Increasing the capacity of a fault tolerant digital telephone exchange

Muijselaar, J.N.M.

Award date:
1990

Eindhoven University of Technology
Department of Electrical Engineering
Digital Systems Group

Increasing the capacity of a fault tolerant digital telephone exchange

by J.N.M. Muijselaar

August 1990, Master Thesis Report
Eindhoven, the Netherlands

Supervisors : prof. ir. M.P.J. Stevens
ir. M.J.M. van Weert

Increasing the Capacity of a Fault Tolerant Digital Telephone Exchange

by John Muijselaar

Summary

At the Digital Systems Group, Department of Electrical Engineering, Eindhoven University of Technology, a fault tolerant digital telephone exchange is being developed.

This report describes the research that has been done in order to increase the capacity of the prototype exchange that is being developed.

Various architectures of time and space switch element for the switching network of the exchange have been designed, implemented, simulated and tested using a hardware design tool.
With IC technology moving into the sub micron area some of the architectures proposed can be used in order to design VLSI or ULSI circuits for a high capacity fault tolerant digital exchange.

Table of Contents

List of Figures

1 Introduction

In 1986 a project group was set up at the Digital Systems Group, Department of Electrical Engineering, Eindhoven University of Technology, with the task to design and implement a fault tolerant digital telephone exchange. Since that time students are working on this project, which was first under the supervision of prof. ir. A. Heetman and ir. M.J.M. van Weert, and is now under the supervision of prof. ir. M.P.J. Stevens and ir. M.J.M. van Weert.

The purpose of the project is to develop an exchange that can process 2048 (i.e. 4096 half) telephone calls. The intermediate goal of the project is to design and implement a prototype exchange that can switch 256 half telephone calls.

This report describes the research that has been done in order to increase the capacity of the prototype exchange.

Chapter 2 of this report gives an overview of some of the aspects of communication theory and VLSI design the reader should be familiar with.
The prototype exchange is described in chapter 3.
Chapter 4 describes the architecture of traditional and new time switch elements, and chapter 5 describes the architecture of various space switch elements.
The resulting architecture for the switching network is discussed in chapter 6.
Conclusions and recommendations are discussed in chapter 7.

2 Theory

2.1 Introduction

In this chapter we will briefly discuss some of the theoretical aspects of telecommunication and VLSI-design. Of course we do not intend to give a summary of these parts of the wide field of electrical engineering. This chapter only gives a short overview of the various subjects the reader should be familiar with in order to understand the entire report.

First we will discuss some basic aspects of digital transmission of signals. Next we discuss the basic aspects of digital communication switching. Then the (4,2) fault tolerance concept is briefly discussed and the fundamentals of state of the art VLSI-design are mentioned.

2.2 Digital Transmission of Analog Signals

2.2.1 Sampling

It is often desired and useful to represent an analog signal in terms of its sampled values taken at appropriately spaced intervals. After the sampling process the signal can be quantised, and the resulting digital signal can be processed in many ways. It is clear that this technique is widely used nowadays. See [Arnbak] for an overview of the advantages and disadvantages of digital versus analog communication systems.
The principle of sampling can be explained using the switching sampler shown in figure 2.1

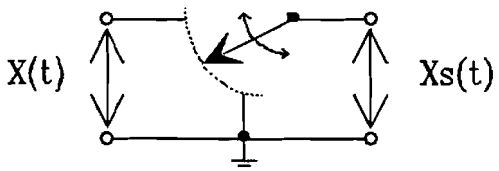


Figure 2.1 *Switching sampler*

By applying a little bit Fourier theory it can be shown that the original signal can only be recovered without distortion from the sampled signal if the minimal sampling frequency satisfies $f_s >= 2 f_x$, or $T_s < 1/(2 f_x)$. Here f_x is the maximum frequency of the signal to be samped.
The minimal sampling frequency $fx_{min} = 2 f_x$ is called the Nyquist rate.
Analog speech signals are sampled at f_x=8000 Hz

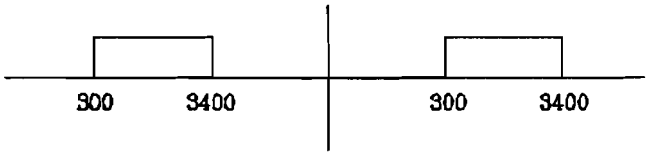


Figure 2.2 *Spectrum of speech signals*

2.2.2 Quantizing

Representing analog sampled values by a finite set of levels is called quantizing.

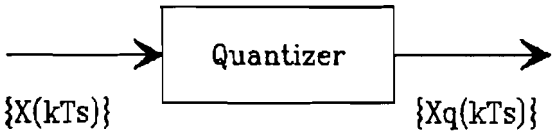


Figure 2.3 *Quantizer*

The random waveform X(t) is sampled between time intervals T_s and the sampled values $X(kT_s)$ are converted to one of Q allowable levels, m_1, m_2, m_3, .. m_Q, according to some predefined rule.

$$X_q(kT_s) = m_i \text{ if } x_{i-1} <= X(kT_s) < x_i$$
$$x_0 = -\infty, x_Q = +\infty$$

x_i are the decision levels of the quantizer.

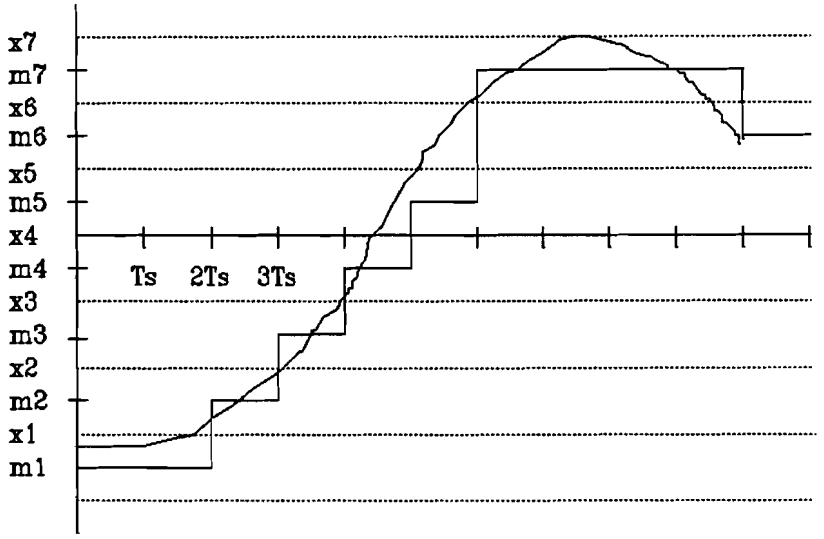


Figure 2.4 *Quantizer operation*

When doing uniform quantizing, the quantizing intervals in which the variable X is divided are all of equal length. A nonuniform quantizer divides the range of the continuous variable X in intervals of variable length. It can be shown that such a quantizer yields a higher average signal to quantizing noise power ratio than the uniform quantizer when the signal probability density function is nonuniform - which is the case in many situations. In practice, a nonuniform quantizer is realized by sample compression followed by a uniform quantizer.

Winking to international standardization efforts there are two commonly used compression laws, namely the so-called u-law and the A-law.

2.2.3 Pulse Code Modulation (PCM)

The system of transmission in which sampled and quantised values of an analog signal are transmitted via a sequence of codewords is called Pulse Code Modulation. Sampling, quantizing and coding using A law PCM codecs is shown in figure 2.5

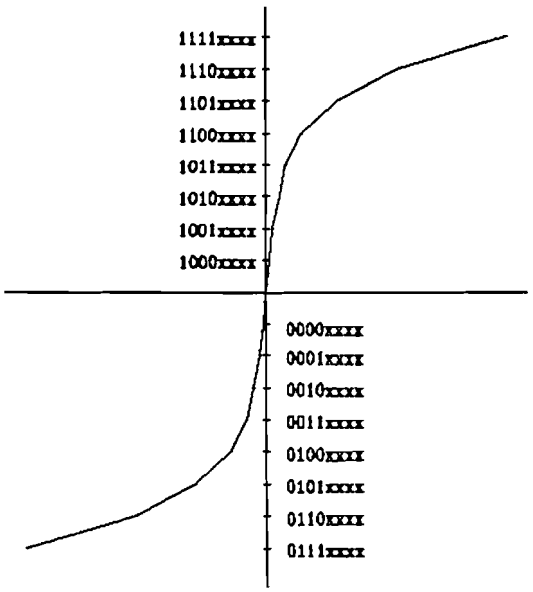


Figure 2.5 *Sampling, quantizing and coding using A law PCM codecs*

2.2.4 Time Division Multiplexing

Multiplexing is the process of combining a number of independent baseband signals to form a higher order single signal. This process and its inverse (demultiplexing) are employed in communication systems to share the limited resources in an orderly way. Time Division Multiplexing (TDM) is a technique used for transmitting several signals over a communication channel by dividing the time frame into slots, one slot for each message.
Again winking to international standardization efforts there are two widely used standards on TDM. The European standard defines a 30 voice channels primary rate (with two channels for signalling information this yields a bit rate of 2048 KBit/sec), while the North American/Japanese hierarchy uses a 24 voice channel (1544 KBit/sec) primary rate.

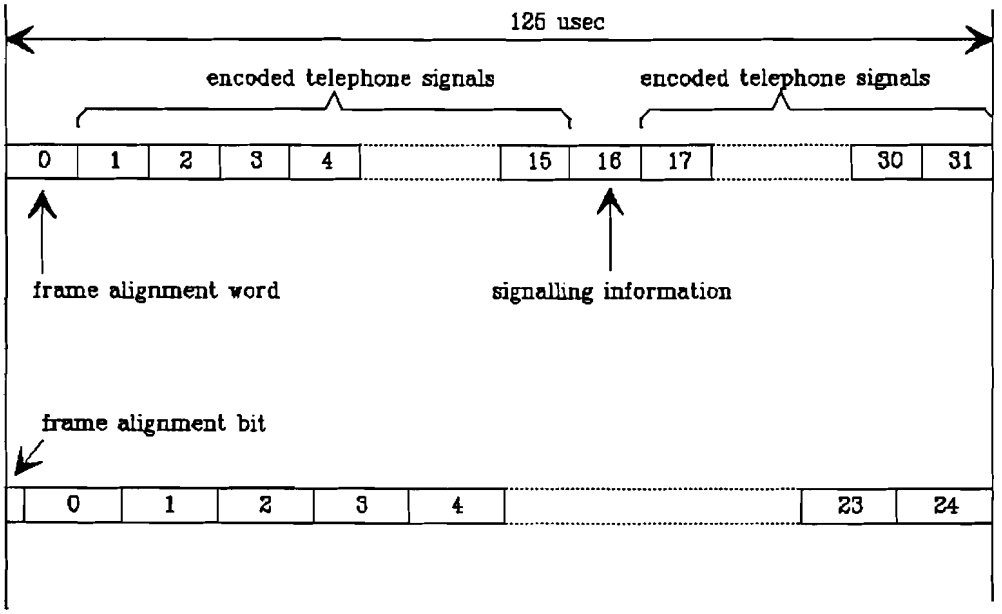


Figure 2.6 *Primary rate (T1) CCITT standards*

In the prototype exchange our own standard, based on a double European primary rate channel (e.g. 4096 KBit/sec, 64 PCM channels), is used.

2.3 Telecommunication Switching

2.3.1 Telecommunication network

There are about 300 million telephones in the world. From each it is possible to call any of the others. The number of potential connections is therefore about 4.5 10^{16}

The most straightforward way of implementing the network would simply be to run 300 million copper wire's into everyone's house, each leading to a different telephone. This method, although conceptually simple, has some drawbacks, to put it mildly. The next simple approach might be to have one giant switch somewhere, with one line running to each of the worlds telephones.
In reality the telephone network is organized as a highly redundant, multilevel hierarchy (Figure 2.7).

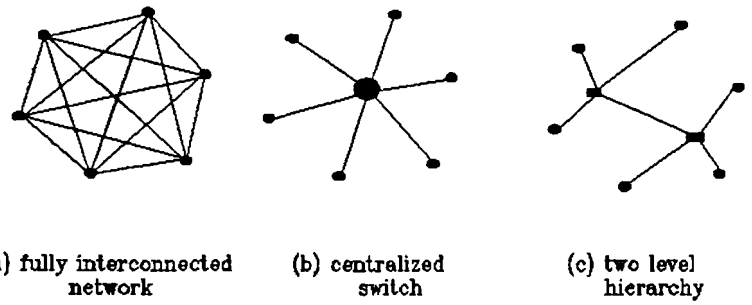


Figure 2.7 *Telephony network structures*

Each telephone has two wires coming out of it and going directly to the nearest local exchange. The connection between subscriber and a local exchange is called a local loop. The local exchanges are connected to higher order exchanges.

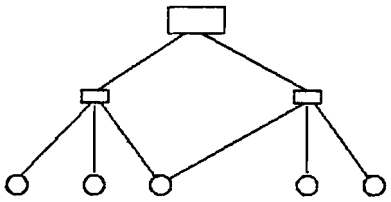


Figure 2.8 *Telephony network hierarchy*

2.3.2 Circuit switching

The telephone network is a so called circuit switched network. This means that when you make a telephone call, the switching equipment within the telephone system seeks out a physical 'wire' route from your telephone to the receiver's telephone. This technique is shown schematically in Figure 2.9. Each of the rectangulars represent a carrier switching office (end exchange, toll exchange, etc) In this example, each exchange has three incoming lines and three outgoing lines. When a call passes through a switching exchange a physical connection is (conceptually) established between the line on which the call came in and the one for outgoing lines, as shown by the dotted lines.

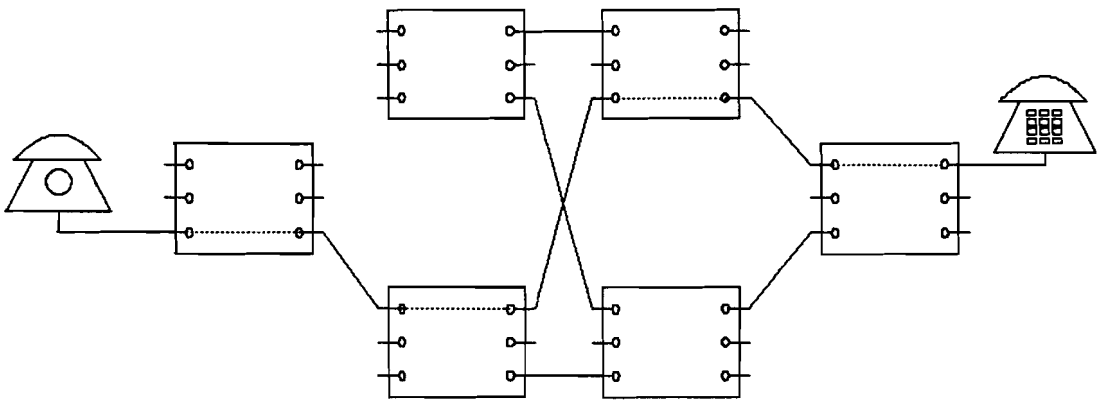


Figure 2.9 *Circuit switching*

The actual switching in the exchanges is done using space switches and time switches.

2.3.3 Space Switching

The principle of space switching can be easily understood by considering a crosspoint switch. At each crosspoint there is a relay. The contacts of the relay can be open or closed, and in this way inlets can be connected to outlets.
An exchange that fully interconnect 100 inlets to 100 outlets can be made using an array of 10.00 relays.

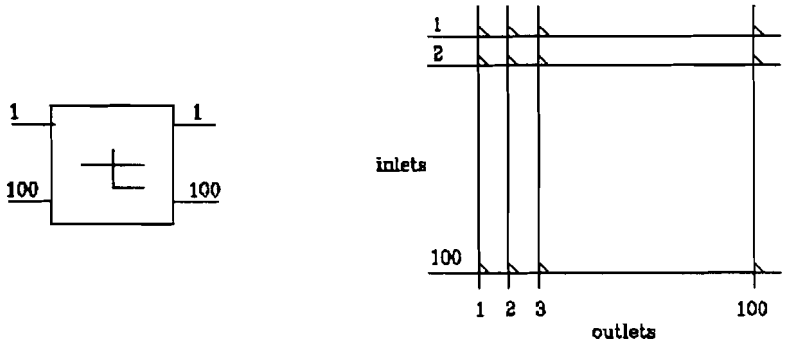


Figure 2.10 *Crosspoint switch*

A more cost effective method of a switching network is shown in figure 2.11.

