8	5535	3311	CCE8	DA40	A728	1	FFFF	A728	OFOF	25BF	0	1	1	1
3290000	26	AAC4	CCE8	5535	3311	FOFO	DA40	CB30	1	FFFF	CB30	OFOF	25BF	0	1	1	1
3310000	36	AAC4	CCE8	5535	3311	AAC4	0001	AAC5	1	FFFF	AAC5	OFOF	34CF	1	0	1	1
3330000	2E	AAC5	CCE8	5535	3311	CCE8	0001	CCE9	1	FFFF	CCE9	OFOF	34CF	1	0	1	1

3350000	11	AAC5	CCE9	5535	3311	CCE9	CB30	9819	1	FFFF	9819	OFOF	34CF	0	1	1	1
3370000	26	AAC5	CCE9	5535	3311	FOFO	CB30	BC20	1	FFFF	BC20	OFOF	34CF	0	1	1	1
3390000	36	AAC5	CCE9	5535	3311	AAC5	0001	AAC6	1	FFFF	AAC6	OFOF	43DF	1	0	1	1
3410000	2E	AAC6	CCE9	5535	3311	CCE9	0001	CCEA	1	FFFF	CCEA	OFOF	43DF	1	0	1	1
3430000	11	AAC6	CCEA	5535	3311	CCEA	BC20	890A	1	FFFF	890A	OFOF	43DF	0	1	1	1
3450000	26	AAC6	CCEA	5535	3311	FOFO	BC20	AD10	1	FFFF	AD10	OFOF	43DF	0	1	1	1
3470000	36	AAC6	CCEA	5535	3311	AAC6	0001	AAC7	1	FFFF	AAC7	OFOF	52EF	1	0	1	1
3490000	2E	AAC7	CCEA	5535	3311	CCEA	0001	CCEB	1	FFFF	CCEB	OFOF	52EF	1	0	1	1
3510000	11	AAC7	CCEB	5535	3311	CCEB	AD10	79FB	1	FFFF	79FB	OFOF	52EF	0	1	1	1
3530000	26	AAC7	CCEB	5535	3311	FOFO	AD10	9E00	1	FFFF	9E00	OFOF	52EF	0	1	1	1
3550000	36	AAC7	CCEB	5535	3311	AAC7	0001	AAC8	1	FFFF	AAC8	OFOF	61FF	1	0	1	1
3570000	2E	AAC8	CCEB	5535	3311	CCEB	0001	CCEC	1	FFFF	CCEC	OFOF	61FF	1	0	1	1
3590000	11	AAC8	CCEC	5535	3311	CCEC	9E00	6AEC	1	FFFF	6AEC	OFOF	61FF	0	1	1	1
3610000	26	AAC8	CCEC	5535	3311	FOFO	9E00	8EFO	1	FFFF	8EFO	OFOF	61FF	0	1	1	1
3630000	36	AAC8	CCEC	5535	3311	AAC8	0001	AAC9	1	FFFF	AAC9	OFOF	710F	1	0	1	1
3650000	2E	AAC9	CCEC	5535	3311	CCEC	0001	CCED	1	FFFF	CCED	OFOF	710F	1	0	1	1
3670000	11	AAC9	CCED	5535	3311	CCED	8EFO	5BDD	1	FFFF	5BDD	OFOF	710F	0	1	1	1
3690000	26	AAC9	CCED	5535	3311	FOFO	8EFO	7FEO	1	FFFF	7FEO	OFOF	710F	0	1	1	1
3710000	36	AAC9	CCED	5535	3311	AAC9	0001	AACA	1	FFFF	AACA	OFOF	801F	1	0	1	1
3730000	2E	AACA	CCED	5535	3311	CCED	0001	CCEE	1	FFFF	CCEE	OFOF	801F	1	0	1	1
3750000	11	AACA	CCEE	5535	3311	CCEE	7FEO	4CCE	1	FFFF	4CCE	OFOF	801F	0	1	1	1
3770000	00	AACA	CCEE	5535	3311	CCEE	7FEO	4CCE	1	FFFF	4CCE	OFOF	801F	0	1	1	1

NODE	INST	XMEMR	MEMR	XBG	XBACK	XBR	XBREQ	XRESET	XRDR	XS1	XS9	XS10	XINST
INDX	HEX	HEX	HEX	HEX	HEX	HEX	HEX	HEX	HEX	HEX	HEX	HEX	HEX
SIM	1	1	1	1	1	1	1	1	1	1	1	1	1
I/O													
10000	00	1	1	1	1	1	0	0	1	1	1	1	00
30000	00	1	1	1	1	1	0	1	1	1	1	1	00
50000	00	1	0	1	1	1	0	1	1	1	1	1	00
70000	00	1	0	1	0	1	0	1	1	1	1	1	00
90000	20	1	0	1	0	1	1	1	0	0	1	1	20
110000	20	1	0	1	1	1	1	1	0	0	1	1	20
150000	20	0	0	1	1	1	1	1	0	0	1	1	20
170000	20	1	0	1	1	1	1	1	0	0	1	1	20
190000	20	0	1	1	1	1	1	1	0	0	1	1	20
210000	10	1	0	1	1	1	1	1	0	1	1	1	10
230000	10	0	1	1	1	1	1	1	0	1	1	1	10
250000	30	1	0	1	1	1	1	1	0	1	1	1	30
270000	30	0	1	1	1	1	1	1	0	1	1	1	30
290000	08	1	0	1	1	1	1	1	0	1	1	1	08
310000	08	0	1	1	1	1	1	1	0	1	1	1	08
330000	28	1	0	1	1	1	1	1	0	1	1	1	28
350000	28	0	1	1	1	1	1	1	0	1	1	1	28
370000	18	1	0	1	1	1	1	1	0	1	1	1	18
390000	18	0	1	1	1	1	1	1	0	1	1	1	18
410000	38	1	0	1	1	1	1	1	0	1	1	1	38
430000	38	0	1	1	1	1	1	1	0	1	1	1	38
450000	04	1	0	1	1	1	1	1	0	1	1	1	04
470000	04	0	1	1	1	1	1	1	0	1	1	1	04
490000	24	1	0	1	1	1	1	1	0	1	0	1	24
510000	24	0	1	1	1	1	1	1	0	1	0	1	24
530000	14	1	0	1	1	1	1	1	0	1	1	0	14
550000	14	1	1	1	1	1	1	1	0	1	1	0	14
570000	34	1	0	1	1	1	1	1	1	1	1	1	34
590000	0C	1	0	1	1	1	1	1	1	1	1	1	0C
610000	1C	1	0	1	1	1	1	1	1	1	1	1	1C
630000	3C	1	0	1	1	1	1	1	1	1	1	1	3C
650000	22	1	0	1	1	1	1	1	1	1	1	1	22
670000	12	1	0	1	1	1	1	1	1	1	1	1	12
690000	2A	1	0	1	1	1	1	1	1	1	1	1	2A
710000	1A	1	0	1	1	1	1	1	1	1	1	1	1A
1010000	1A	1	0	0	1	1	1	1	1	1	1	1	1A
1030000	3A	1	0	0	1	1	1	1	1	1	1	1	3A
1050000	06	1	0	0	1	1	1	1	1	1	1	1	06