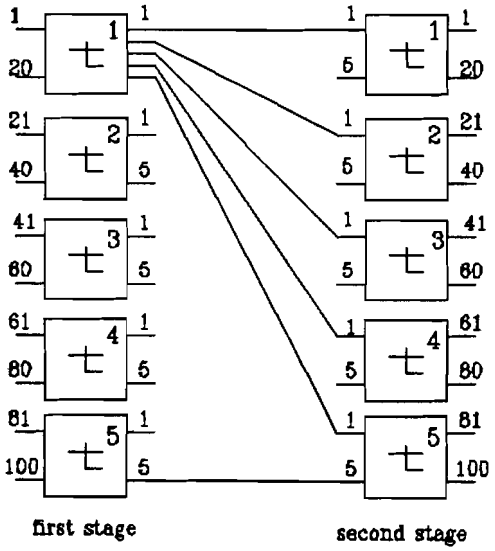


Figure 2.11 *Two stage network*

This switching network only needs 1000 relays, but it can be shown easily that the network is blocking. This means that not always an inlet can be interconnected to a free outlet. Later in this paragraph we will discuss the design of a non-blocking three stage network.

2.3.4 Time Switching

The principle of time switching is illustrated in figure 2.12

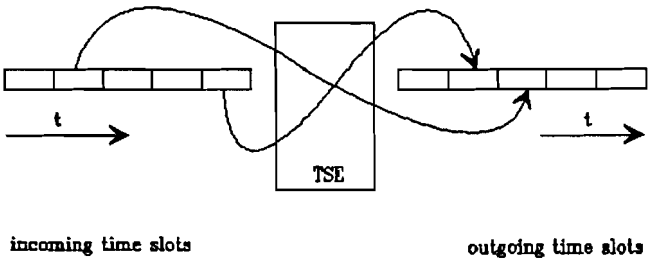


Figure 2.12 *Time Switch Element*

A time switch element must be capable of transferring the contents of any particular inlet timeslot into any specified outlet timeslot for every inlet/outlet timeslot pair in any arbitrary pairing. We should note that if broadcasting or 'telephony meetings' should be provided these requirements may not be enough.

2.3.5 Multiple Stage Network

As we already have seen we can increase the number of switching stages of a network. This is usually done to increase the capacity of the network, and to do switching at reasonable costs. Figure 2.13 illustrates a three stage network.

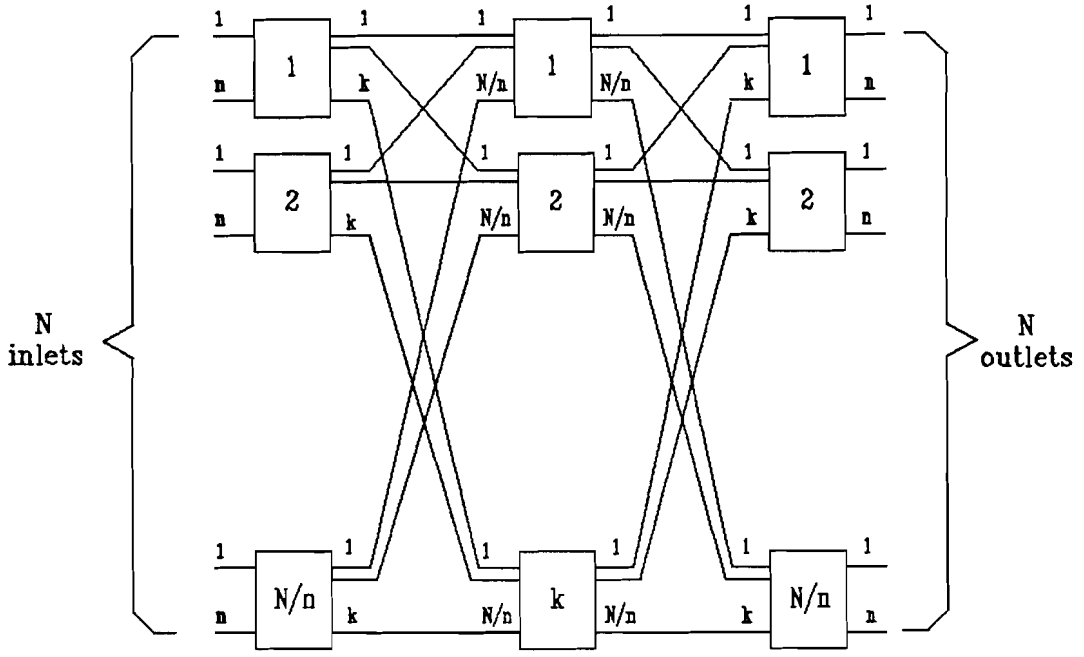


Figure 2.13 *Three stage network*

For such an array to be non-blocking it can be shown [Clos] that the number of second stage switches k must be at least be equal to 2n-1. In our exchange k=2n.

The capacity of the basic two stage network (figure 2.9) can be further enhanced, by time multiplexing at the input (and therefore having to switch at higher speed) and by adding more stages.
For full connectivity it is necessary to have both time and space switch functions in a switching network. There are several ways to implement a three stage network. Three stage arrangements have proved effective both in TST (figure 2.15) and STS (figure 2.14) configurations.

We will see later that the VLSI implementation of a space switch element requires a lot of wiring, whereas a time switch element requires a lot of RAM. Based on this simple observation, and keeping in mind that wiring an IC is becoming a major problem in IC design, whereas much research is done to make RAMs very efficiently (e.g. very much RAM on small piece of silicium) we use a TST network in the (4,2) prototype exchange.

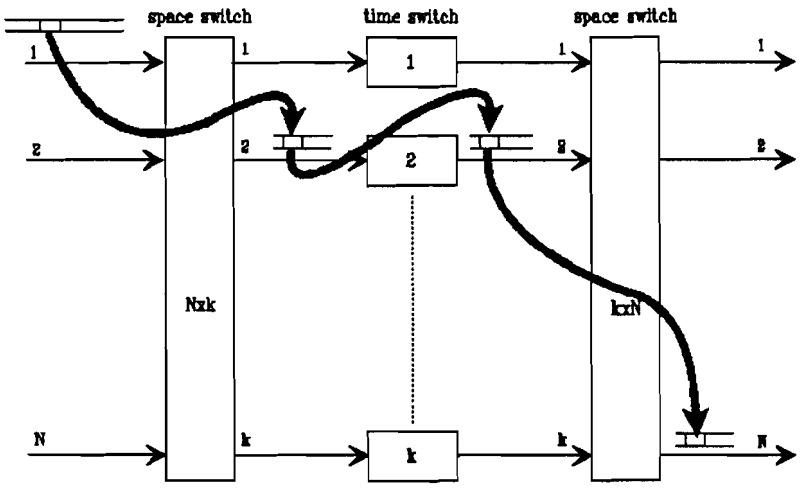


Figure 2.14 *STS switching network*

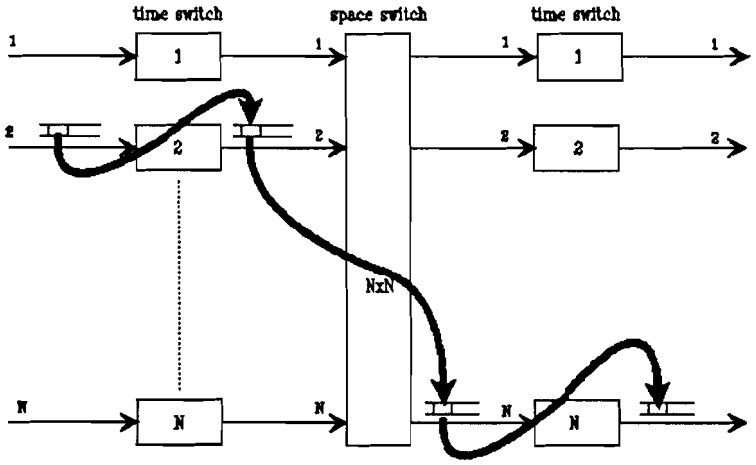


Figure 2.15 *TST switching network*

2.4 (N,K) Concept Fault Tolerance

In many computer applications a degree of reliability and availability is required which cannot be fulfilled by normal general purpose computers. In the last two decades various methods have been proposed and implemented for improving the reliability and availability of computer systems.
The (N,K)-concept fault tolerance [Krol] is based upon a 'distributed implementation' of a symbol error correcting code. The (N,K)-concept makes it possible to choose the ratio between processor and memory redundancy and in this way to optimize the total amount of redundancy.
The (4,2)-concept is to be considered as a specific version of the more general (N,K)-concept. The basic architecture of the (4,2)-concept is given in figure 4.16. It consists of four 'fault isolation areas'. Within these fault isolation areas (slices) faults may be interdependent, whereas they are assumed to be independent between the slices.
In the example each slice contains a 8 bit microprocessor and a 4 bit wide memory. The data stored in the memories are encoded using a special symbol error correcting code. Each dataword consisting of two symbols of four bits is encoded into a codeword of four symbols of four bits. Each of the four memories contains exactly one of these four symbols, a different symbol in each memory. Any two of these symbols can be regarded as data symbols, while the other two symbols are check symbols. So the processor part is quadrupled and the memory part is only doubled as compared with a non-redundant computer.

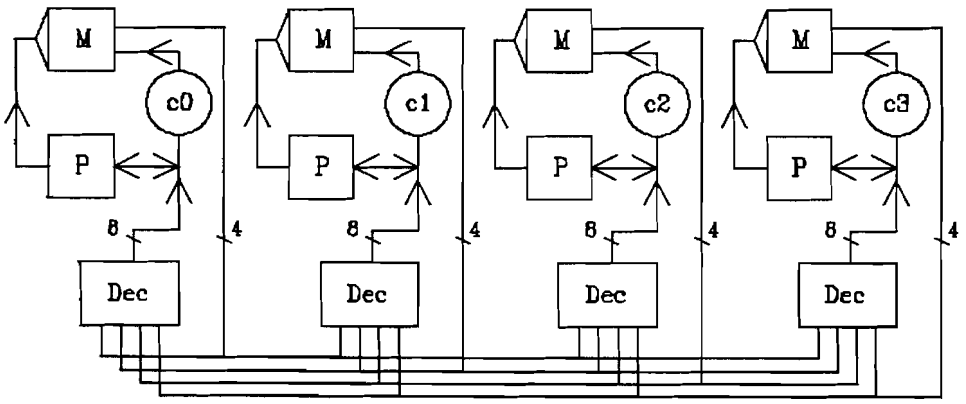


Figure 2.16 *Basic architecture of the (4,2) concept*

The applied symbol error correcting code has the following properties. In random mode it can correct error patterns that are one of the following types:
- one single symbol error
- double bit errors (not necessarily located in the same symbol)

In the erasure mode (i.e. we explicitly know that a certain slice is malfunctioning) it can correct

- one single bit error in one of the other slices.

Four slices of the (4,2)-concept are running synchronously. If there are no faults in the system all four microprocessors contain at any time the same information. When information is sent to the memory, i.e. a write operation, in each slice the 16-bit word is encoded in a different 8 bit symbol. The four symbols in the four slices form a codeword of the above mentioned symbol-error-correcting code. Note that during a write operation there is no interchange of information between the slices. When information is transferred from the memory to the processor (i.e. a read operation), each processor receives the complete codeword of four 8-bit symbols. If one of these symbols is incorrect it will be corrected by the decoders available in each slice. The only interconnection between the slices are the 8 bit symbols. So whatever the nature of a fault occurring in a slice may be, it will always be masked by the decoders. To a large extend the faults in a slice are caused by the memories and will manifest themselves as single bit errors because the memories are as usual bit-sliced. Due to the special properties of the applied symbol-error-correcting code, therefore, the failure rate of the memory has hardly any influence on the availability of the system.
The decoder normally operates in random mode and will remain in that mode whenever a bit error occurs, bus as soon as a symbol error occurs that is not a bit error the decoder automatically switches to erasure mode. So from that moment on the faulty slice will be ignored by the system, and additionally a single bit error in one of the three remaining slices can be tolerated.

2.5 VLSI design

The development process of a VLSI circuit consists of various phases (or steps).
During the 'Structured VLSI Design Course', which is given by the Digital Systems Group, Department of Electrical Engineering, Eindhoven University of Technology, various aspects of the VLSI design trajectory are discussed.
State of the art VLSI design is becoming so complex that we even not try to give a complete summary. For further reading we refer to the course notes 'Structured VLSI Design Course' and to [Sikkelman], [Lewin] and [Hörbst].
A 'serious' VLSI design traject consists at least of the following phases.

2.5.1 Specification

The specification phase is the first step in the VLSI design traject. The goal is to formulate the functional requirements of the system. Already in this phase it is important to keep the testability of the design in mind. Specification of the tests have to be made. Very much research is being done in order to support the designer during the specification phase.

2.5.2 System design

In the system design phase processes and functions of the functional specifications are mapped on architectures and modules. The design of the architecture and modules is done according to a top down approach.
The system functions are decomposed into a layered (hierarchical) structure of subfunctions by stepwise refinement.

2.5.3 Implementation

In the implementation phase each module will be mapped onto logic elements.
It highly depends on the kind of implementation and the available technology what logic elements may be used.

2.5.4 Integration and Testing

In this phase all modules are interconnected and tested. The specification will be verified.

3 The (4,2)-prototype exchange

3.1 Introduction

A major goal of the research which is done at the Digital Systems Group, Department of Electrical Engineering, Eindhoven University of Technology, is to design and implement parts of a digital telephone exchange.
Since 1986 students are working on the project.
Currently much effort is being done in designing and simulating the prototype exchange [van Lier, Timmer, Bink, Cornelisse, Dijkhuis]. The software system for controlling the system is being developed by [Verhoof] and [Ligtenberg]

3.2 The (4,2)-prototype exchange

Figure 3.1 illustrates the architecture of the (4,2)-prototype exchange. The prototype exchange consists of PCM synchronisation systems with (4,2)-coders, non blocking central switching networks (the TST networks) and (4,2)-decoders.

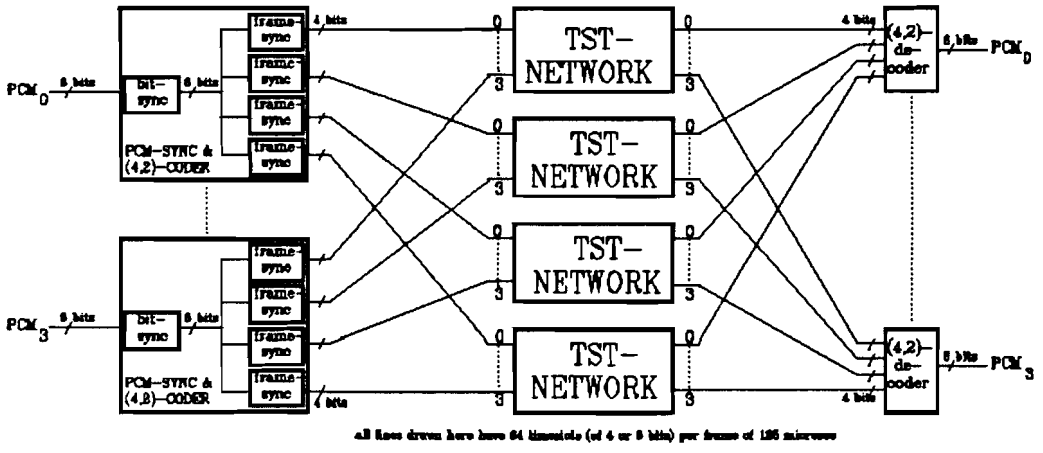


Figure 3.1 *(4,2) exchange*

The central switching network consists of four equal TST networks, as we can see in figure 3.1. Each of those TST networks is placed in one chip, so each TST network is a fault isolation area. Because of the (4,2)-concept, each time slot in the central switching network consists of four bits (at the input of the exchange, a time slot consists of eight bits). Therefore the PCM bit rate of the incoming and outgoing lines

of the central switching network is only half the bit rate outside of the exchange. (e.g. 2 MBit/sec instead of 4 MBit/sec).

Figure 3.2 illustrates the TST-network of the prototype exchange.

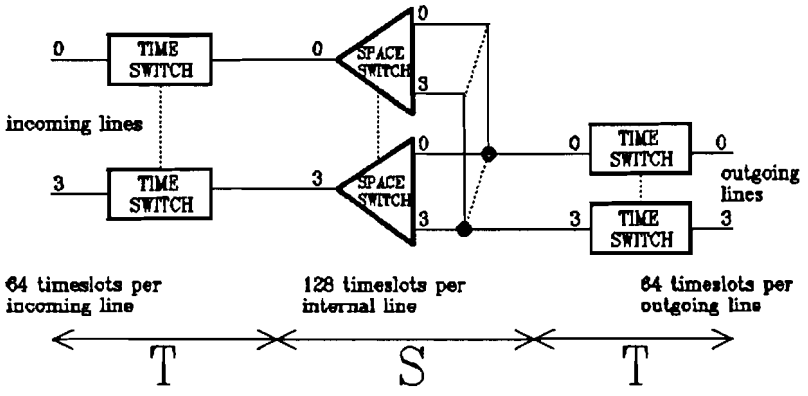


Figure 3.2 *TST network (4,2) prototype exchange*

To achieve a non blocking switching network the number of internal time slots equals twice the number of external time slots. Therefore the bit rate of the internal lines of the TST networks is twice the bit rate of the incoming and outgoing lines.
A connection from one subscriber to another subscriber is set up as follows. First an incoming time slot is switched to another time slot on the same incoming line by the first time switch. Then this time slot is switched to another line (i.e. outgoing line) by a space switch. On this outgoing line, the time slot is switched again to another time slot by the second time switch.
The TST networks consists mainly of three switches as shown in figure 3.3. The time switches have to transfer incoming PCM data to other time slots. Therefore the incoming data is stored in a random access memory (the data memory RAMD). This data memory is written sequentially. The order, in which the data memory is read out, is stored in a connection memory (RAMC). This connection memory is read out sequentially, and this data is used to point to the time slot in the data memory that has to be read out.

The space switches receive the PCM data time slots from the first time switches, and demultiplexers send the data to several (outgoing) lines. Connection memories determine to which of the lines the data has to be send.

Figure 3.3 illustrates a route through a TST network from a subscriber on incoming line 0 and time slot 10 to subscriber on outgoing line 3 and time slot 41. The data memories are read in and the connection memories are read out, using a time slot counter for the addressing. Due to internal delays, a time slot sent out by the data memory of a first time switch (during time slot N) arrives two time slots later in the data memory of a second time switch (i.e. it is stored in time slot N+2). Furthermore a time slot is read out in a second time switch (during time slot M), leaves the TST network one time slot later (i.e. during time slot M+1). The time slot received by the TST network from the PCM synchronisation part also has a delay of one time slot. So a time slot sent by the PCM synchronisation part during time slot N-1 is stored in the data memory location N of the first time switch.

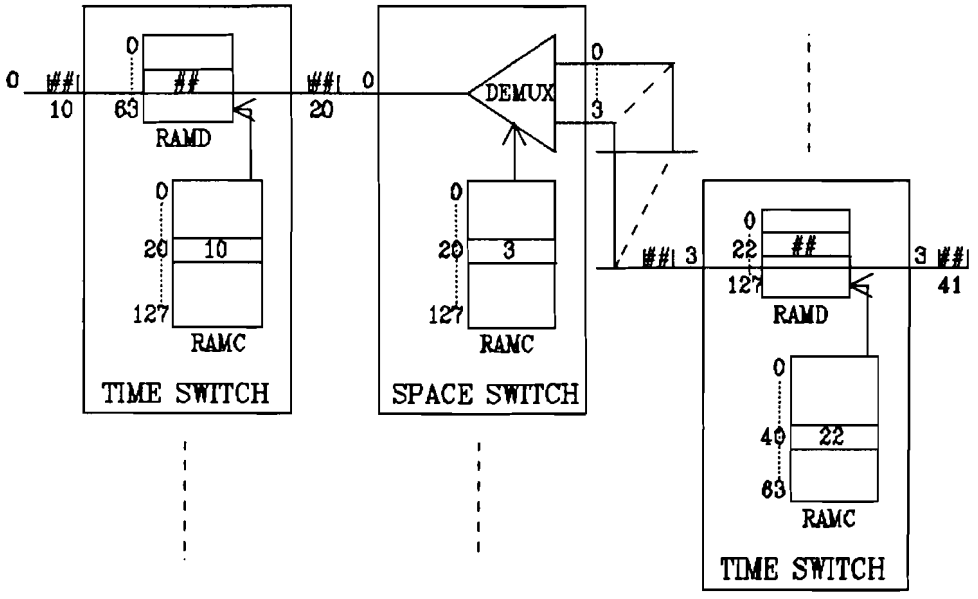


Figure 3.3 *Routing through a TST network*

3.3 Increasing the capacity

In this report we will describe the main results of the research that has been done in order to increase the capacity of the prototype exchange. Based upon timing characteristics the maximum feasible capacity of the current architecture is determined. Some other alternative architectures for time switch and space switch element are also examined. These research results are combined to indicate which number of telephone calls could be switched with this architecture.
Finally a rough guess has been made to determine the expected chip area of these new architectures.

4 Time Switch Element

4.1 Introduction

In this chapter we will discuss how we can increase the capacity of the Time Switch Element designed by [van Lier] and [Timmer].
The requirement of a time switch element is illustrated in figure 4.1.
It must be capable of transferring the contents of any particular inlet time slot into any specified outlet time slot and be able to perform this operation for every inlet/outlet time slot pair in arbitrary pairing.

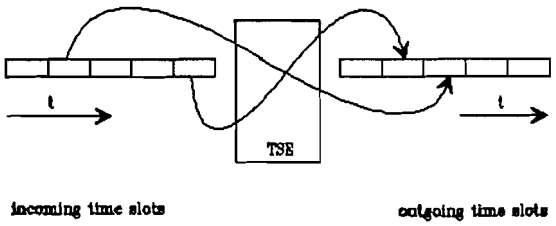


Figure 4.1 *The time slot requirement*

4.2 Traditional Time Switch Element

4.2.1 Introduction

Figure 4.2 illustrates the method by which time switching is performed in a traditional time switch element. The incoming bit stream is stored time-slot by time-slot as it arrives in a data memory. In a separate control memory the information is stored indicating what sample is to be sent in which outgoing time-slot. The outgoing samples are therefore read from the data memory in the order defined by the control memory. Thus, sample x received in time slot i is stored in time slot numerical order in the data memory. The outlet bit stream is made up of samples as directed by the control memory so that the sample value x is output in outlet time slot m.

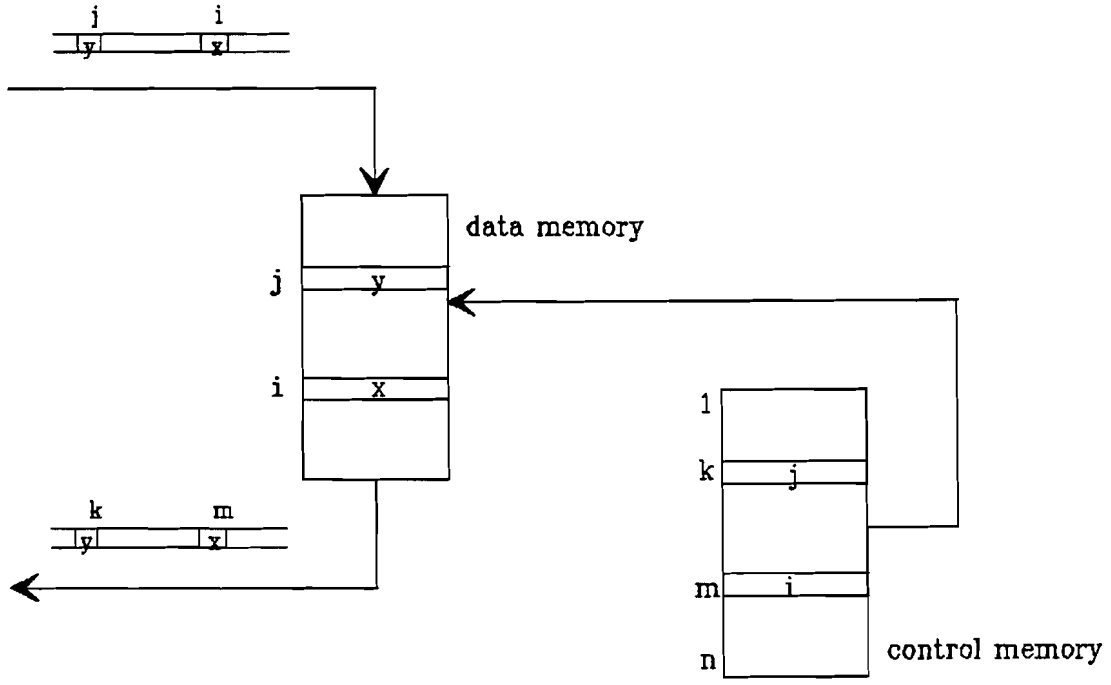


Figure 4.2 *Addressing and use of memory elements of the time switch element*

4.2.2 Implementation

Figure 4.3 shows the actual implementation of the time switch element (T1data) of the first stage of the TST network designed by [Timmer].

T1data works as follows.
The 4 bit wide encoded PCM data words are serial to parallel converted and written into RAMD. This bit stream has a rate of 2048 KBit/sec, so one bit time equals $1/(2048 \times 10^3)$ = 488 nsec. So in a period of 4x488 nsec (the samples are coded and have a width of 4 bits) we have to do one write operation to the DRAM, and two read operations from the DRAM. After a coded speech sample is stored in DRAM, we do two read operations from the DRAM. In this way the internal number of timeslots is doubled, and the switching network is non-blocking. The PCM samples are encoded (because of the (4,2)-concept), so the width of the DRAM is 4 bits. We specified ingoing PCM lines on a double primary rate frequency. Ignoring signalling information this results 64 PCM channels, so the DRAM has a length of 64 words of 4 bits.

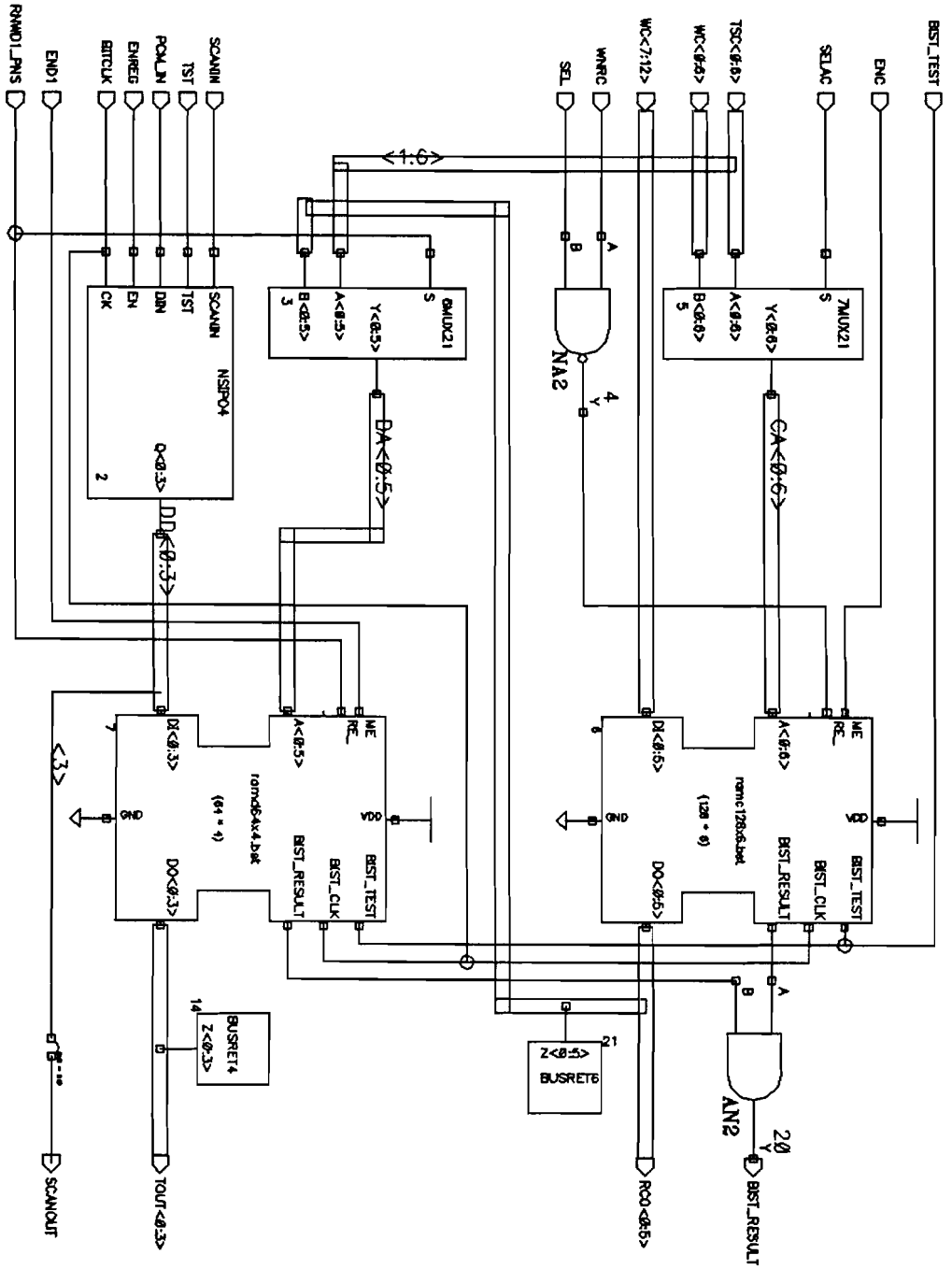


Figure 4.3 *Schematic of T1data*

The incoming PCM samples are written into RAMD in the address which is denoted by parts of the Time Slot Counter (TSC).
The timing of T1data is shown in figure 4.4.

We already have seen that we have to double the number of internal time slots to make the network non-blocking. Therefore in 4x488 nsec we do two reads from the DRAM. These read addresses are read from the CRAM. The CRAM is read in sequential order, so by changing the contents of the CRAM we can control the time switch operation and we can set up and break down connections.
CRAM consists of 128 words (the number of internal time slots) and each word has a width of 6 bits (i.e. $^2\log$ (#PCM channels) = $^2\log 64 = 6$.
For a detailed description of T1data we refer to [Timmer] page 18.

Figure 4.4 illustrates the timing of the time switch element running at 2 MBit/sec (ingoing PCM lines of the exchange are running on 4 MBit/sec).

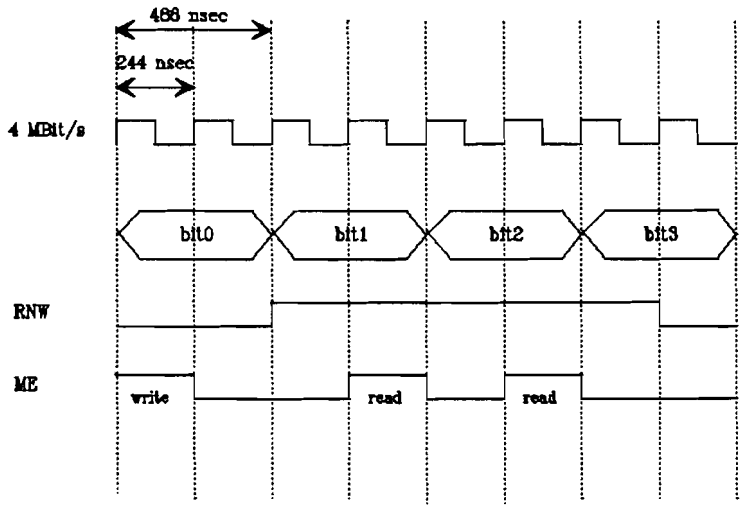


Figure 4.4 *Timing T1data, 4 MBit/sec time switch element*

We remark that the PCM data has to be written in RAM, and has to be read from the RAM. The timing of the RAM is mainly characterised by the access time when reading RAMs. Or in other words, we expect the access time of the RAM to be the delimiting factor when specifying the timing of the time switch element.
From the datasheets of the RAM blocks generated in SDA we can see that worst case access time of all RAM is about 25-30 nsec. The largest RAM block (RAM 1024x9) has a worst case access time (Timing Information, military) of 31.3 nsec

With these observations kept in mind, the following timing diagrams were specified.

4.2.3 Increasing the capacity

4x64 PCM channels

By quadrupling the clock frequency we can increase the capacity of the time switch element to 4x64 PCM channels. Figure 4.5 illustrates the timing diagram of a Time Switch Element running at 16 MBit/sec.

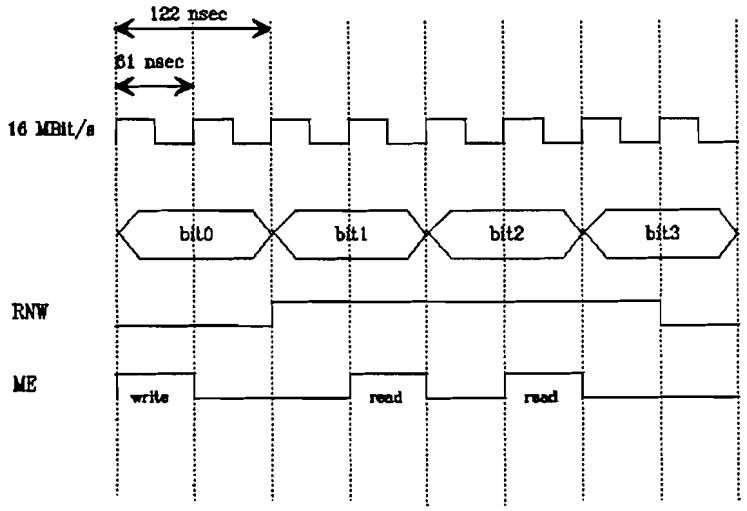


Figure 4.5 *Timing 16 MBit/sec time switch element*

It should be noted that no attention has been paid to critical path analysis. At the time of the research the actual schematic entry and simulation of the time switch element was not finished yet. After simulating the design a critical path analysis will have to be done to check the delay of control and other signals. However with a clock with intervals of 61 nsec, and with careful design of the logic functions (i.e. restricting the logic depth of logical functions) it should be possible to make a time switch element with a capacity of 4 x 64 PCM channels.