1070000	11	1	0	0	1	0	1	1	1	1	1	1	11
1090000	26	1	0	0	1	1	1	1	1	1	1	1	26
1110000	36	1	0	0	1	1	1	1	1	1	1	1	36
1130000	2E	1	0	0	1	1	1	1	1	1	1	1	2E
1150000	11	1	0	0	1	0	1	1	1	1	1	1	11
1170000	26	1	0	0	1	1	1	1	1	1	1	1	26
1190000	36	1	0	0	1	1	1	1	1	1	1	1	36
1210000	2E	1	0	0	1	1	1	1	1	1	1	1	2E
1230000	11	1	0	0	1	0	1	1	1	1	1	1	11
1250000	26	1	0	0	1	1	1	1	1	1	1	1	26
1270000	36	1	0	0	1	1	1	1	1	1	1	1	36
1290000	2E	1	0	0	1	1	1	1	1	1	1	1	2E
1310000	11	1	0	0	1	0	1	1	1	1	1	1	11
1330000	26	1	0	0	1	1	1	1	1	1	1	1	26
1350000	36	1	0	0	1	1	1	1	1	1	1	1	36
1370000	2E	1	0	0	1	1	1	1	1	1	1	1	2E
1390000	11	1	0	0	1	0	1	1	1	1	1	1	11
1410000	26	1	0	0	1	1	1	1	1	1	1	1	26
1430000	36	1	0	0	1	1	1	1	1	1	1	1	36
1450000	2E	1	0	0	1	1	1	1	1	1	1	1	2E
1470000	11	1	0	0	1	0	1	1	1	1	1	1	11
1490000	26	1	0	0	1	1	1	1	1	1	1	1	26
1510000	36	1	0	0	1	1	1	1	1	1	1	1	36
1530000	2E	1	0	0	1	1	1	1	1	1	1	1	2E
1550000	11	1	0	0	1	0	1	1	1	1	1	1	11
1570000	26	1	0	0	1	1	1	1	1	1	1	1	26
1590000	36	1	0	0	1	1	1	1	1	1	1	1	36
1610000	2E	1	0	0	1	1	1	1	1	1	1	1	2E
1630000	11	1	0	0	1	0	1	1	1	1	1	1	11
1650000	26	1	0	0	1	1	1	1	1	1	1	1	26
1670000	36	1	0	0	1	1	1	1	1	1	1	1	36
1690000	2E	1	0	0	1	1	1	1	1	1	1	1	2E
1710000	11	1	0	0	1	0	1	1	1	1	1	1	11
1730000	26	1	0	0	1	1	1	1	1	1	1	1	26
1750000	21	1	0	0	1	1	1	1	1	1	1	1	21
1770000	11	1	0	0	1	0	1	1	1	1	1	1	11
1790000	26	1	0	0	1	1	1	1	1	1	1	1	26
1810000	36	1	0	0	1	1	1	1	1	1	1	1	36
1830000	2E	1	0	0	1	1	1	1	1	1	1	1	2E
1850000	11	1	0	0	1	0	1	1	1	1	1	1	11
1870000	26	1	0	0	1	1	1	1	1	1	1	1	26
1890000	36	1	0	0	1	1	1	1	1	1	1	1	36
1910000	2E	1	0	0	1	1	1	1	1	1	1	1	2E
1930000	11	1	0	0	1	0	1	1	1	1	1	1	11
1950000	26	1	0	0	1	1	1	1	1	1	1	1	26
1970000	36	1	0	0	1	1	1	1	1	1	1	1	36
1990000	2E	1	0	0	1	1	1	1	1	1	1	1	2E
2010000	11	1	0	0	1	0	1	1	1	1	1	1	11
2030000	26	1	0	0	1	1	1	1	1	1	1	1	26
2050000	36	1	0	0	1	1	1	1	1	1	1	1	36
2070000	2E	1	0	0	1	1	1	1	1	1	1	1	2E
2090000	11	1	0	0	1	0	1	1	1	1	1	1	11
2110000	26	1	0	0	1	1	1	1	1	1	1	1	26
2130000	36	1	0	0	1	1	1	1	1	1	1	1	36
2150000	2E	1	0	0	1	1	1	1	1	1	1	1	2E
2170000	11	1	0	0	1	0	1	1	1	1	1	1	11
2190000	26	1	0	0	1	1	1	1	1	1	1	1	26
2210000	36	1	0	0	1	1	1	1	1	1	1	1	36
2230000	2E	1	0	0	1	1	1	1	1	1	1	1	2E
2250000	11	1	0	0	1	0	1	1	1	1	1	1	11

2270000	26	1	0	0	1	1	1	1	1	1	1	1	26
2290000	36	1	0	0	1	1	1	1	1	1	1	1	36
2310000	2E	1	0	0	1	1	1	1	1	1	1	1	2E
2330000	11	1	0	0	1	0	1	1	1	1	1	1	11
2350000	26	1	0	0	1	1	1	1	1	1	1	1	26
2370000	36	1	0	0	1	1	1	1	1	1	1	1	36
2390000	2E	1	0	0	1	1	1	1	1	1	1	1	2E
2410000	11	1	0	0	1	0	1	1	1	1	1	1	11
2430000	26	1	0	0	1	1	1	1	1	1	1	1	26
2450000	36	1	0	0	1	1	1	1	1	1	1	1	36
2470000	2E	1	0	0	1	1	1	1	1	1	1	1	2E
2490000	11	1	0	0	1	0	1	1	1	1	1	1	11
2510000	26	1	0	0	1	1	1	1	1	1	1	1	26
2530000	36	1	0	0	1	1	1	1	1	1	1	1	36
2550000	2E	1	0	0	1	1	1	1	1	1	1	1	2E
2570000	11	1	0	0	1	0	1	1	1	1	1	1	11
2590000	26	1	0	0	1	1	1	1	1	1	1	1	26
2610000	36	1	0	0	1	1	1	1	1	1	1	1	36
2630000	2E	1	0	0	1	1	1	1	1	1	1	1	2E
2650000	11	1	0	0	1	0	1	1	1	1	1	1	11
2670000	26	1	0	0	1	1	1	1	1	1	1	1	26
2690000	36	1	0	0	1	1	1	1	1	1	1	1	36
2710000	2E	1	0	0	1	1	1	1	1	1	1	1	2E
2730000	11	1	0	0	1	0	1	1	1	1	1	1	11
2750000	26	1	0	0	1	1	1	1	1	1	1	1	26
2770000	36	1	0	0	1	1	1	1	1	1	1	1	36
2790000	2E	1	0	0	1	1	1	1	1	1	1	1	2E
2810000	11	1	0	0	1	0	1	1	1	1	1	1	11
2830000	26	1	0	0	1	1	1	1	1	1	1	1	26
2850000	36	1	0	0	1	1	1	1	1	1	1	1	36
2870000	2E	1	0	0	1	1	1	1	1	1	1	1	2E
2890000	11	1	0	0	1	0	1	1	1	1	1	1	11
2910000	26	1	0	0	1	1	1	1	1	1	1	1	26
2930000	36	1	0	0	1	1	1	1	1	1	1	1	36
2950000	2E	1	0	0	1	1	1	1	1	1	1	1	2E
2970000	11	1	0	0	1	0	1	1	1	1	1	1	11
2990000	26	1	0	0	1	1	1	1	1	1	1	1	26
3010000	36	1	0	0	1	1	1	1	1	1	1	1	36
3030000	2E	1	0	0	1	1	1	1	1	1	1	1	2E
3050000	11	1	0	0	1	0	1	1	1	1	1	1	11
3070000	26	1	0	0	1	1	1	1	1	1	1	1	26
3090000	21	1	0	0	1	1	1	1	1	1	1	1	21
3110000	11	1	0	0	1	0	1	1	1	1	1	1	11
3130000	26	1	0	0	1	1	1	1	1	1	1	1	26
3150000	36	1	0	0	1	1	1	1	1	1	1	1	36
3170000	2E	1	0	0	1	1	1	1	1	1	1	1	2E
3190000	11	1	0	0	1	0	1	1	1	1	1	1	11
3210000	26	1	0	0	1	1	1	1	1	1	1	1	26
3230000	36	1	0	0	1	1	1	1	1	1	1	1	36
3250000	2E	1	0	0	1	1	1	1	1	1	1	1	2E
3270000	11	1	0	0	1	0	1	1	1	1	1	1	11
3290000	26	1	0	0	1	1	1	1	1	1	1	1	26
3310000	36	1	0	0	1	1	1	1	1	1	1	1	36
3330000	2E	1	0	0	1	1	1	1	1	1	1	1	2E
3350000	11	1	0	0	1	0	1	1	1	1	1	1	11
3370000	26	1	0	0	1	1	1	1	1	1	1	1	26
3390000	36	1	0	0	1	1	1	1	1	1	1	1	36
3410000	2E	1	0	0	1	1	1	1	1	1	1	1	2E
3430000	11	1	0	0	1	0	1	1	1	1	1	1	11
3450000	26	1	0	0	1	1	1	1	1	1	1	1	26