8 x 64 PCM channels

The next step in order to increase the capacity of the time switch element might be to increase the clock frequency to 32 MBit/sec.
However due to the timing of the RAM this will cause severe problems. Figure 4.6 shows the timing diagram for a 32 MBit/sec time switch element.
Due to the access times of the RAMs (between 25 nsec and 30 nsec typically) it would not be safe to do a read operation in one clockcycle. Therefore one read needs two clock cycles. If we assume that a write operation can be done in one cycle then the total timing still doesn't fit in the available time interval. The datasheets of the RAMs of the ES2 RAM cell generator specify that when reading the RAM the RNW should be high before the ME goes high.
A 'safe' 32 MBit/sec RAM timing therefore needs at least 9 periods in 244 nsec, this means a clock interval of 32 x (9/8) = 36 MBit/sec. It is clear that a design with these clock requirements (i.e. a 32 MBit/sec clock and a 36 MBit/sec clock) has its drawbacks.

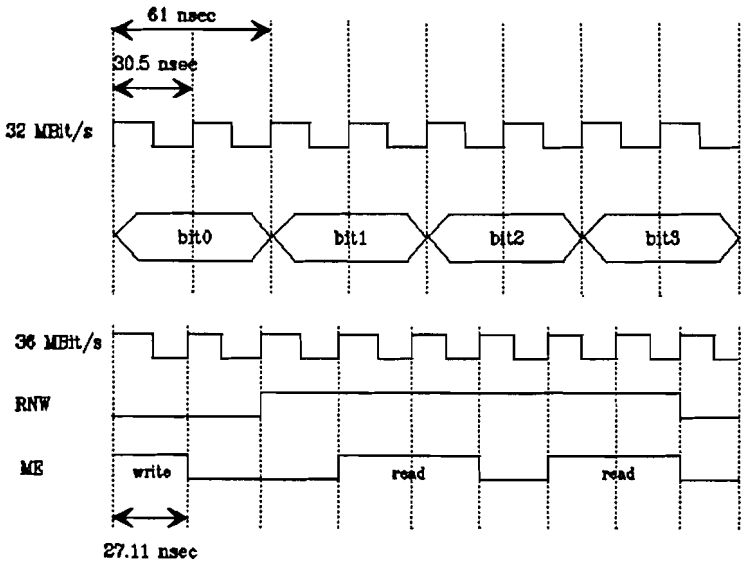


Figure 4.6 *Timing 32 Mbit/sec time switch element*

Another possibility of implementing a time switch element running at 32 MBit/sec would be of using interleaved RAM
The concept scheme of such an architecture is shown in figure 4.7.

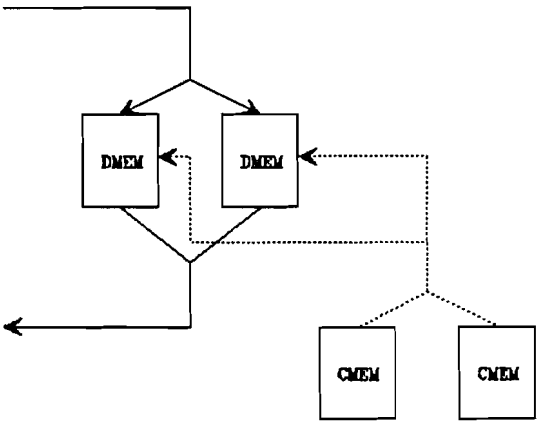


Figure 4.7 *Concept schematic interleaved RAM time switch element*

This concept needs further research.

4.3 Parallel Time Switch Element

4.3.1 Introduction

After an extensive literature search a new concept in time switching was found [Nikaido].
Based on the observation that the time switch element proposed operates in parallel mode and is built according to a pipeline architecture it was decided to design and simulate such a time switch element.
The basic idea of the parallel time switch element is illustrated by a parallel time switch element that can interchange two time slots of 4 bit codewords.
The time switch element operates as follows. The incoming time slots are serially clocked into the flip flips on the left side. When the whole frame has arrived, all samples are clocked into the second row of flip flops. Then the samples are gated through the multiplexers, and clocked into the third row of flip flops.

The sequence of time slots is determined by the SEL signal. This value (in this case SEL is only one bit, in the case of four speech samples SEL is 2 bits, (in general SEL = $^{2}\log$(# PCM-channels)), is read from the control RAM.

4.3.2 Implementation

Schematic Entry

The actual implementation of the parallel time switch element is shown in appendix 2.
Although the top down concept is widely used when designing VLSI circuits, the time switch element was implemented bottom-up. The main reason for this approach is that the available software (i.e. the SDA silicon compiler) only supports bottom-up schematic entry. Therefore we have to start at the lowest level. Figure 4.9 shows the schematic of a so-called Mux2. On the left side are two registers through which PCM samples are serially clocked in. Next to them we see two registers into the whole PCM frame is clocked. On the right side we see the registers to which the outgoing samples are clocked through the multiplexers. The order of outgoing samples is determined by the multiplexer, which is controlled by the SEL-signal.

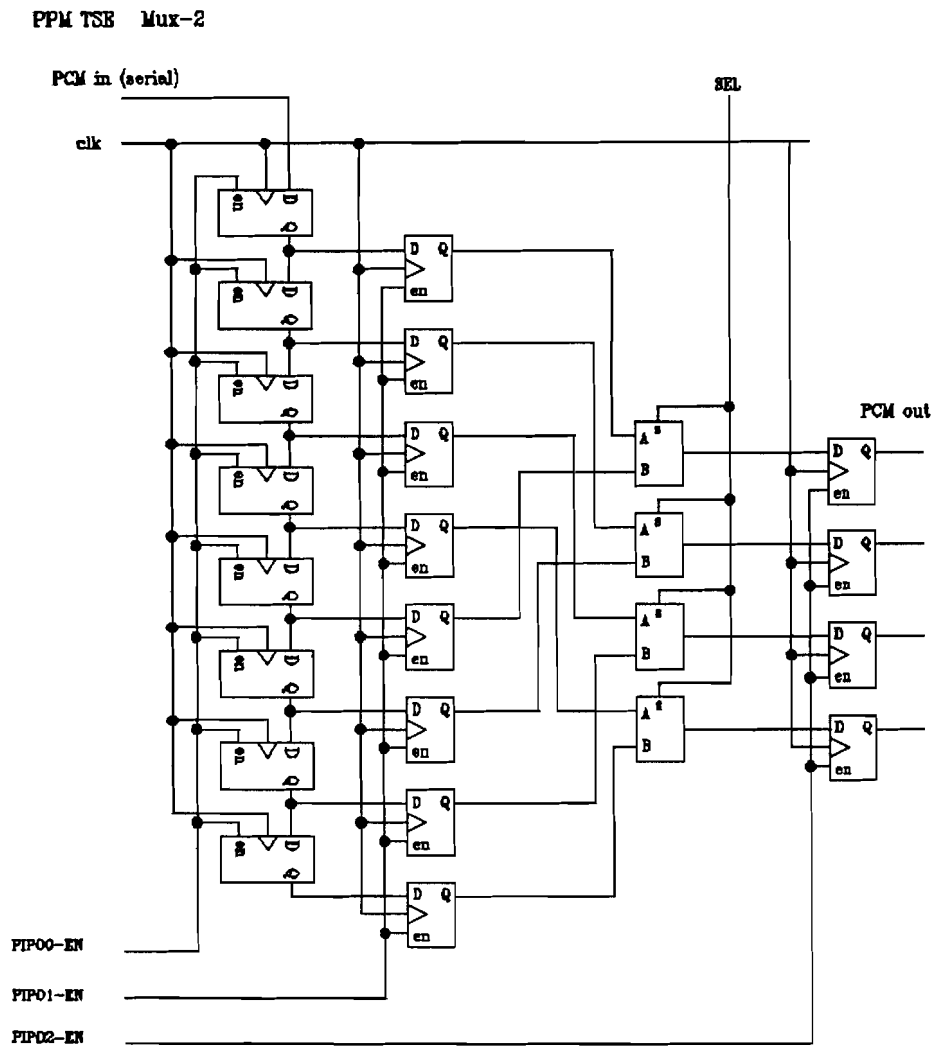


Figure 4.8 *Parallel time switch element that can interchange two timeslots*

Figure 4.10 shows a Mux4, which consists of two Mux2 elements, and a single multiplexer and a register.

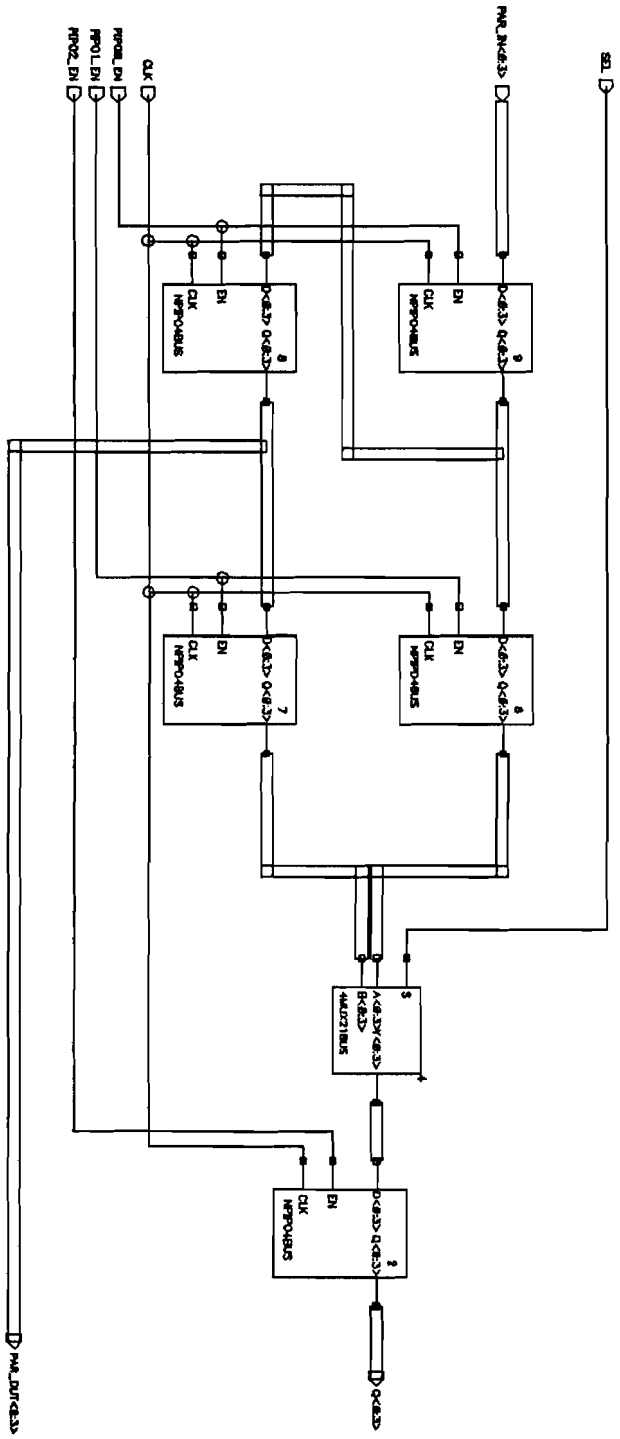


Figure 4.9 *Mux2*

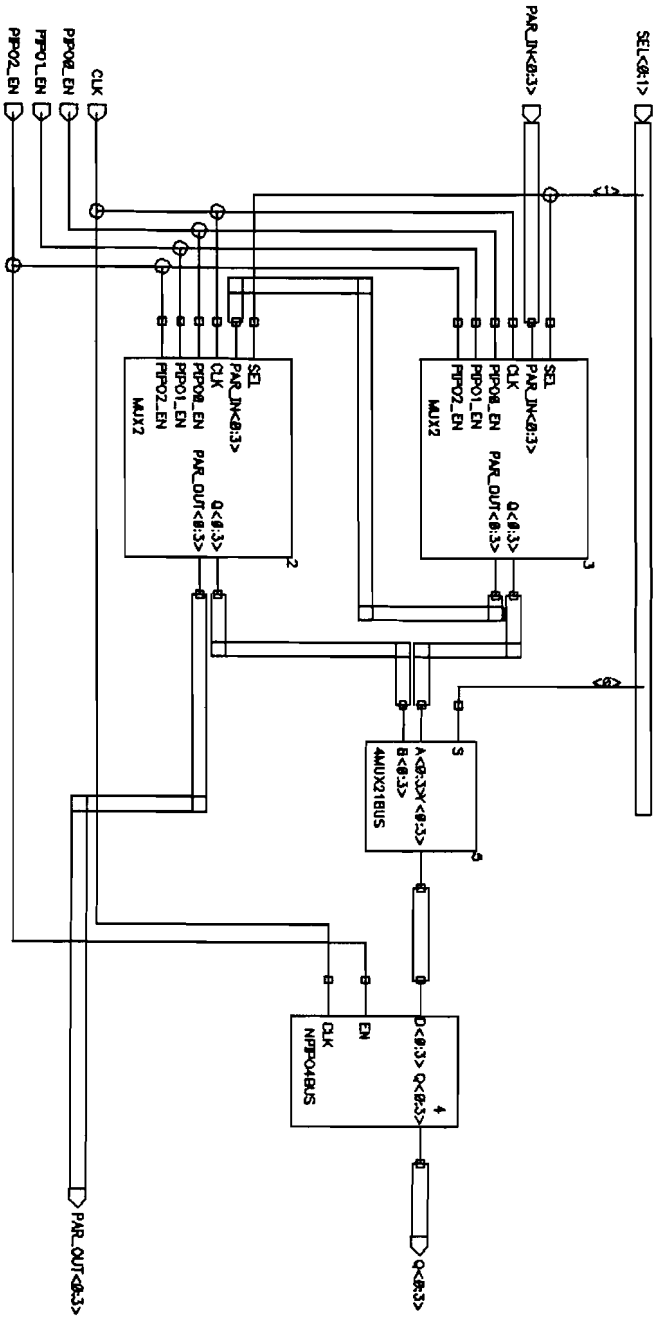


Figure 4.10 *Mux4*

Exploiting the recursive structure of the design, we can simply built a Mux32, which is shown in figure 4.11

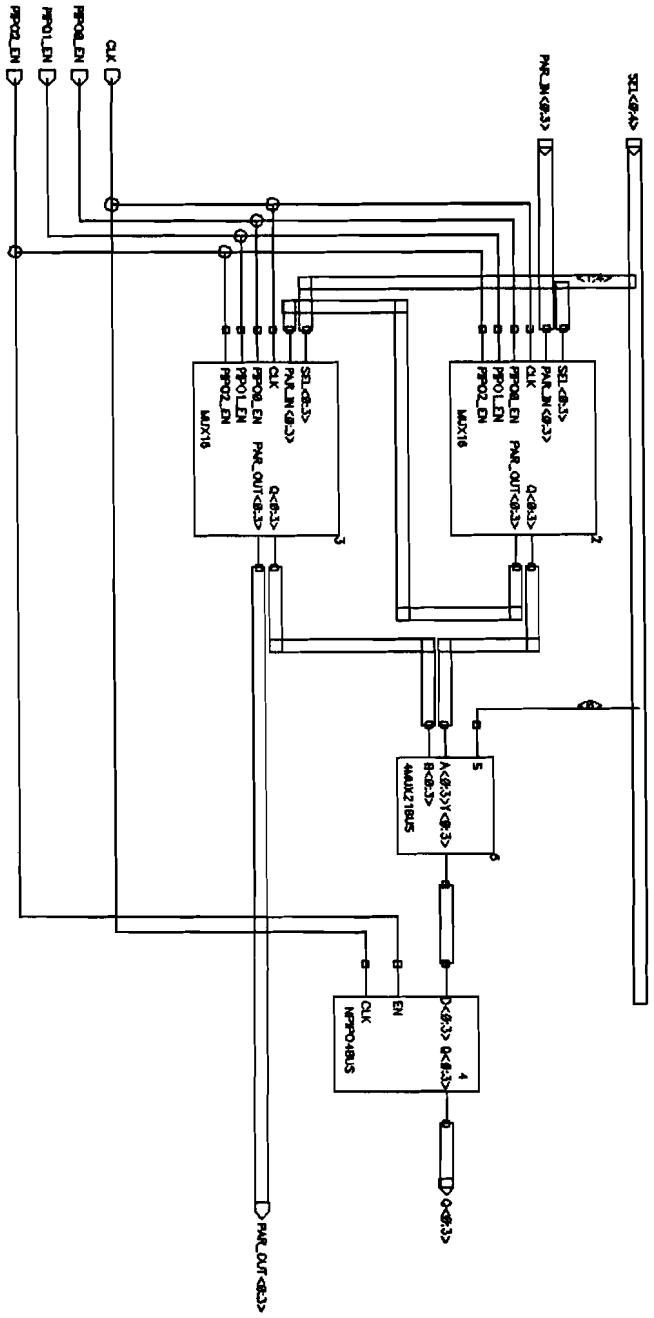


Figure 4.11 *Schematic Mux32*

We already pointed out that the switching is controlled by the SEL bus. This means that the sequence of outgoing time slots is determined by the SEL signals. Therefore this signal is read from the control memory of the parallel time switch element.

Because of the parallel mode of operation of the time shifter we should keep in mind that we cannot simply use the RAM as in the traditional time shift element.
Figure 4.12 shows the contents of the control RAM when it is read out sequentially and we want to place time slot 0, 1, 2, 3, 4, etc on the outgoing PCM line.
Because of the 'pipelined' way of operation the pattern of the SEL signal must be as shown in figure 4.12.

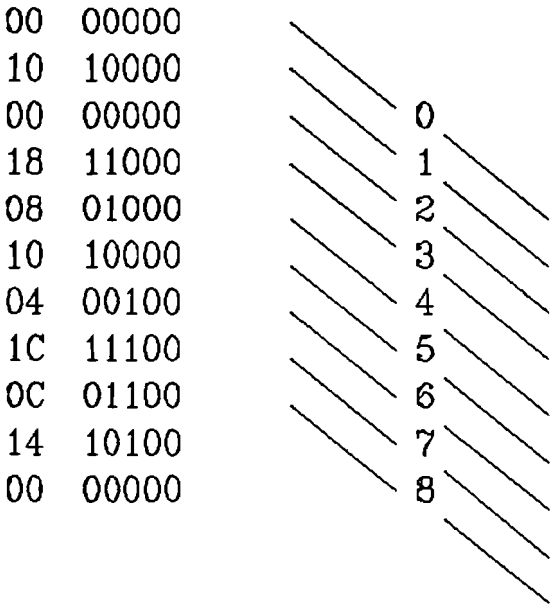


Figure 4.12 *Contents of Control RAM*

It is clear that it is at least 'inconvenient' to arrange the contents of the CRAM in this way. If we want to set up a new connection we have to do a number of read and write operations into the CRAM. Therefore module SEL is introduced, which delays the signals of the SEL bus in an appropriate way. In this way the CRAM can be updated in one write cyclus.

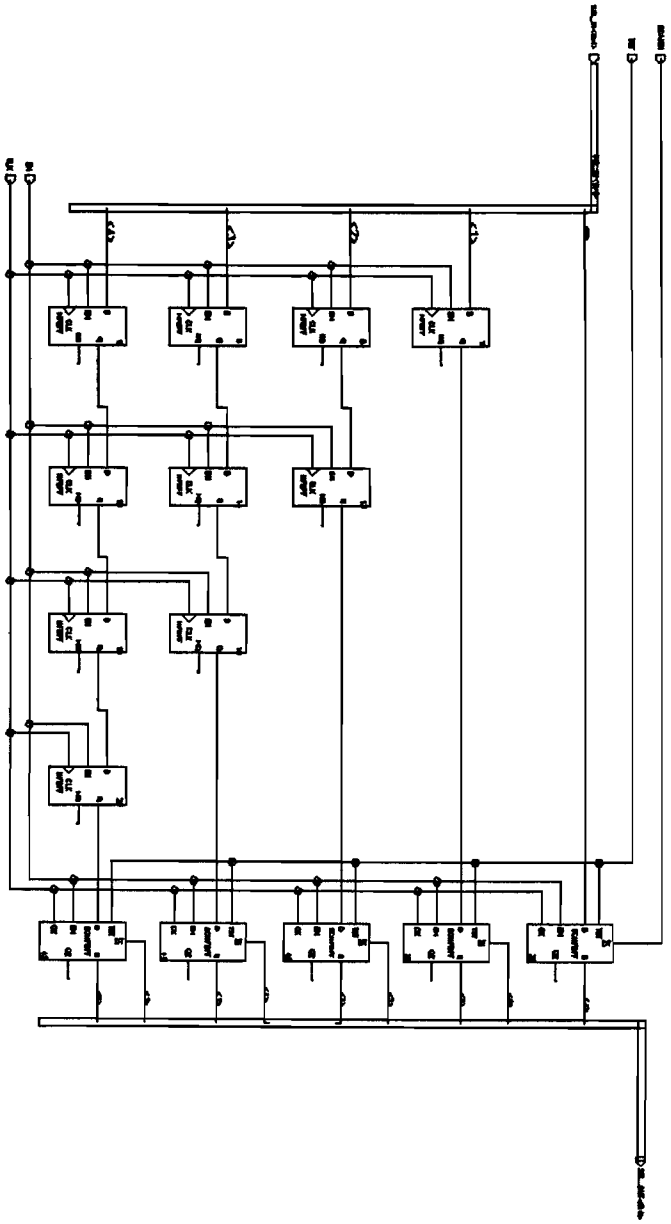


Figure 4.13 *Module SEL*

Top level design

The resulting top level design that has been simulated is shown in figure 4.14

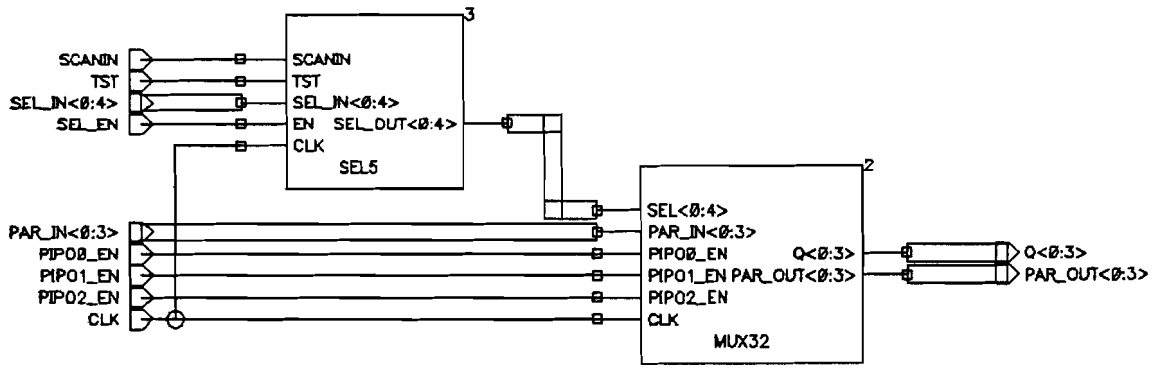


Figure 4.14 *Mux32test*

Simulation problems

Simulating the design on various levels of the architecture caused the following problems

Mux32 fan out values of various nets have unrealistic values

Mux64 the design can be netlisted, bus the simulation stops with error report 'memory fault'

See appendix 3 for the error reports.
These problems are most likely caused by the fact that the hardware platform (i.e. the Apollo workstations) has a stack size which is fixed on 256K. Netlisting and simulating the design runs the Apollo out of its fixed stack. This problem is already known at Apollo's but it is unknown whether this bug is fixed in newer releases of the UNIX operating system. This problem was also reported to Pijnenburg, who are now trying to run the simulations on Sun workstations. The results of these efforts are also unknown yet. See appendix 4 for the SDA service reports.

Simulation results

Although the schematic is very simple, the simulation of the design caused many problems, and some of these problems are still not solved.
Yet we can remark the following.

The time switch element operates correctly. Functional simulations show that samples are time switched correctly.
But due to the recursive architecture the size of the circuits 'explodes' rapidly. The 'explosion' of circuit size is shown in table 4.1.

Table 4.1 *Explosion of circuit size*

circuit	#transistors	#nets
Mux2	804	201
Mux4	1804	451
Mux8	3804	951
Mux16	7804	1951
Mux32	15804	3951

Another drawback of the design is that due to the number of flip flop elements the fan out of the clock signal and some enable signals also explodes.

Table 4.2 shows the size of a parallel time switch element.

Table 4.2 *Size of Mux64*

ES2 elements	size
764 DFF	2.7 10^{-6} m^2
892 MUX21	6.3 10^{-6} m^2
1 RAM 128x6	1.5 10^{-6} m^2
total	10.5 mm^2

Based upon the size of ES2 library elements, and based upon the total number of transistors (which is given after simulating a design) we could say that each transistor has a size of about 2.7 10^{-10} m^2.
The size of T1data is (roughly) as follows.

Table 4.3 *Size of T1data*

ES2 elements	size
1 RAM 128x6	1.6 10^{-6} m^2
1 RAM 64x4	7.2 10^{-7} m^2
1924 transistors	1.5 10^{-6} m^2
total	2.7 mm^2

We see that a Mux64 time switch element is much larger then T1data. However by careful redesign of the time switch element the number of multiplexers can be decreased dramatically.
In the current design of the parallel time switch element all memory elements are connected to the same clock signal, and the actual switching is controlled by an enable signal. In order to design the memory elements synchronously, on every rising clock either the old data is clocked into the memory element (flipflop) again, or new data (when enable is high) is clocked in. Therefore each memory element consists of a flipflop and a MUX21. By introducing a second clock signal, on which all flipflops are clocked, the number of MUX21 can be decreased, and the total chip area of these elements can be decreased to from 6.10^{-6} m^2 to 1.10^{-6} m^2.

4.4 Conclusions

Although all simulation runs could not be completed we can remark the following. We have seen that by increasing the clock frequency it is likely that traditional time shift element can run 16 MBit/sec, and therefore can time shift 4x64 PCM channels.
With current technology a time shift element that runs on 32 MBit/sec will be very hard to implement.
By adopting another architecture based upon pipelining and multiplexing a new time shift element was designed.
Due to problems with the available silicon compiler this time switch element could not be fully simulated. Therefore we cannot say what the maximum operating frequency will be. Despite its conceptually simple structure we expect that the new architecture cannot switch much more calls than the traditional time switch element. It may be more suited to switch on higher frequencies, because of the fact that no data memory is used. But by increasing the clock frequency also the number of speech samples is increased. Due to the recursive structure of the design the total circuit size of the parallel time switch element becomes very large. It will be very hard to distribute the clock signal over the large number of flipflops.

5 Space Switch Element

5.1 Introduction

In this chapter we will discuss how we can increase the capacity of the space switch element of the central switching network. The original architecture of the TST network designed by [van Lier] and [Timmer] is shown in figure 5.1.

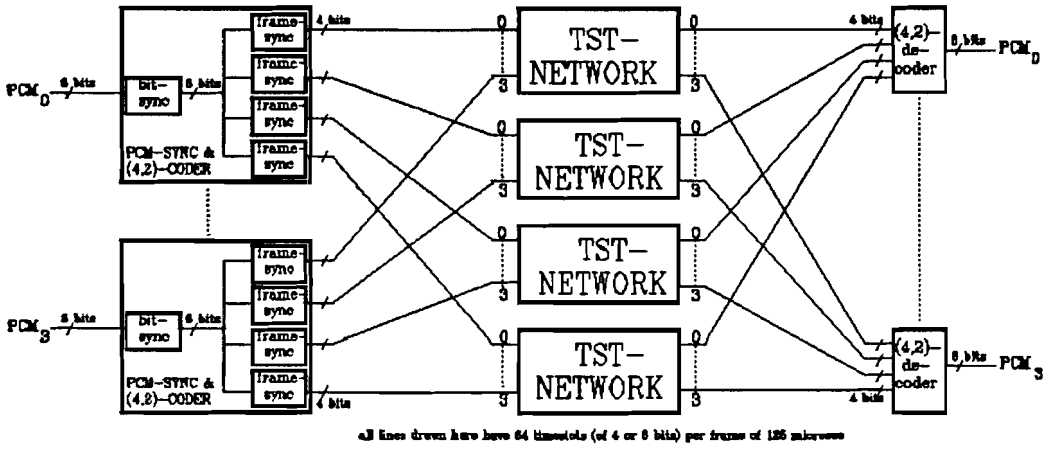


Figure 5.1 *(4,2) exchange*

The space switch element is implemented by using demultiplexers, so we can also illustrate the design as follows

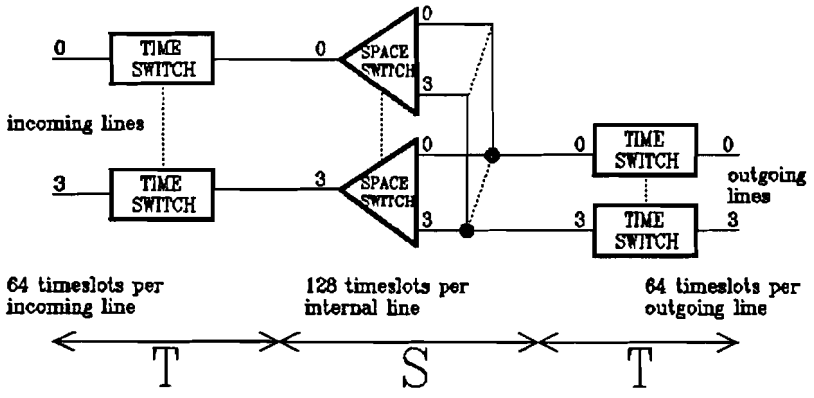


Figure 5.2 *TST network*

5.2 Traditional Space Switch Element

Figure 5.3 illustrates the space switch element designed by [Timmer], which is based upon 4 ingoing and 4 outgoing PCM lines.

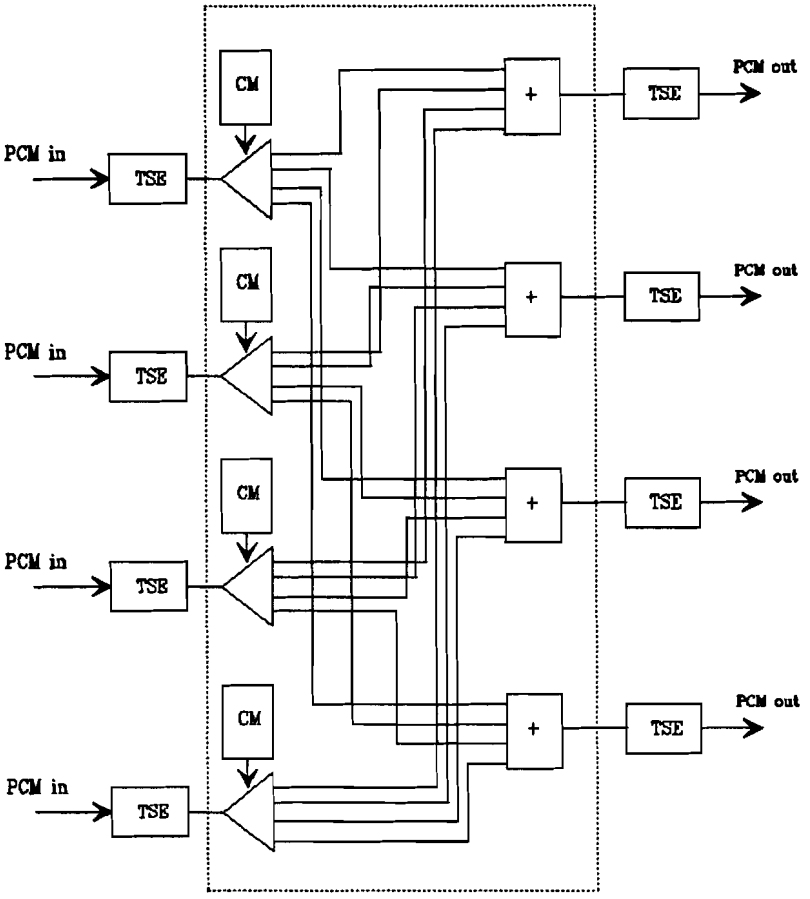


Figure 5.3 *4x4 space switch element*

The interconnection of the ingoing PCM lines to the outgoing PCM lines is done by demultiplexers and or-gates. In the case of a 4x4 space switch element this results in a wiring path of 4*4 = 16 lines.

Wiring problem

In the case of a 64x64 space switch element the wiring will have a width of 64*64 = 4096 lines, which is not feasible in current CMOS technology.
Based on these observations it is expected that a 16x16 space switch element is the largest possible space switch element that can be made.

Beside the specific wiring problem we should remark that the wiring of an IC is becoming a major problem in VLSI design. Therefore the architecture of the space switch element was reconsidered and another architecture, based upon a time shared bus was designed.

5.3 Shared Bus Space Switch Element

In order to avoid the wiring problem discussed in paragraph 5.2 an architecture based upon a time sharing bus has been designed.
Figure 5.4 illustrates the basic idea of such a space switch element.