3470000	36	1	0	0	1	1	1	1	1	1	1	36
3490000	2E	1	0	0	1	1	1	1	1	1	1	2E
3510000	11	1	0	0	1	0	1	1	1	1	1	11
3530000	26	1	0	0	1	1	1	1	1	1	1	26
3550000	36	1	0	0	1	1	1	1	1	1	1	36
3570000	2E	1	0	0	1	1	1	1	1	1	1	2E
3590000	11	1	0	0	1	0	1	1	1	1	1	11
3610000	26	1	0	0	1	1	1	1	1	1	1	26
3630000	36	1	0	0	1	1	1	1	1	1	1	36
3650000	2E	1	0	0	1	1	1	1	1	1	1	2E
3670000	11	1	0	0	1	0	1	1	1	1	1	11
3690000	26	1	0	0	1	1	1	1	1	1	1	26
3710000	36	1	0	0	1	1	1	1	1	1	1	36
3730000	2E	1	0	0	1	1	1	1	1	1	1	2E
3750000	11	1	0	0	1	0	1	1	1	1	1	11
3770000	00	1	0	0	1	1	0	1	1	1	1	00

NODE	INST	BR	RXY	FIVEB	SRA	DIV	ABBA	BA2	AA1	RB	YA22	YA11	SINY	SINX	SENY	SENX	SMAN	SUBA1	SUBA2	LA	LB	RA	X1A1	CLEARFF	LX1L	LX1U	LY1L	LY1U	LX2L	LX2U	LY2L	LY2U	BREQ	RDR	NXRA
INDX	HEX	HEX	HEX	HEX	HEX	HEX	HEX	HEX	HEX	HEX	HEX	HEX	HEX	HEX	HEX	HEX	HEX	HEX	HEX	HEX	HEX	HEX	HEX	HEX	HEX	HEX	HEX	HEX	HEX	HEX	HEX	HEX	HEX	HEX	HEX
SIM.	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
I/O																																			
10000	00	0	0	0	1	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1
90000	20	0	0	0	1	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
210000	10	0	0	0	1	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
250000	30	0	0	0	1	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	1
290000	08	0	0	0	1	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	1
330000	28	0	0	0	1	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	1
370000	18	0	0	0	1	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	1
410000	38	0	0	0	1	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	1
450000	04	0	0	0	1	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	1
490000	24	0	0	0	1	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1
530000	14	0	0	0	1	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1
570000	34	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	1	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0
590000	0C	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	1
610000	1C	0	0	0	1	0	0	0	0	1	1	1	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1
630000	3C	0	0	0	1	0	0	0	0	1	1	1	1	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1
650000	22	0	0	0	1	0	0	1	1	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
670000	12	0	0	0	1	0	0	1	0	1	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
690000	2A	0	0	0	1	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
710000	1A	0	0	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1
730000	1A	0	0	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1
790000	1A	0	0	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
870000	1A	0	0	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1
890000	1A	0	0	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1
950000	1A	0	0	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
1030000	3A	0	0	0	1	0	0	1	0	1	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
1050000	06	0	0	1	1	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
1070000	11	1	0	0	1	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
1090000	26	0	0	0	1	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1
1110000	36	0	1	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	1
1130000	2E	0	1	0	1	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	1
1150000	11	1	0	0	1	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
1170000	26	0	0	0	1	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1
1190000	36	0	1	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	1
1210000	2E	0	1	0	1	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	1
1230000	11	1	0	0	1	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
1250000	26	0	0	0	1	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1
1270000	36	0	1	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	1
1290000	2E	0	1	0	1	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	1

1310000	11	1	0	0	1	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
1330000	26	0	0	0	1	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1
1350000	36	0	1	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	1
1370000	2E	0	1	0	1	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1
1390000	11	1	0	0	1	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
1410000	26	0	0	0	1	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1
1430000	36	0	1	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	1
1450000	2E	0	1	0	1	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1
1470000	11	1	0	0	1	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
1490000	26	0	0	0	1	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1
1510000	36	0	1	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	1
1530000	2E	0	1	0	1	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1
1550000	11	1	0	0	1	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
1570000	26	0	0	0	1	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1
1590000	36	0	1	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	1
1610000	2E	0	1	0	1	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1
1630000	11	1	0	0	1	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
1650000	26	0	0	0	1	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1
1670000	36	0	1	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	1
1690000	2E	0	1	0	1	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1
1710000	11	1	0	0	1	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
1730000	26	0	0	0	1	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1
1750000	21	0	1	0	1	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1
1770000	11	1	0	0	1	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
1790000	26	0	0	0	1	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1
1810000	36	0	1	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	1
1830000	2E	0	1	0	1	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1
1850000	11	1	0	0	1	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
1870000	26	0	0	0	1	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1
1890000	36	0	1	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	1
1910000	2E	0	1	0	1	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1
1930000	11	1	0	0	1	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
1950000	26	0	0	0	1	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1
1970000	36	0	1	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	1
1990000	2E	0	1	0	1	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1
2010000	11	1	0	0	1	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
2030000	26	0	0	0	1	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1
2050000	36	0	1	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	1
2070000	2E	0	1	0	1	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1
2090000	11	1	0	0	1	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
2110000	26	0	0	0	1	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1
2130000	36	0	1	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	1
2150000	2E	0	1	0	1	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1
2170000	11	1	0	0	1	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
2190000	26	0	0	0	1	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1
2210000	36	0	1	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	1
2230000	2E	0	1	0	1	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1
2250000	11	1	0	0	1	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
2270000	26	0	0	0	1	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1
2290000	36	0	1	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	1
2310000	2E	0	1	0	1	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1
2330000	11	1	0	0	1	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
2350000	26	0	0	0	1	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1
2370000	36	0	1	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	1
2390000	2E	0	1	0	1	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1
2410000	11	1	0	0	1	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
2430000	26	0	0	0	1	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1
2450000	36	0	1	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	1
2470000	2E	0	1	0	1	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1
2490000	11	1	0	0	1	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

2510000	26	0	0	0	1	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1
2530000	36	0	1	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	1
2550000	2E	0	1	0	1	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1
2570000	11	1	0	0	1	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
2590000	26	0	0	0	1	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1
2610000	36	0	1	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	1
2630000	2E	0	1	0	1	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1
2650000	11	1	0	0	1	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
2670000	26	0	0	0	1	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1
2690000	36	0	1	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	1
2710000	2E	0	1	0	1	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1
2730000	11	1	0	0	1	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
2750000	26	0	0	0	1	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1
2770000	36	0	1	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	1
2790000	2E	0	1	0	1	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1
2810000	11	1	0	0	1	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
2830000	26	0	0	0	1	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1
2850000	36	0	1	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	1
2870000	2E	0	1	0	1	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1
2890000	11	1	0	0	1	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
2910000	26	0	0	0	1	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1
2930000	36	0	1	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	1
2950000	2E	0	1	0	1	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1
2970000	11	1	0	0	1	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
2990000	26	0	0	0	1	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1
3010000	36	0	1	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	1
3030000	2E	0	1	0	1	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1
3050000	11	1	0	0	1	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
3070000	26	0	0	0	1	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1
3090000	21	0	1	0	1	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1
3110000	11	1	0	0	1	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
3130000	26	0	0	0	1	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1
3150000	36	0	1	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	1
3170000	2E	0	1	0	1	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1
3190000	11	1	0	0	1	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
3210000	26	0	0	0	1	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1
3230000	36	0	1	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	1
3250000	2E	0	1	0	1	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1
3270000	11	1	0	0	1	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
3290000	26	0	0	0	1	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1
3310000	36	0	1	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	1
3330000	2E	0	1	0	1	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1
3350000	11	1	0	0	1	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
3370000	26	0	0	0	1	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1
3390000	36	0	1	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	1
3410000	2E	0	1	0	1	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1
3430000	11	1	0	0	1	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
3450000	26	0	0	0	1	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1
3470000	36	0	1	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	1
3490000	2E	0	1	0	1	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1
3510000	11	1	0	0	1	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
3530000	26	0	0	0	1	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1
3550000	36	0	1	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	1
3570000	2E	0	1	0	1	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1
3590000	11	1	0	0	1	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
3610000	26	0	0	0	1	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1
3630000	36	0	1	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	1
3650000	2E	0	1	0	1	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1
3670000	11	1	0	0	1	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
3690000	26	0	0	0	1	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1

3710000	36	0	1	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	1
3730000	2E	0	1	0	1	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1
3750000	11	1	0	0	1	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
3770000	00	0	0	0	1	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

```
options
        ALL
        PERIOD=200ns
        XPROP
        TMAX=2000
        NPARAL=2
!        NPARAL=1
        SIM(1)=MAX
        SIM(2)=MIN
!faults
!  clasp
!  AUTO
!        STROBE

inputs

XCLOCK=Q
XRESET=0>1,1>800


;1
XBACK=1>3,0>5,$
   1>200,$
;2
      1>+3,0>+2,$
   1>400,$
;3
      1>+3,0>+2,$
   1>600,$
;4
      1>+3,0>+2,$
   1>800,$
;5
      1>+3,0>+2,$
   1>1000,$
;6
      1>+3,0>+2,$
   1>1200,$
;7
      1>+3,0>+2,$
   1>1400,$
;8
      1>+3,0>+2,$
   1>1600,$
;9
      1>+3,0>+2,$
   1>1800,$
;10
      1>+3,0>+2,$
   1>2000



;1
XBG=  1>50,$
   0>200,$
;2
      1>+50,$
     0>+92,$
     1>+7,$
   0>400,$
;3
      1>+50,$
   0>600,$
;4
      1>+50,$
   0>800,$
;5
      1>+50,$
   0>1000,$
;6
      1>+50,$
   0>1200,$
;7
      1>+50,$
        0>+9,$
        1>+3,$
   0>1400,$
;8
      1>+50,$
   0>1600,$
;9
      1>+50,$
   0>1800,$
;10
```

```
      1>+50,$
   0>2000

XMEMR=  1>7,  0>8,     1>9,  0>10,$
        1>11, 0>12,    1>13, 0>14,    1>15, 0>16,    1>17, 0>18, $
        1>19, 0>20,    1>21, 0>22,    1>23, 0>24,    1>25, 0>26, $
     1>200,$
        1>+7,  0>+1,     1>+1,  0>+1,$
        1>+1, 0>+1,    1>+1, 0>+1,    1>+1, 0>+1,    1>+1, 0>+1, $
        1>+1, 0>+1,    1>+1, 0>+1,    1>+1, 0>+1,    1>+1, 0>+1, $
      1>400,$
        1>+7,  0>+1,     1>+1,  0>+1,$
        1>+1, 0>+1,    1>+1, 0>+1,    1>+1, 0>+1,    1>+1, 0>+1, $
        1>+1, 0>+1,    1>+1, 0>+1,    1>+1, 0>+1,    1>+1, 0>+1, $
      1>600,$
        1>+7,  0>+1,     1>+1,  0>+1,$
        1>+1, 0>+1,    1>+1, 0>+1,    1>+1, 0>+1,    1>+1, 0>+1, $
        1>+1, 0>+1,    1>+1, 0>+1,    1>+1, 0>+1,    1>+1, 0>+1, $
      1>800,$
        1>+7,  0>+1,     1>+1,  0>+1,$
        1>+1, 0>+1,    1>+1, 0>+1,    1>+1, 0>+1,    1>+1, 0>+1, $
        1>+1, 0>+1,    1>+1, 0>+1,    1>+1, 0>+1,    1>+1, 0>+1, $
      1>1000,$
        1>+7,  0>+1,     1>+1,  0>+1,$
        1>+1, 0>+1,    1>+1, 0>+1,    1>+1, 0>+1,    1>+1, 0>+1, $
        1>+1, 0>+1,    1>+1, 0>+1,    1>+1, 0>+1,    1>+1, 0>+1, $
      1>1200,$
        1>+7,  0>+1,     1>+1,  0>+1,$
        1>+1, 0>+1,    1>+1, 0>+1,    1>+1, 0>+1,    1>+1, 0>+1, $
        1>+1, 0>+1,    1>+1, 0>+1,    1>+1, 0>+1,    1>+1, 0>+1, $
      1>1400,$
        1>+7,  0>+1,     1>+1,  0>+1,$
        1>+1, 0>+1,    1>+1, 0>+1,    1>+1, 0>+1,    1>+1, 0>+1, $
        1>+1, 0>+1,    1>+1, 0>+1,    1>+1, 0>+1,    1>+1, 0>+1, $
      1>1600,$
        1>+7,  0>+1,     1>+1,  0>+1,$
        1>+1, 0>+1,    1>+1, 0>+1,    1>+1, 0>+1,    1>+1, 0>+1, $
        1>+1, 0>+1,    1>+1, 0>+1,    1>+1, 0>+1,    1>+1, 0>+1, $
      1>1800,$
        1>+7,  0>+1,     1>+1,  0>+1,$
        1>+1, 0>+1,    1>+1, 0>+1,    1>+1, 0>+1,    1>+1, 0>+1, $
        1>+1, 0>+1,    1>+1, 0>+1,    1>+1, 0>+1,    1>+1, 0>+1, $
      1>2000

;1
DATAEX<7:0>/HEX=00>11,  AA >13, AA >15,  CC >17, CC >19,  CA >21, AA >23,  EE >25, CC >27,
;
;  co-ord          (   x1l     x1u   ,  y1l     y1u )  ( x2l     x2u   ,   y2l     y2u   )
      00>200,$

;2
                00>+11,  55 >+2, 55 >+2,  33 >+2, 33 >+2,  6C >+2, 55 >+2,  48 >+2, 33 >+2,
;
;  co-ord           (   x1l     x1u   ,  y1l     y1u )  ( x2l     x2u   ,   y2l     y2u   )
      00>400,$

;3
                00>+11,  77 >+2, 77 >+2,  44 >+2, 11 >+2,  6D >+2, 77 >+2,  3A >+2, 11 >+2,
;
;  co-ord           (   x1l     x1u   ,  y1l     y1u )  ( x2l     x2u   ,   y2l     y2u   )
      00>600,$

;4
                00>+11,  88 >+2, 88 >+2,  BB >+2, EE >+2,  78 >+2, 88 >+2,  A2 >+2, EE >+2,
;
;  co-ord           (   x1l     x1u   ,  y1l     y1u )  ( x2l     x2u   ,   y2l     y2u   )
      00>800,$

;5
                00>+11,  B3 >+2, FF >+2,  FF >+2, FE >+2,  B3 >+2, FF >+2,  FF >+2, FE >+2,
;
;  co-ord           (   x1l     x1u   ,  y1l     y1u )  ( x2l     x2u   ,   y2l     y2u   )
      00>1000,$

;6
                00>+11,  00 >+2, 00 >+2,  00 >+2, 00 >+2,  03 >+2, 00 >+2,  03 >+2, 00 >+2,
;
;  co-ord           (   x1l     x1u   ,  y1l     y1u )  ( x2l     x2u   ,   y2l     y2u   )
      00>1200,$

;7
                00>+11,  00 >+2, 00 >+2,  00 >+2, 00 >+2,  05 >+2, 00 >+2,  00 >+2, 00 >+2,
;
;  co-ord           (   x1l     x1u   ,  y1l     y1u )  ( x2l     x2u   ,   y2l     y2u   )
      00>1400,$
```

```
;8
                00>+11,  07 >+2, 00 >+2,  00 >+2, 00 >+2,  00 >+2, 00 >+2,  00 >+2, 00 >+2,
;
;  co-ord            (   x1l     x1u   ,  y1l     y1u )  ( x2l     x2u   ,   y2l     y2u   )
      00>1600,$

;9
                00>+11,  00 >+2, 00 >+2,  00 >+2, 00 >+2,  00 >+2, 00 >+2,  06 >+2, 00 >+2,
;
;  co-ord            (   x1l     x1u   ,  y1l     y1u )  ( x2l     x2u   ,   y2l     y2u   )
      00>1800,$
;10
                00>+11,  0C >+2, 00 >+2,  03 >+2, 00 >+2,  00 >+2, 00 >+2,  00 >+2, 00 >+2,
;
;  co-ord            (   x1l     x1u   ,  y1l     y1u )  ( x2l     x2u   ,   y2l     y2u   )
      00>2000

end
```

Appendix 6.
CLASP data.