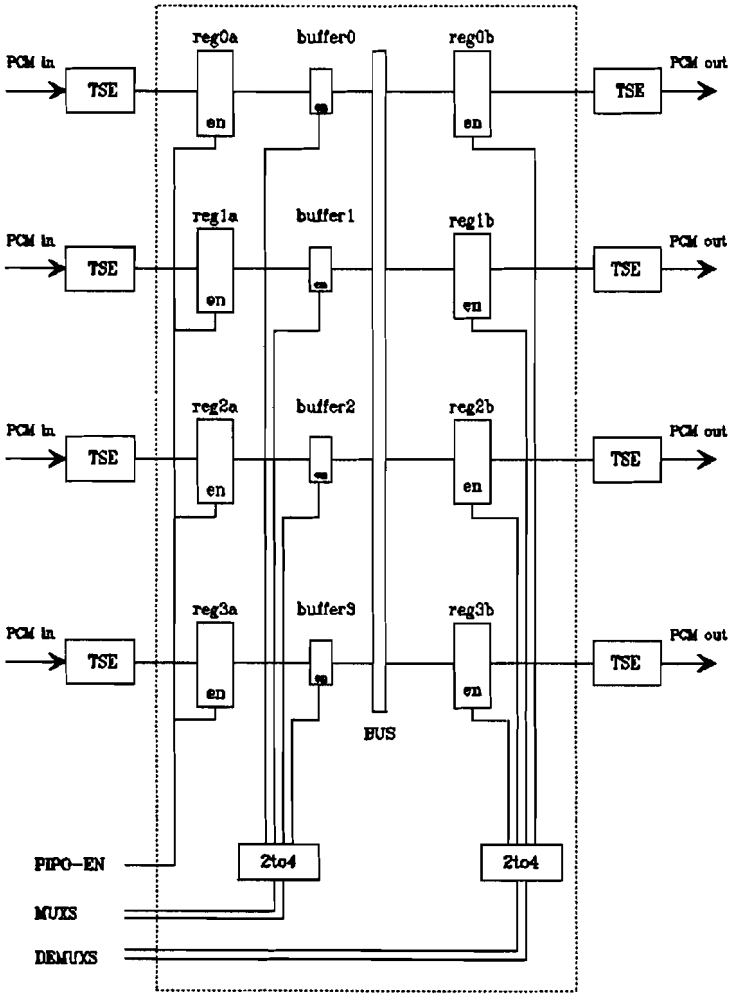


Figure 5.4 *Time shared bus space switch element*

This figure shows a 4x4 space switch element. Incoming and outgoing PCM lines are timeswitched in the time switch elements. Space switching is done by writing PCM data to the bus (e.g. enabling buffer0 .. buffer3) and reading from the bus (e.g. enabling register0 .. register3). In one space switch cyclus we have to do 4 reads, so the bus is time shared by the four ingoing PCM lines. The flow of the PCM data is controlled by MUXS and DEMUX signals. One of these signals counts from 0 to 3, and the other signal is read from the Control Ram, in which the connection status of the space switch element is stored.
The timing of a 4x4 space switch element is given in figure 5.5

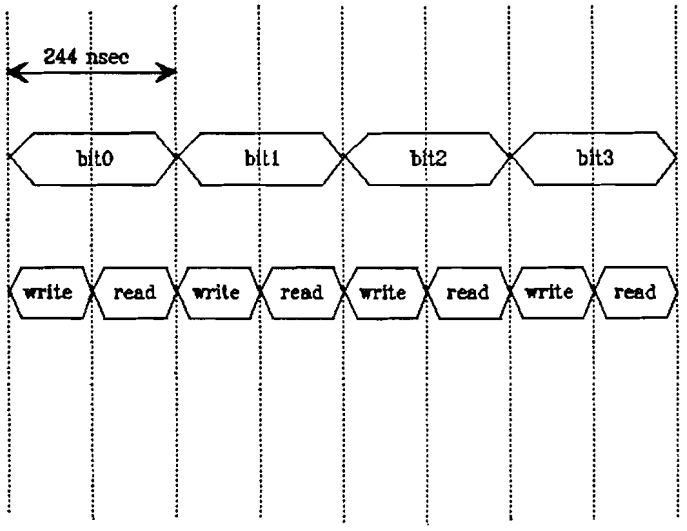


Figure 5.5 *Timing 4x4 time shared bus space switch element*

In one space switch cyclus (which lasts 4x244 nsec) we have to make 4 connections. Therefore we have to do one read from the control memory every 244 nsec. In order to break down and set up new connections we also should be able to write the Control Memory. In this case we can write the Control Memory every 244 nsec.
Due to the fact that the bus is time shared between the PCM lines it is important to analyze the timing of larger space switch elements.
As we can see in figure 5.5 a 4x4 time shared bus space switch element there are no timing problems. In a period of 122 nsec we can implement a read or write operation. But it is likely that when we increase the number of PCM lines timing problems might occur.

Figure 5.6 shows the timing diagram of a 8x8 time shared bus space switch element. In a period of 244 nsec two PCM samples have to share the bus. Therefore we have to read the Control Memory every 122 nsec. Because we also have to update the Control Memory, read cycli are also necessary.

A configuration of the Control Memory that meet these timing requirements is one with interleaved RAM blocks.

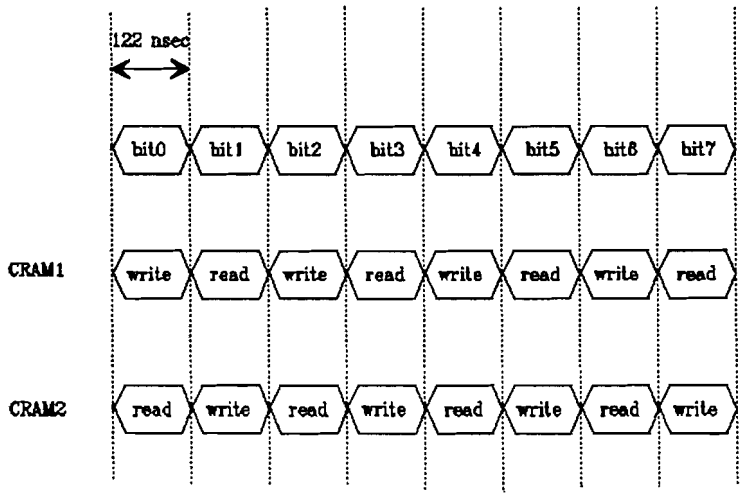


Figure 5.6 *8x8 time shared bus space switch element*

In appendix 6 timing diagrams of a 4x4 time shared bus time switch element with 4x64 PCM lines multiplexed to 16 MBit/sec and of a 8x8 time shared bus time switch element with 4x64 PCM lines multiplexed to 16 MBit/sec are given. The first needs a RAM configuration that consists of two interleaved RAM blocks, whereas the latter needs 8 interleaved RAM blocks.
This clearly shows that a space switch element with a time shared bus is not really a good solution for the wiring problem.
We could propose a space switch element with multiple time shared buses, see figure 5.7

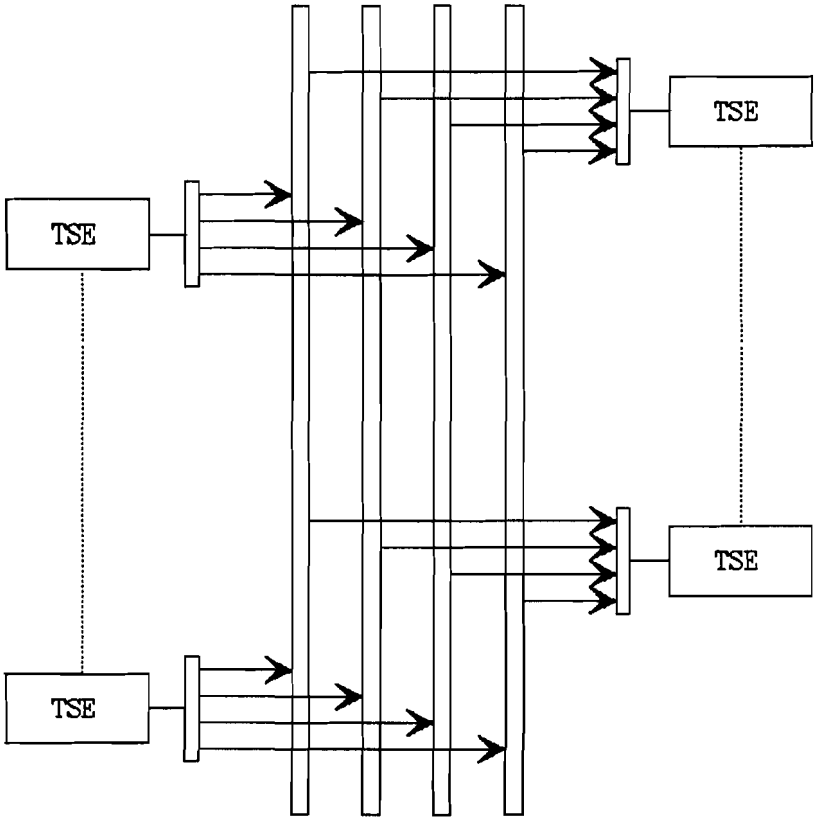


Figure 5.7 *Concept schematic multiple buses space switch element*

We expect the writing to and reading from the different buses to introduce a rather untransparant structure. This architecture needs further research.

5.4 SSS-network

An usual approach to increase capacity of a switching network is to increase the total number of switching stages.
The maximum number of PCM lines that can be time switched in one single time switch is limited by the maximum allowable delay between incoming and outgoing PCM samples. More PCM can be switched by introducing a second switching stage, e.g. a space switch element.

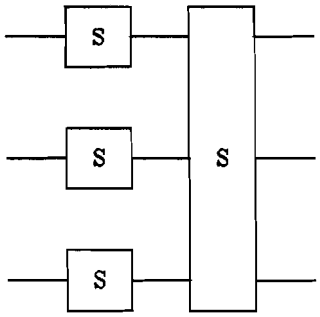


Figure 5.8 *Two stage network*

By adding a switching stage we increase the capacity of the switching network. But in this way the network is not non-blocking anymore.
This means that the situation can occur (and therefore also will occur, according to Murphy) that the network cannot interconnect an incoming timeslot to an outgoing timeslot.
Because we would like to have a non blocking switching network, a third stage is added.

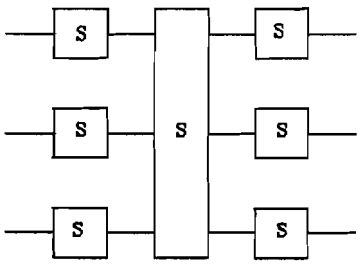


Figure 5.9 *Three stage network*

In this way the switching network is proven to be non-blocking.

With this idea kept in mind another design of the space switch element was made, namely a SSS-network.

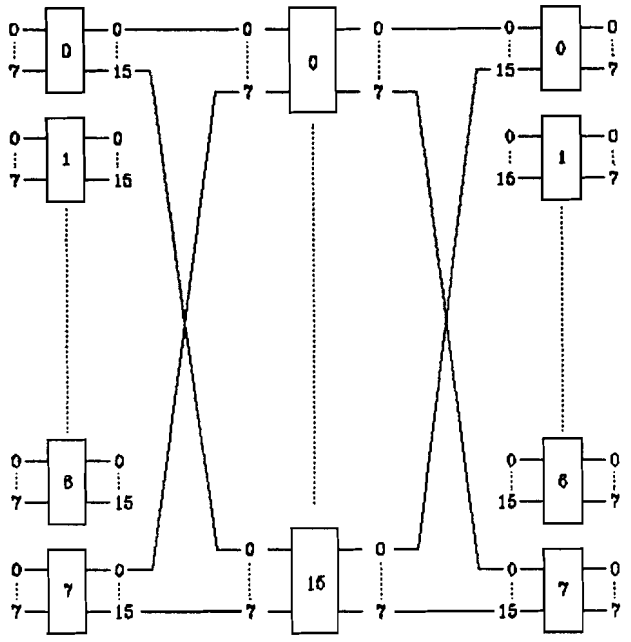


Figure 5.10 *SSS network*

Figure 5.10 shows a - what we call - 8-16-8 SSS network. 8-16-8 means 8 space switch elements in the first stage, 16 in the second and 8 in the third stage. In order to keep the network non-blocking the internal number of connections must be doubled.

Restrictions

In this case we restricted ourselves to a 64x64 space switch element (to solve the wiring problems denoted by [van Lier] and [Timmer]).
Therefore the goal was to design an 'optimal' architecture for a 64x64 space switch element. Optimal here means at least 'feasible'. Furthermore we restricted ourselves to four possible SSS network configurations, namely a 4-32-4, an 8-16-8, a 16-8-16 and a 32-4-32 architecture.

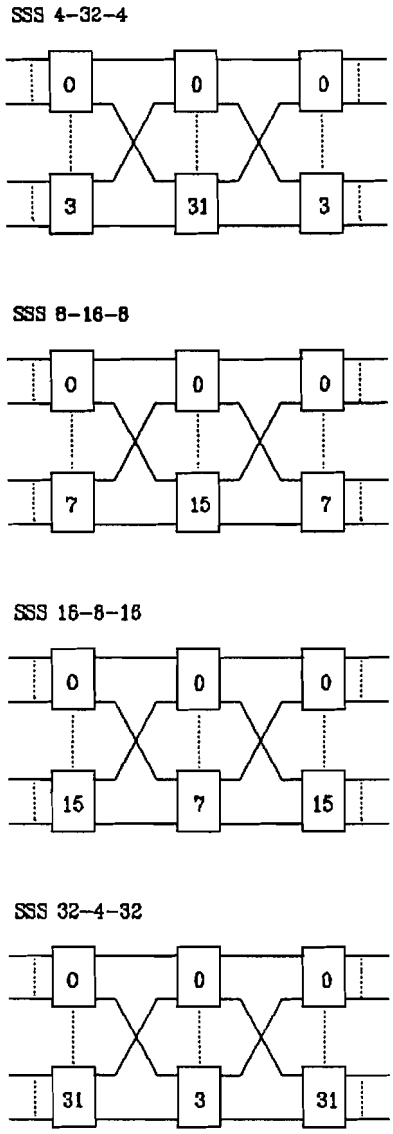
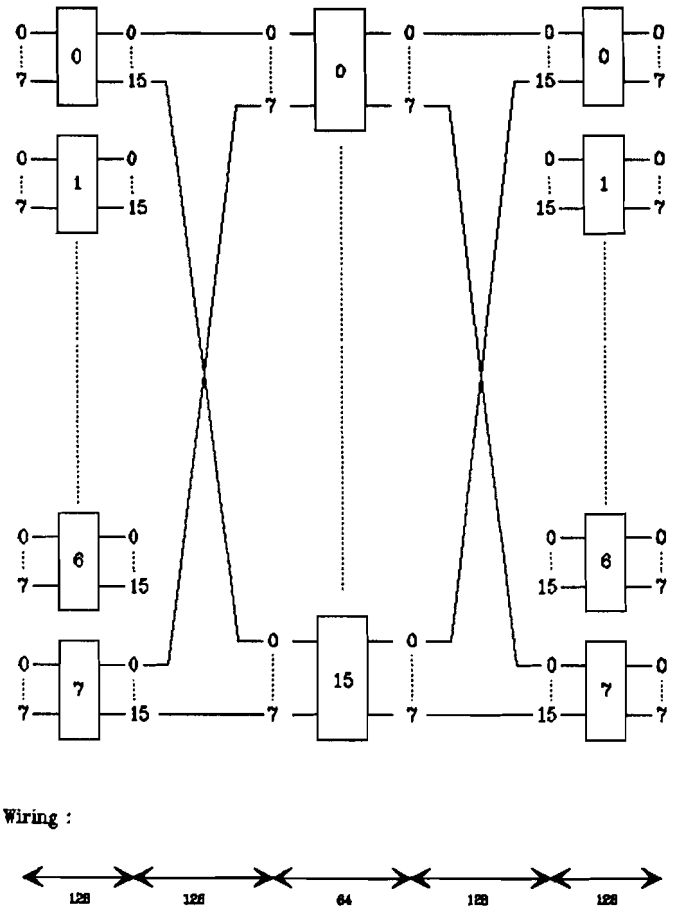


Figure 5.11 *SSS networks overview*

Figure 5.12 shows again the architecture of a 8-16-8 network. In this figure the wiring inside and between the space switches is shown. Also the size of the Control Ram is shown.

The wiring width of the left space switch is calculated as follows: 8 incoming lines must each be multiplexed to 16 outgoing lines, i.e. 8x16 = 128, therefore the maximum width of the wiring between first and second s stage is also 8x16=128.
The size of the RAM is calculated as follows: in 8 S stages are 128 timeslots on 8 incoming lines that must be switched to 16 (^{2}log16 = 4 bits memory) outgoing lines Therefore the size of the RAM is 8x128x8x4.