This appendix consists of a set of results from a CLASP analysis of the full system.

TEST ANALYSIS REVIEW

Uncontrollable nodes

BUSAA<0>	BUSA11<0>	BUSA21<0>	BUSA21<1>	BUSA21<2>	BUSA21<3>
BUSA21<4>	BUSA21<5>	BUSA21<6>	BUSA21<7>	BUSA21<8>	BUSA21<9>
BUSA21<10>	BUSA21<11>	BUSA21<12>	BUSA21<13>	BUSA21<14>	BUSA21<15>
T1__OB1<0>	T1__OB2<0>	T5__OB1<0>	T5__OB2<0>	T9__OB1<0>	T9__OB1<1>
T9__OB1<2>	T9__OB1<3>	T9__OB1<4>	T9__OB1<5>	T9__OB1<6>	T9__OB1<7>
T9__OB1<8>	T9__OB1<9>	T9__OB1<10>	T9__OB1<11>	T9__OB1<12>	T9__OB1<13>
T9__OB1<14>	T9__OB1<15>	T9__OB2<0>	T9__OB2<1>	T9__OB2<2>	T9__OB2<3>
T9__OB2<4>	T9__OB2<5>	T9__OB2<6>	T9__OB2<7>	T9__OB2<8>	T9__OB2<9>
T9__OB2<10>	T9__OB2<11>	T9__OB2<12>	T9__OB2<13>	T9__OB2<14>	T9__OB2<15>

Total number of uncontrollable nodes : 54

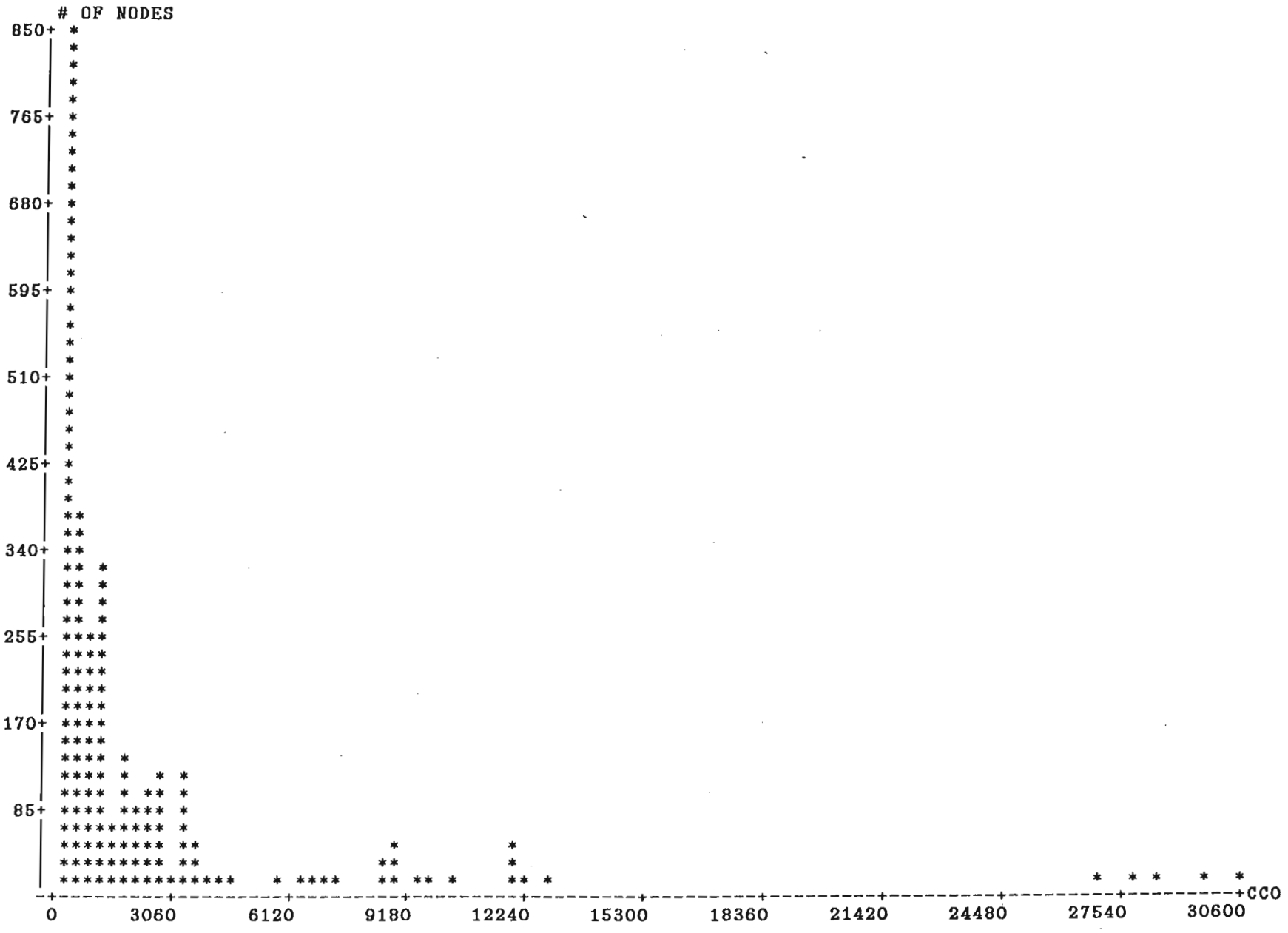
Unobservable nodes

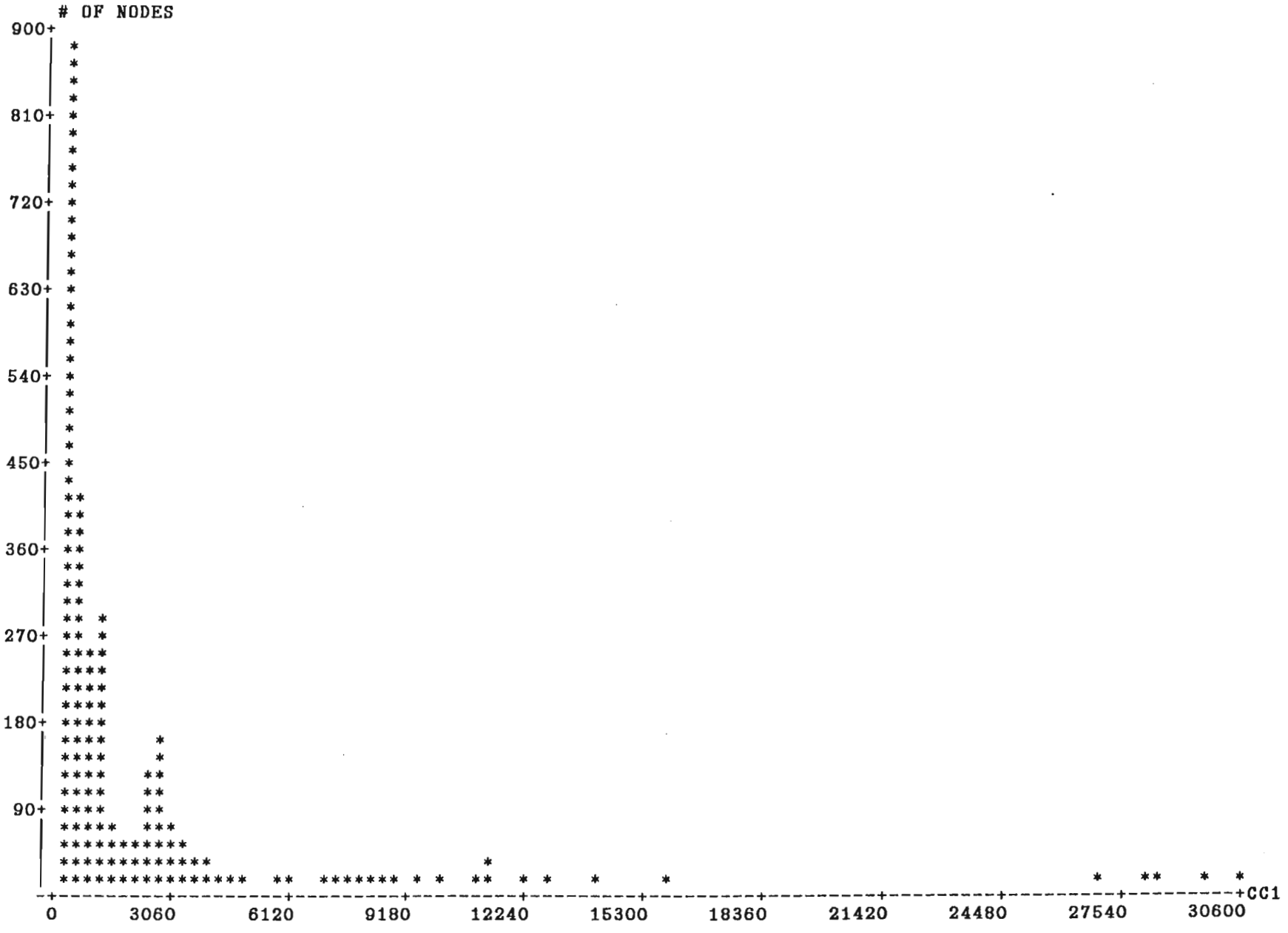
Total number of unobservable nodes : 0

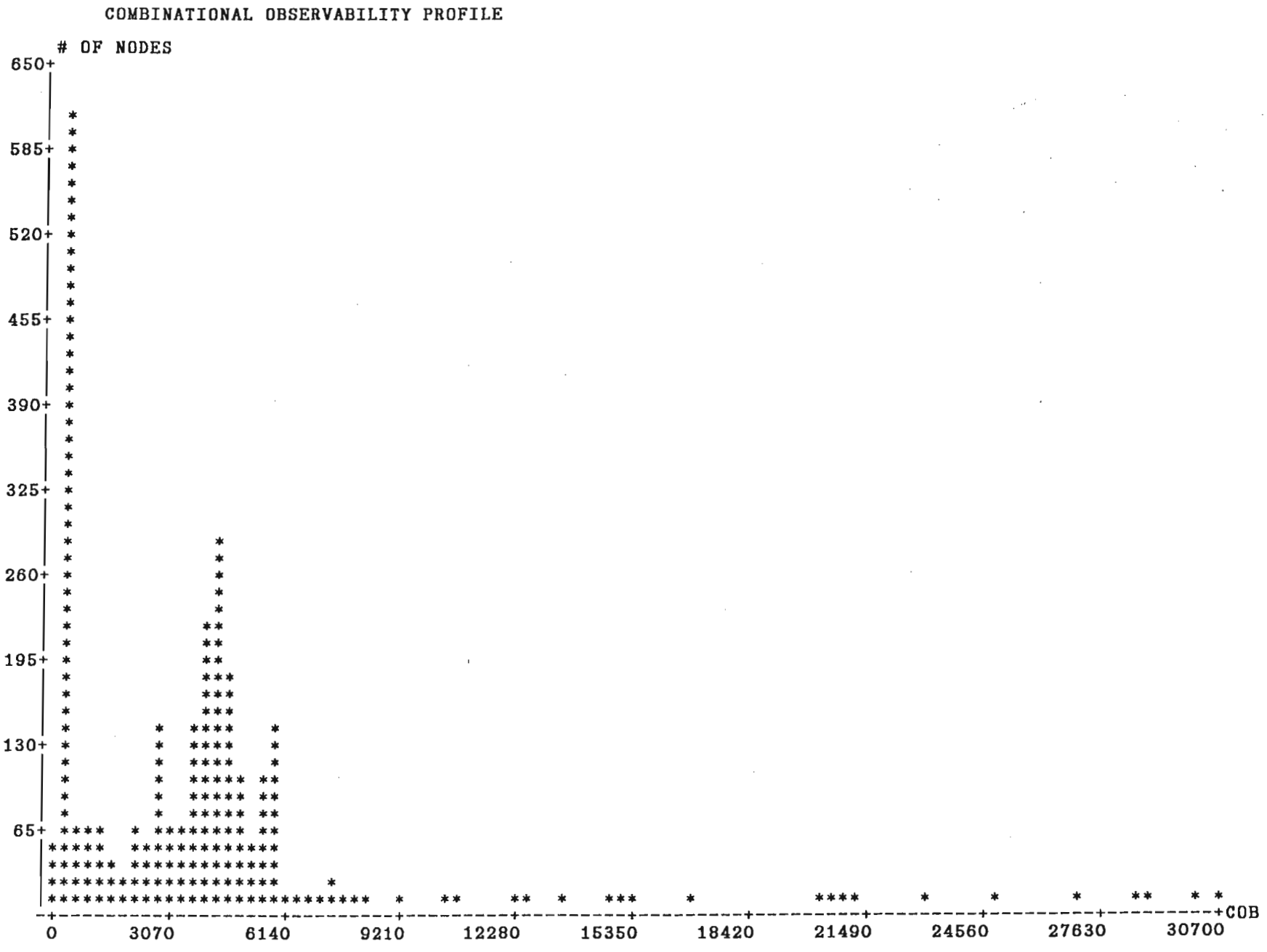
Max value for com cont : 30517 Node name : P5C4
Max value for seq cont : 4447 Node name : P5B10
Max value for com obsv : 30675 Node name : CC1_INCK
Max value for seq obsv : 4467 Node name : CC1_ENB

TEST ANALYSIS SUMMARY

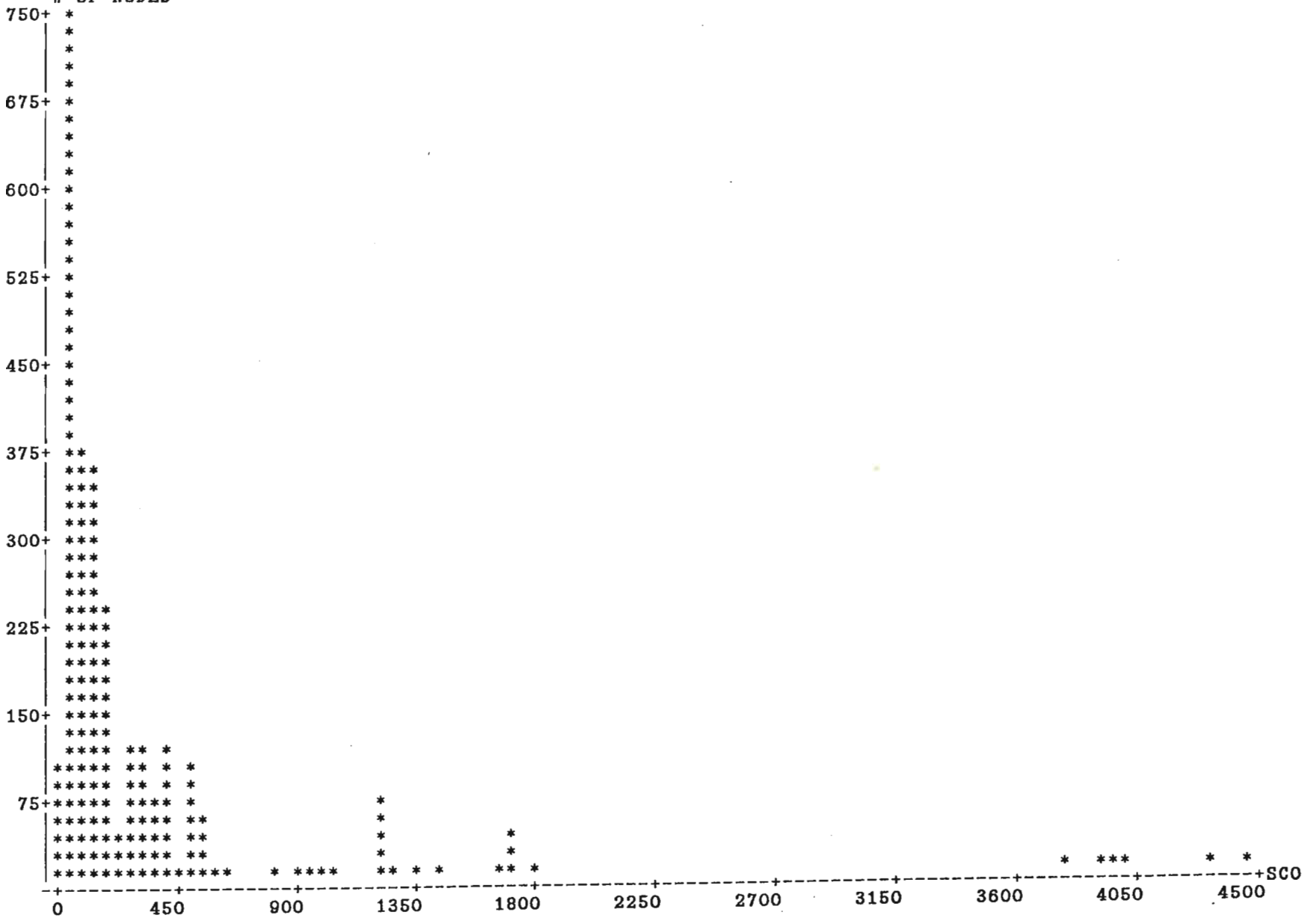
Number of circuit nets = 2495
Total number of uncontrollable or unobservable nodes = 54
This is 2.1% of total number of nodes
Combinational uncontrollable 0 nodes = 3
Combinational uncontrollable 1 nodes = 51
Combinational unobservable nodes = 0
Sequential uncontrollable 0 nodes = 3
Sequential uncontrollable 1 nodes = 51
Sequential unobservable nodes = 0
Number of primary inputs = 13
Number of primary outputs = 44
Number of test points = 0
Number of fanout branches = 5552
Number of feedback loops or reconvergent fanouts = 3057
Total fault population = 4990