RAM:

8 x 128 x 8 x 4	16 x 128 x 8 x 3	8 x 128 x 16 x 3
8 x 128 x 32	16 x 128 x 64	8 x 128 x 48
8 x 4096	16 x 3072	8 x 6144
32.766	49.152	49.152

Figure 5.12 *SSS 8-16-8 network*

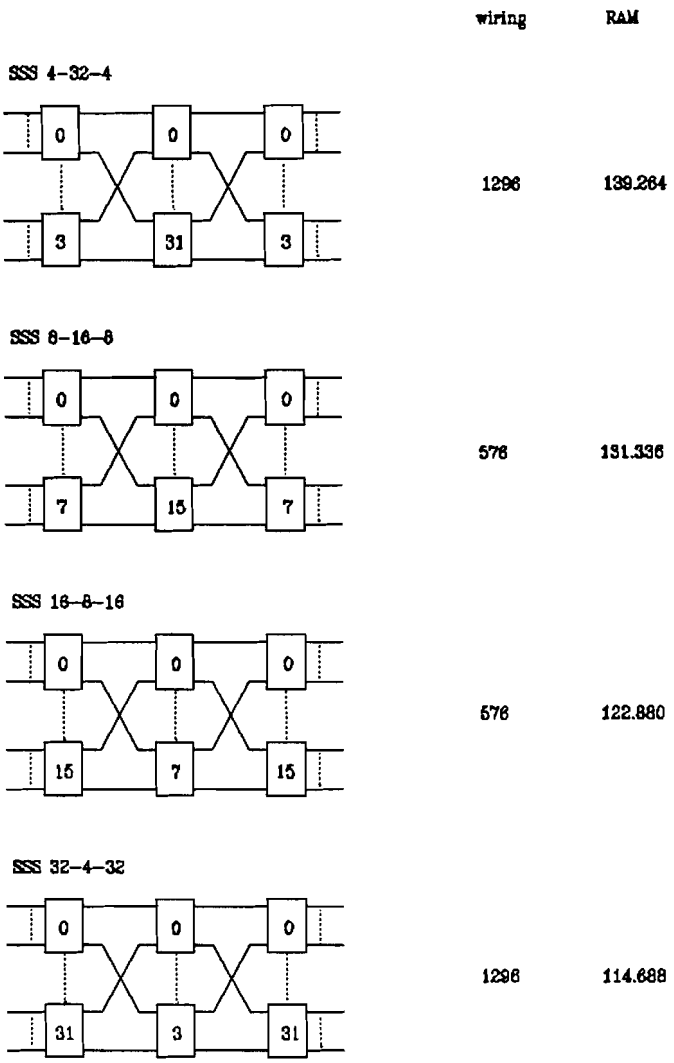


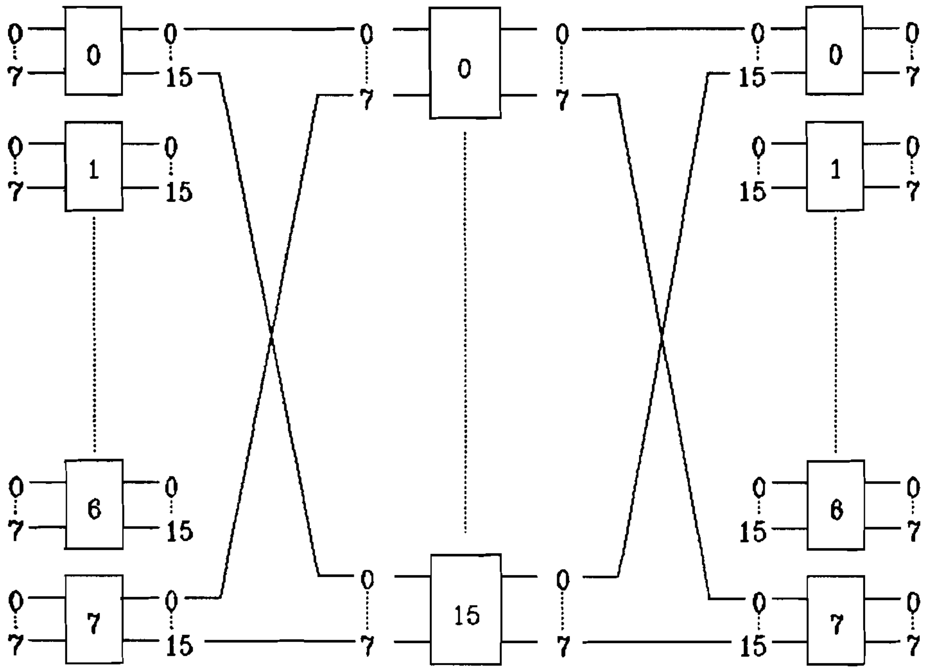
Figure 5.13 *SSS networks trade off*

Based upon these two figures, namely the size of the wiring and the amount of RAM a trade off of the SSS networks was made.
As we can see from figure 5.13 a SSS 8-16-8 network yields the best wiring and RAM ratio.

See appendix 7 for the top down design of the SSS 8-16-8 network.

5.5 Conclusions

Figure 5.14 shows the results of the design of the SSS network.



RAM		
8 x 128 x 32	16 x 128 x 24	8 x 128 x 48
43.52 mm^2	66.88 mm^2	65.52 mm^2
transistors		
17556	37564	32488

totals:
176 mm^2 RAM +
67606 transistors +
wiring +
control logic

Figure 5.14 *SSS 8-16-8 network architecture details*

As we can see in 1.5 u CMOS technology the total chip area is very large.
We are convinced that a SSS-network is a better architecture for a 64x64 space switch element than a single space switch element.
The advantages and disadvantages of this architecture are that the wiring has been distributed over the whole chip. The precise effects of this distribution can only be known after the lay-out of the chip has been made. This has not been done yet.
On the other hand the distribution of the switching has some major disadvantages.
We will see in chapter 6 that when we increase the capacity of the switching network, the total amount of RAM also increases. We will also see that by integrating various RAM blocks into single RAM-modules we can further increase the capacity of the network. Therefore it is very difficult to make a trade-off between on the one side integration and on the other side a distributed approach that may be more suited to handle the total wiring of the chip.

6 Expected chip area of TST and TSSST networks

6.1 Introduction

Now we have seen how we can increase the capacity of time and space switch elements it is important to see how we can increase the capacity of the whole switching network by using these switching elements.

We will illustrate this by making some calculations of the expected chip area, and by describing the architecture, the expected chip area and the capacity of a TST and a TSSST network.

6.2 Expected Chip Area

In order to make a rough guess of the feasibility of the proposed and partly designed circuits we have made a rough guess of the expected chip area.
We should note that lay outing and wiring has not been done yet. All figures may be subject to - may be even major - changes.
We can see from table 4.3 that the size of the RAM is a main factor in the total chip size. Moreover, the size of the RAM is the only figure that is automatically generated from the SDA software.
In table 4.3 we also made a guess of the size of all logic elements. Based upon the size of ES2 library elements, and based upon the total amount of transistors (which is given after simulating a design) we said that each transistor has a size of about
$2.7\ 10^{-10}\ m^2$. However we expect that when we use the small library elements there will be much wiring, so this figure is rather unreliable. Therefore all calculations of chip area will be made based upon RAM size.
Table 6.1 gives an overview of the size of some RAM blocks.

Table 6.1 *Size of RAM blocks*

#words x #bits	total chip area	chip area per bit
128 x 2	0.75 10^{-6} m^2	2.9 10^{-9} m^2/bit
128 x 6	1.45 10^{-6} m^2	1.9 10^{-9} m^2/bit
64 x 7	1.13 10^{-6} m^2	2.5 10^{-9} m^2/bit
128 x 4	1.00 10^{-6} m^2	2.0 10^{-9} m^2/bit
64 x 4	0.7 10^{-6} m^2	2.78 10^{-9} m^2/bit
512 x 8	5.6 10^{-6} m^2	1.37 10^{-9} m^2/bit
1024 x 9	11.23 10^{-6} m^2	1.22 10^{-9} m^2/bit

Based upon the RAM generation results we remark that by increasing the size of the RAM the chip area per bit decreases. We could expect this.
The smaller RAM blocks yield a size of 2 10^{-9} m^2 per bit generated by the SDA ES2 RAM Cell Generator in 1.5 u CMOS technology.
According to the size of the 2 u SDA ES2 RAM Cell Generator estimated from the SDA manuals these RAMs have a size of about 6 10^{-9} m^2/bit

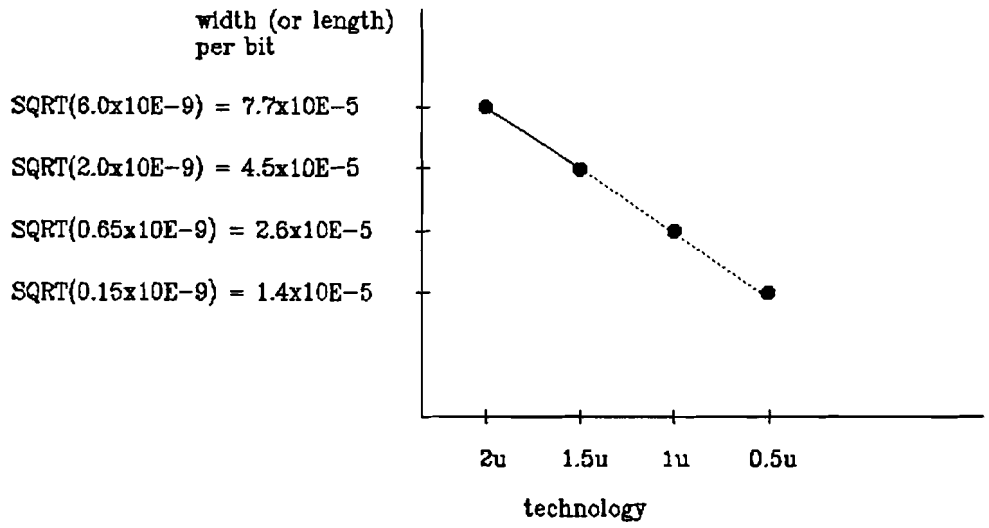


Figure 6.1 *RAM width (or length) per bit*

In figure 6.1 these two figures are extrapolated for 1 u and 0.5 u technology

6.3 TST Network

Based upon the research results described in chapter 4 and 5 we suggest a TST network as follows.
We feel that a 16x16 space switch element with demultiplexers and 'full' wiring should be feasible in current CMOS technology. However we have not done any lay-outing and wiring yet. When both lay-outing and wiring has been done it will be safer to do any suggestions about the wiring problem.
Based upon the timing analysis of the current time switch element, and based upon the timing of the ES2 RAM Cell Generator we think that a time switch element which is running at four times the speed of T1data of [Timmer] can be made successfully.

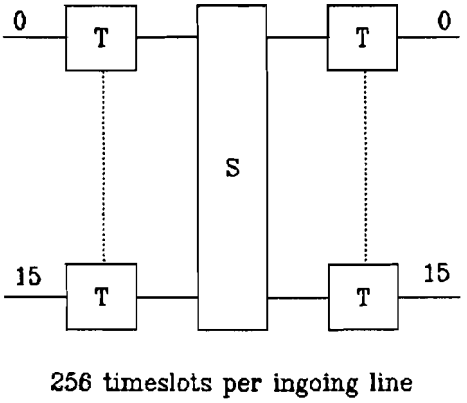


Figure 6.2 *16x16 256 channel TST network*

The capacity of the switching network is 64x4x16 (=4096) half telephone calls.
The expected circuit size of time switch elements can be estimated as follows.
As we already have seen the RAM occupies the greater part of the total chip size (wiring not included). Therefore we calculate the chip area of the RAM.

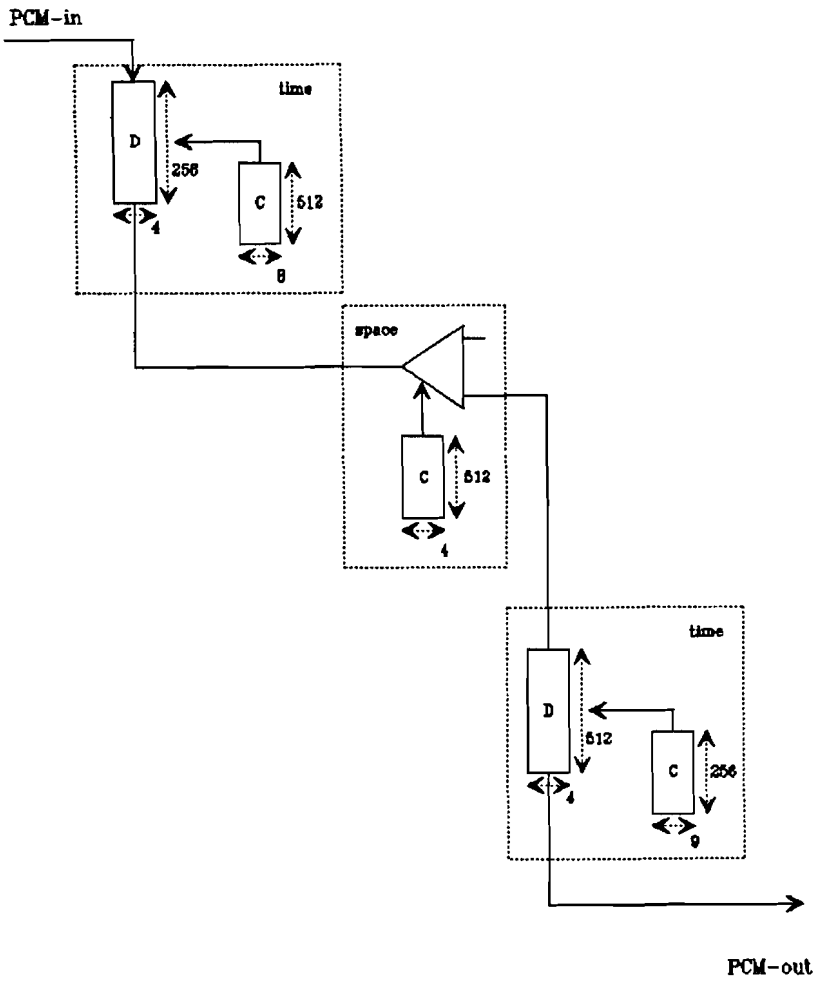


Figure 6.3 *RAMs of 16x16 256 channel TST network*

Table 6.2 *size of RAM of time, space and time switch elements*

technology	m2 per bit	size time	size space	size time
1.5 u	$2.2\ 10^{-9}$	11.3 mm^2	4.5 mm^2	9.6 mm^2
1 u	$6.5\ 10^{-10}$	3.3 mm^2	1.3 mm^2	2.8 mm^2
0.5 u	$1.5\ 10^{-10}$	0.76 mm^2	0.31 mm^2	0.65 mm^2

Table 6.3 *Size of RAM of 16 x 16 256 channel TST network per row*

technology	time	space	time
1.5 u	180 mm^2	72 mm^2	153 mm^2
1 u	53 mm^2	21 mm^2	45 mm^2
0.5 u	12 mm^2	5 mm^2	10 mm^2

Table 6.2 gives an overview of the size of the RAM of time and space switch elements of the TST network. We see that in 0.5u technology the chip area of the RAM of a time switch element is about 0.76 mm^2.
Table 6.3 gives the chip area of the RAM of a 16x16 256 channel (4096 Erlang) TST network per row. In 0.5 u all RAM of the first stage time switch elements yields 12mm^2, all RAM of the space switch elements yields 5 mm^2 and all RAM of the third stage time switch elements occupy 10 mm^2 chip area.

6.4 TSSST Network

In order to avoid the so-called wiring problem we suggested a SSS network in chapter 5.
With technology moving into the sub-micron area this architecture might be worthwhile.

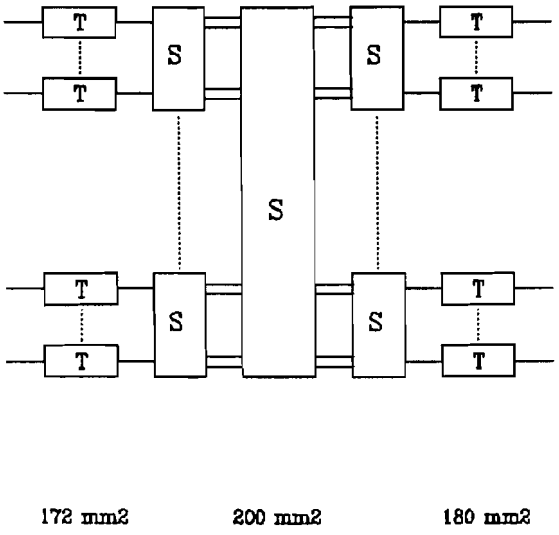


Figure 6.4 *TSSST network*

See appendix 8 for the calculation of the size of the SSS network.
We calculated that in 1.5 u technology the chip area of the SSS network is about 200 mm^2.

6.5 Configuration of the RAM

In order to optimize the design [Timmer] proposed an architecture with only 5 large RAM blocks.
At a first sight this architecture has some nice features. However we think that this kind of integration of RAM cannot be made work successful. FIgure 6.4 illustrates the use of RAM in the prototype exchange. From this figure it is clear that only the four control memories can be integrated into one memory. Each control memory generates its own read address for the data memory, and it is very likely that these addresses are all unequal. Therefore integration of the data memory is impossible.
Furthermore the integration of various RAM blocks introduces more wiring.