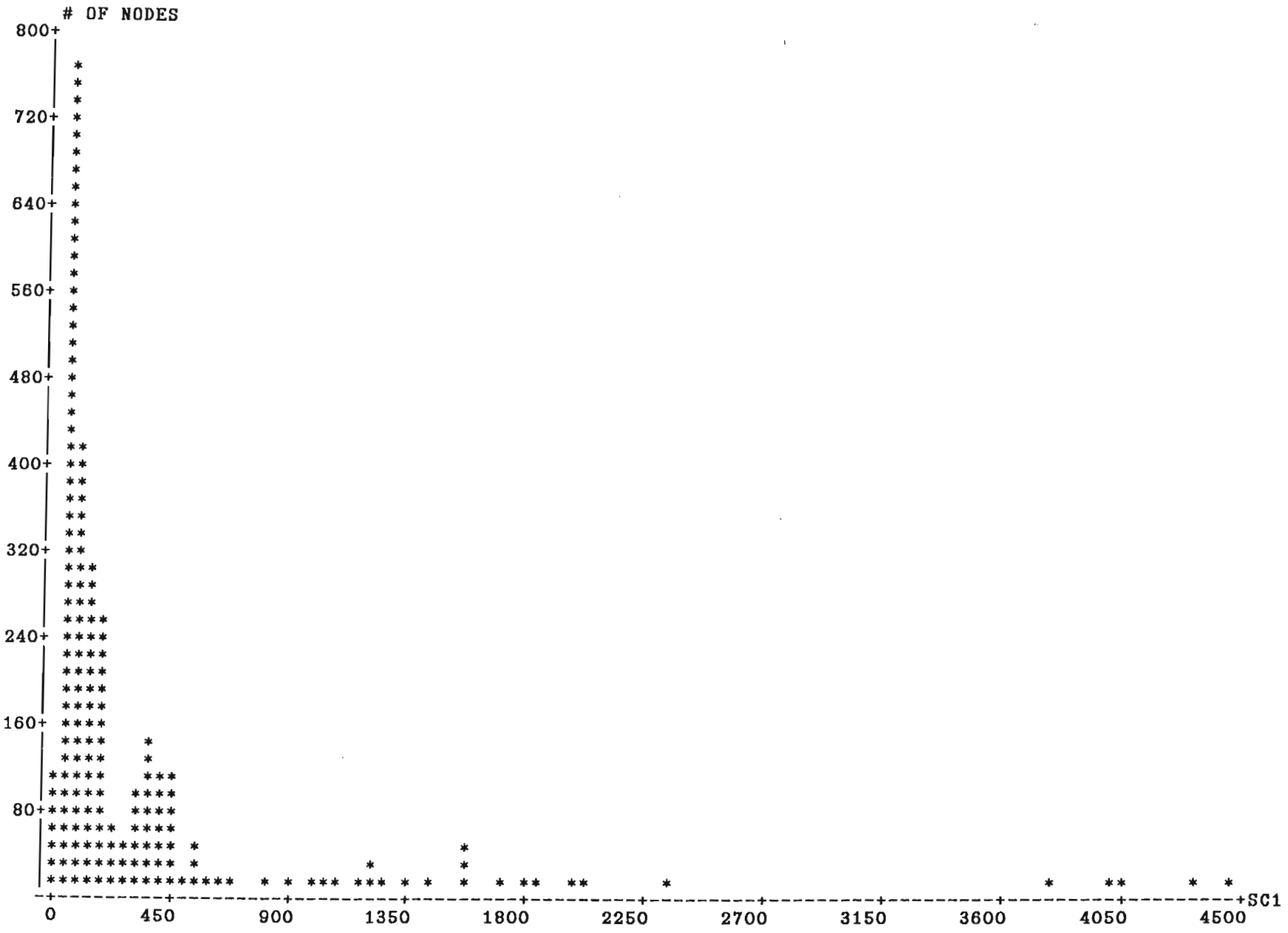

COMBINATIONAL O-CONTROLLABILITY PROFILE
OF NODES
850+
765+
680+
595+
510+
425+
340+
255+
170+
85+
0
3060
6120
9180
12240
15300
18360
21420
24480
27540
30600
CCO

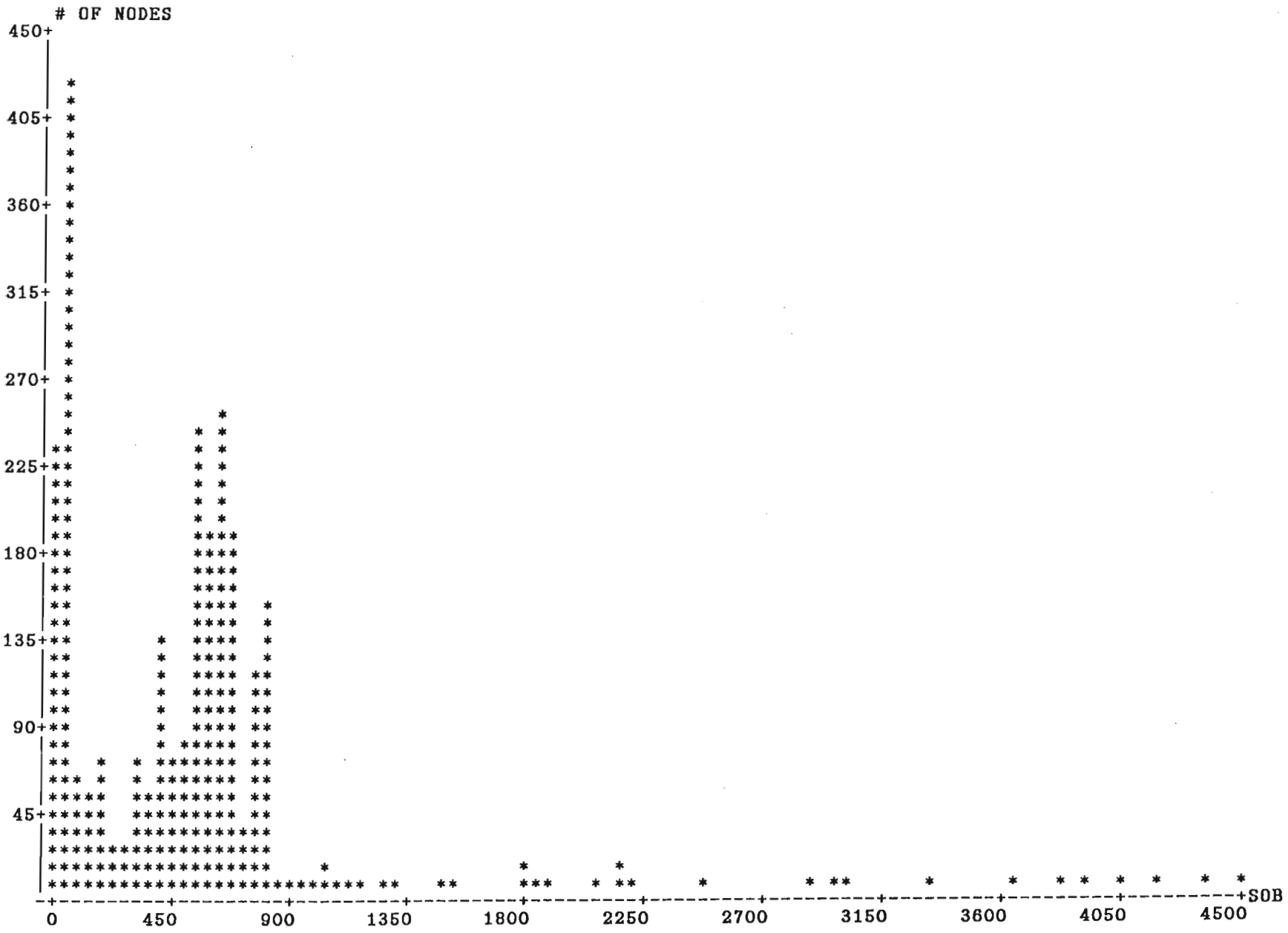

COMBINATIONAL 1-CONTROLLABILITY PROFILE
OF NODES
900+
810+
720+
630+
540+
450+
360+
270+
180+
90+
0
3060
6120
9180
12240
15300
18360
21420
24480
27540
30600
CC1


COMBINATIONAL OBSERVABILITY PROFILE
OF NODES
650+
585+
520+
455+
390+
325+
260+
195+
130+
65+
0
3070
6140
9210
12280
15350
18420
21490
24560
27630
30700
COB

SEQUENTIAL O-CONTROLLABILITY PROFILE




SEQUENTIAL 1-CONTROLLABILITY PROFILE
OF NODES
800+
720+
640+
560+
480+
400+
320+
240+
160+
80+
0
450
900
1350
1800
2250
2700
3150
3600
4050
4500
SC1


SEQUENTIAL OBSERVABILITY PROFILE
OF NODES
450+
405+
360+
315+
270+
225+
180+
135+
90+
45+
0
450
900
1350
1800
2250
2700
3150
3600
4050
4500
SOB

Appendix 7.
Design review.

This is a reprint of the final design review.

GATE ARRAY DESIGN REVIEW 3

Date: 19-Sep-86

CLA5524

Those present:

Name	Function
M. Izzard	Designer
B. Roper	Reviewer

1: GENERAL AND COMMERCIAL

1.1 Company Name : R.E SWANN

1.2 Device Title : Rasteriser

1.3 Commercial No :

1.4 Mask No :

1.5 Has an order been placed ? :

1.6 Differences between actual device and previous esimate in design review 2 :

	Des. Rev. 2	Actual
Array Type		CLA5524
Utilization		78%
Package : Dev		68LCC
: Prod		68LCC
Quantity : Dev		10 tested parts
: Prod		Refer Andy Bliss
Release		Commercial

2 : SOFTWARE STATUS Completed at Swindon

2.1 Autorouter Software SCARP Issue Status : V4R5S

2.2 Autorouter Technology Library Issue Status : CLA5000

3 : DEVICE DETAILS

3.1 Have the Peripheral Cell Locations been indicated on the Schematic Blank ? : YES

3.2 Have the Peripheral Cell Locations been checked against the Netlist ? : YES

3.3 Have all extra Supply Pads been added to the Netlist ? : — N/A

3.4 Has the Pin Identification Table been completed ? : YES

3.5 Are the Peripheral Cells placed in the recommended positions per the prefered Bonding Diagram ? : YES

3.6 Have the I/O Port Components been placed at the same Position ? : —

* PSL will produce a plot and the Bonding Diagram. *
* Bonding problems must be resolved before starting Maskmaking. *

3.7 What is the Pin-Out Mix :

Inputs	13
Outputs	44
I/O Ports	/
+VE Supply	3
-VE Supply	8

3.8 Is any O/P Cell placed more than 12 Peripheral Cell locations from the nearest -VE Supply Pad ? : NO

3.9 Has the circuit been compiled and simulated with the track loads included ? : YES

Result of [redacted] Analysis : NO ERRORS. Only ERRORS reported are CK_SKEW on transmission gate select lines. These are safe.

Result of GLITCH Analysis : OK

Result of Max Speed Simulation : OK

Critical Timing Paths : None at 5MHz, plenty of design margin

Critical set-up & hold Times : None at 5MHz, plenty of design margin

3.10 Have any Nodes exceeded the fan-out recommendations ? : NO

3.11 Are there any differences between min and max simulation table outputs. (90% period sample) ? : NO

3.12 Does the circuit use a Xtal Osc maintaining cell ? : NO

Placement coordinates : —

3.13 Will a further Design Review 3 be necessary ? : — NO

New review date :

4 : VAX FILES AND AUTOROUTING INFO

4.1 Directory Name : TS:[IZZARD_M.CCLASSIC.ROUT2]

4.2 File Names :

Compiler Source Input	(.CCL)	:	2.CCL
Userlib	(.CCL)	:	2LIB.CCL
Systemlib	(.CCL)	:	
Include files		:	
Track Loading file	(.CTL)	:	[redacted] RAST.CTL
Simulator Source Input	(.SCL)	:	VEC.SCL
Layout file	(L.CLM)	:	[redacted] RASTL.CLM
Routing file	(R.CLM)	:	[redacted] RASTR.CLM
Priority route file	(PRIORITY.NET)	:	NONE

4.3 Where any Manual Modifications made ? : NO.

6 : TIMSCALES

6.1 Date Files to be received by PSL. : 23-Sept 86

6.2 Estimated Delivery date of ~~Untested A Models~~ 10 Tested Prototypes. : 16 Nov '86

6.3 Estimated date for Customer Approval of ~~A~~ Prototypes Models. : + 4 wks from receipt
~~Including Supply of TWO Working Devices.~~

~~6.4 Estimated data for Shipment of fully tested B Models~~ :

G. B. Ripes
..... 23/9/86
Design Reviewer

.....
Designer

PACKAGE PIN IDENTIFICATION TABLE

PIN	LIBRARY CELL	NODE NAME
1	OPS	XS1Ø
2	OPS	XBREQ
3	IPS	XBACK
4	IPS	XBG
5	V_{DD}	+ve
6	V_{SS}	-ve
7	IPS	RESET
8	OPS	XINST5
9	"	XINST4
10	"	" 3
11	"	" 2
12	"	" 1
13	V_{SS}	-ve
14	OPS	XINSTØ
15	IPS	XMEMR
16	IPS	DATAEX7
17	"	" 6
18	"	" 5
19	"	" 4
20	"	" 3
21	"	" 2
22	"	" 1
23	V_{SS}	-ve
24	IPS	DATAEXØ
25	"	XCLOCK
26	OPS	PIXEL<16>
27	"	<17>
28	"	<18>
29	"	<19>
31	OPS	PIXEL<20>
32		<21>
33		<22>
34		<23>
35		<24>
36		<25>
37		<26>
38		<27>
39		<28>
40	V_{SS}	-ve
41		<29>
42		<30>
43		<31>
44		<Ø>
45		<1>
46		<2>
47	V_{SS}	-ve
48	V_{DD}	+ve
49	OPS	<3>
50		<4>
51		<5>
52		<6>
53		<7>
54		<8>
55		<9>
56		<10>
57	V_{SS}	-ve
58		<11>
59		<12>
61	OPS	<14>
62	OPS	" <15
63	"	XBR
64	V_{SS}	-ve
65	V_{DD}	+ve
66	OPS	XRDR
67	OPS	XS1
68	OPS	XS9
69		
70		
71		
72		
73		
74		
75		
76		
77		
78		
79		
80		
81		
82		
83		
84		