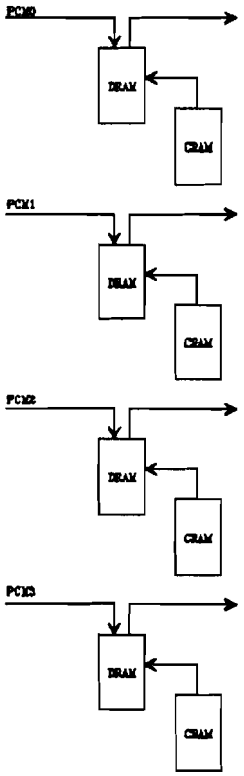


Figure 6.5 *Configuration of the RAM*

Further integration of the RAM needs further study.

7 Conclusions and recommendations

In this report we have seen how we can increase the capacity of the digital fault tolerant telephone exchange that is being developed at the Digital Systems Group.

We have studied and improved the design of a time switch element designed by [Timmer] and [van Lier], and we were able to quadruple its capacity to 4x64 PCM channels.
We have designed another time shift element, based upon a pipeline architecture. This time shift element is more suited for switching small numbers of datastreams at a high rate.
We have searched for various solutions in order to avoid the so-called wiring problem. Implementing a large space switch element in an IC requires a lot of wiring. Various architectures were studied in order to increase the capacity of the space switch element in a feasible way. We proposed a SSS network in order to implement a space switch element with more than 16 incoming lines.
Finally the chip area of all designs was roughly estimated. Because we have not done any lay-outing yet we can only make a rough guess of the size of the chip. These calculations have been done based upon the size of the RAM, which counts for the greater part of the total chip area, wiring not included.

The research that has been done was - sometime even severely - impeded by the use of the SDA hardware design tools.
We were only able to do bottom up schematic entry. For a long time it was impossible to make plots on paper. The SDA ES2 Ram Cell Generator, and the SDA ES2 Build In Self Test Ram Cell Generator still do not work correctly. Simulating a design with a large recursive structure runs the Appollo out of its 256 Kbytes stack. Furthermore there was an almost total lack of support from Pijnenburg, the software supplier and sales representative for CADENCE in Holland. The SDA system is so complex - it is much more complex than the Silvar Lisco System - that more support is absolutely necessary.

We have seen that at the Digital Systems Group they surely know very well how to design complex large digital systems, but when it comes to doing it themselves we see that - like in industrial environments - many problems occur.

We conclude that by increasing the capacity size of the memory elements becomes of great influence. We expect the size of the RAM and the wiring of the chip to count for a very large part of the total chip area.
Therefore we strongly recommend two further research activities.

We have seen that the memories (RAM blocks) of the designs count for a large part of the total chip area. We have also seen that large RAM blocks can store more bits per square unit than smaller ones. Therefore the configuration of the RAM needs further research.

Furthermore we think that when lay-outing and wiring of the prototype exchange has been done and the size of the chip inclusive wiring is known, it should be possible to determine the feasibility of the implementation of a large space switch element with full wiring.

References

Arnbak, J.C.,
DIGITALE TRANSMISSIESYSTEMEN, collegedictaat
Vakgroep telecommunicatie EC,
Afdeling der Elektrotechniek,
Technische Hogeschool Eindhoven.

Bakker and Klip
Course Notes 'Structured VLSI Design Course'
Digital Systems Group,
Department of Electrical Engineering,
Eindhoven University of Technology.

Clos, C.
A STUDY OF NON-BLOCKING SWITCHING NETWORKS
Bell System Technical Journal, Vol. 32, p. 406-424, March 1953

Hörbst, E., Müller Schloer C., Schwärtzel H.,
DESIGN OF VLSI CIRCUITS, BASED ON VENUS
Springer Verlag, Berlin.

Krol, Th.,
(N,K)-CONCEPT FAULT TOLERANCE,
IEEE Transactions on Computers, Vol. C-35, No.4, April 1986, p. 339-349.

Krol Th.,
AN INTRODUCTION TO FAULT TOLERANT COMPUTING
Symposium computer ondersteund ontwerpen in de elektrotechniek, KIvI, afdeling voor Elektrotechniek, 25 en 26 januari, TUE.

Lewin D.,
DESIGN OF LOGIC SYSTEMS
Van Nortrand Reinhold (UK), Co. Ltd.

Nikaido, Yamada, Fukada, Suzuki,
A 1.5 MBit/SEC 4 x 128- CHANNEL TIME SWITCH LSI FOR DIGITAL STILL PICTURES
1986 IEEE International Soild-State Circuits Conference, p. 124-125.

Ronayne, J.P.
INTRODUCTION TO DIGITAL COMMUNICATION SYSTEMS
New York: John Wiley & Sons

Sikkelman, J.G.W., de Jongh, A., Nordholt E.H., Stevens M.P.J.,
ASICs, INVOERINGSERVARINGEN

Appendix 1

List of Reports Telephony Group

Aggenbach, M.H.J.M.
ONTWERP VAN EEN (4,2)-DECODER/CODER VOOR EEN DIGITALE TELEFOONCENTRALE VOLGENS HET (4,2)-CONCEPT.
Master Thesis Report, Digital Systems Group, Department of Electrical Enigineering, Eindhoven University of Technology, 1988.

Bink, S.
IMPLEMENTATIE EN SIMULATIE VAN DE CONTROL_UNIT VAN EEN TST-SCHAKELNETWERK,
Projekt Report, Digital Systems Group, Department of Electrical Enigineering, Eindhoven University of Technology, 1990.

Budzelaar, F.P.M.
EEN IMPLEMENTATIE VAN EEN DIGITAAL SCHAKELKNOOPPUNT VOOR PCM LIJNEN MET 64 SPRAAKKANALEN.
Project Report, Digital Systems Group, Department of Electrical Enigineering, Eindhoven University of Technology, 1986.

Cornelisse, J.
ONTWERP EN IMPLEMENTATIE VAN EEN (4,2) DECODER VOOR EEN DIGITALE TELEFONIECENTRALE.
Master Thesis Report, Digital Systems Group, Department of Electrical Enigineering, Eindhoven University of Technology, 1990.

Dieleman, J.P.
ONTWERP VAN EEN NON-BLOCKING NETWERK VOOR EEN DIGITALE TELEFOONCENTRALE.
Master Thesis Report, Digital Systems Group, Department of Electrical Enigineering, Eindhoven University of Technology, 1987.

Engelen, W.J. van
ONTWERP VAN EEN CONCENTRATOR VOOR EEN DIGITALE TELEFOONCENTRALE VOLGENS HET (4,2)-CONCEPT.
Master Thesis Report, Digital Systems Group, Department of Electrical Enigineering, Eindhoven University of Technology, 1988.

Hense, M.L.G.E.
ONTWERP VAN EEN TST BUSTRANCIEVER VOOR EEN DIGITALE TELEFOONCENTRALE VOLGENS HET (4,2)-CONCEPT.
Master Thesis Report, Digital Systems Group, Department of Electrical Enigineering, Eindhoven University of Technology, 1988.

Hoorens, P.V.W.
ONTWERP VAN EEN TIME SWITCH ELEMENT VOOR EEN DIGITALE TELEFOONCENTRALE VOLGENS HET (4,2)-CONCEPT.
Master Thesis Report, Digital Systems Group, Department of Electrical Enigineering, Eindhoven University of Technology, 1988.

Jonge, C. de
ONTWERP VAN EEN SPACE SHIFT ELEMENT VOOR EEN DIGITALE TELEFOONCENTRALE VOLGENS HET (4,2)-CONCEPT.
Master Thesis Report, Digital Systems Group, Department of Electrical Enigineering, Eindhoven University of Technology, 1988.

Ligtenberg, M
SOFTWARE DESIGN FOR THE CONTROL SYSTEM OF A DIGITAL TELEPHONE EXCHANGE.
Master Thesis Report, Digital Systems Group, Department of Electrical Enigineering, Eindhoven University of Technology, 1990.

Lier, A. van
DESIGN OF A TIME/SPACE SWITCH ELEMENT FOR A DIGITAL TELEPHONE EXCHANGE.
Master Thesis Report, Digital Systems Group, Department of Electrical Enigineering, Eindhoven University of Technology, 1989.

Mil, H. van
ONTWERP VAN EEN SYNCHRONISATIE SYSTEEM VOOR EEN DIGITALE TELEFOONCENTRALE VOLGENS HET (4,2)-CONCEPT.
Master Thesis Report, Digital Systems Group, Department of Electrical Enigineering, Eindhoven University of Technology, 1988.

Reijalt, J.J.M. and R.F.J. de Vries
ONTWIKKELING VAN EEN TIME SHIFTER MET FOUT CORRECTIE VOOR EEN DIGITALE TELEFOONCENTRALE.
Master Thesis Report, Digital Systems Group, Department of Electrical Enigineering, Eindhoven University of Technology, 1987.

Timmer, A.H.
DESIGN OF THE CENTRAL SWITCH AND PCM SYNCHRONISATION OF AN EXCHANGE.
Master Thesis Report, Digital Systems Group, Department of Electrical Enigineering, Eindhoven University of Technology, 1990.

Verhoof, P.H.W.
REQUIREMENTS MODEL FOR THE CONTROL SYSTEM OF A DIGITAL TELEPHONE EXCHANGE.
Master Thesis Report, Digital Systems Group, Department of Electrical Enigineering, Eindhoven University of Technology, 1990.

Vervoort, J.H.A.
REALISATIE VAN EEN TIME SHIFTER VOOR EEN DIGITALE TELEFOONCENTRALE VOLGENS HET (4,2)-CONCEPT.
Master Thesis Report, Digital Systems Group, Department of Electrical Enigineering, Eindhoven University of Technology, 1988.

Witkam, L.F.
RELIABILITY IMPROVEMENT OF A TELEPHONE EXCHANGE CONTROL USING MULTI-VERSION HARDWARE AND SOFTWARE.
Master Thesis Report, Digital Systems Group, Department of Electrical Enigineering, Eindhoven University of Technology, 1989.

Appendix 2

Top Down Design Parallel Time Switch Element

Contents

2.1 Mux32test

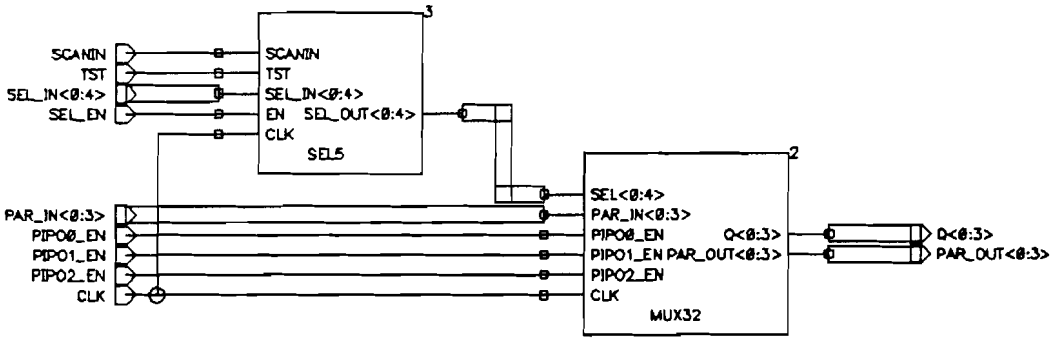


Figure 2.1 *Mux32test*

2.2 Mux32

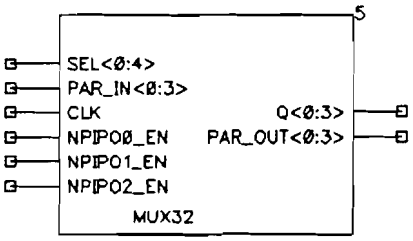


Figure 2.2 *Symbol Mux32*

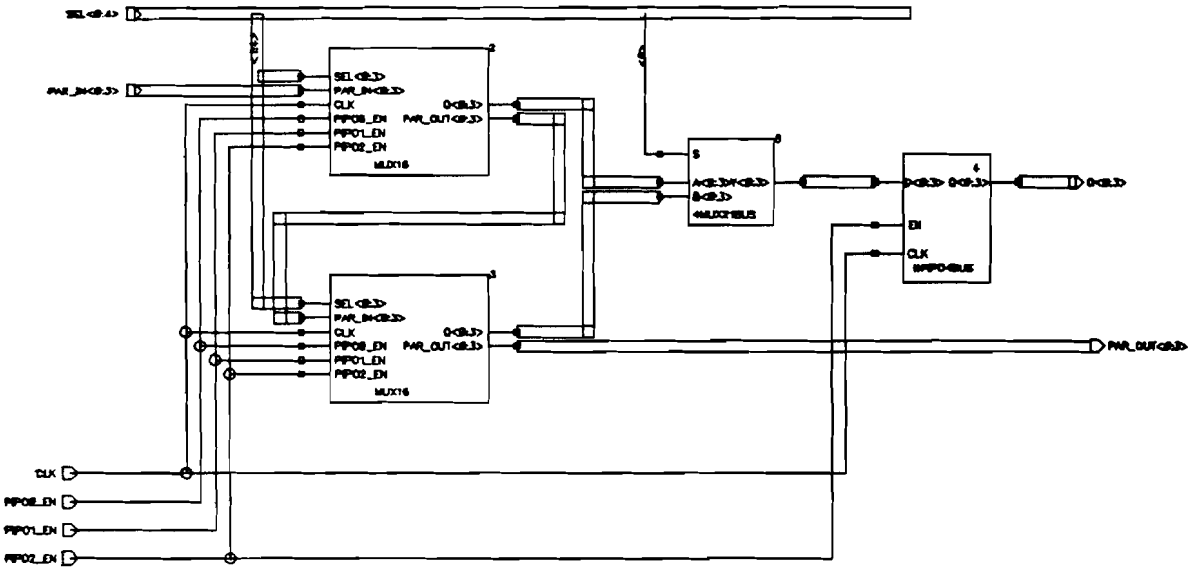


Figure 2.3 *Schematic Mux32*

2.3 Mux16

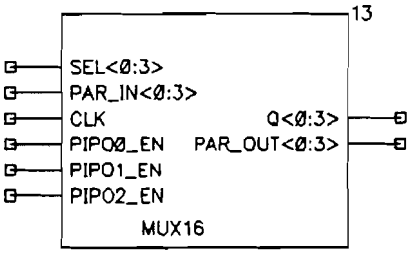


Figure 2.4 *Symbol Mux16*

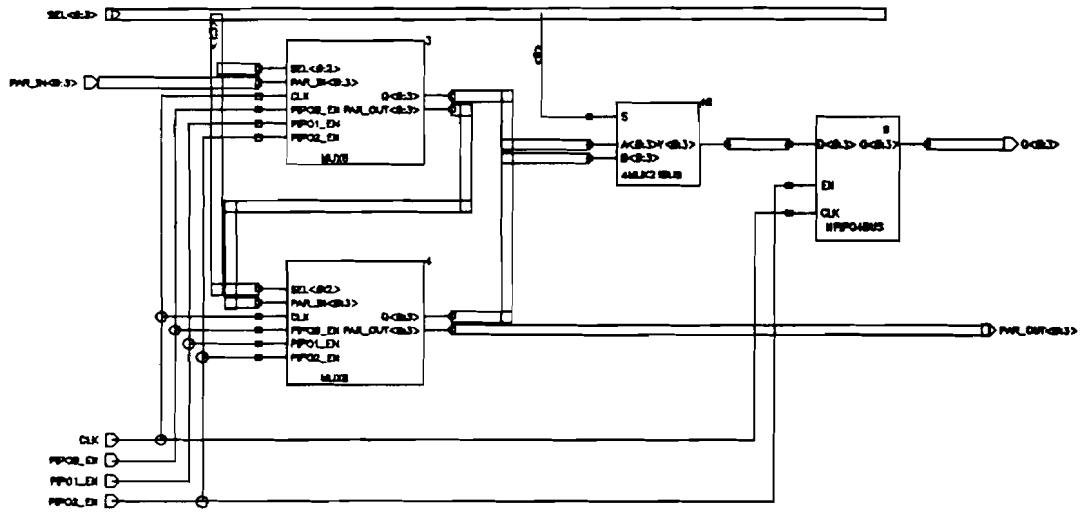


Figure 2.5 *Schematic Mux16*

2.4 Mux8

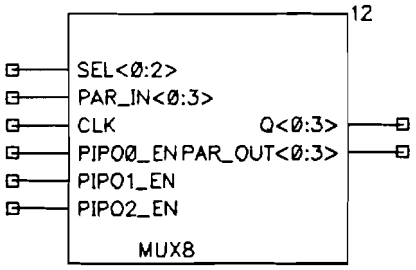


Figure 2.6 *Symbol Mux8*

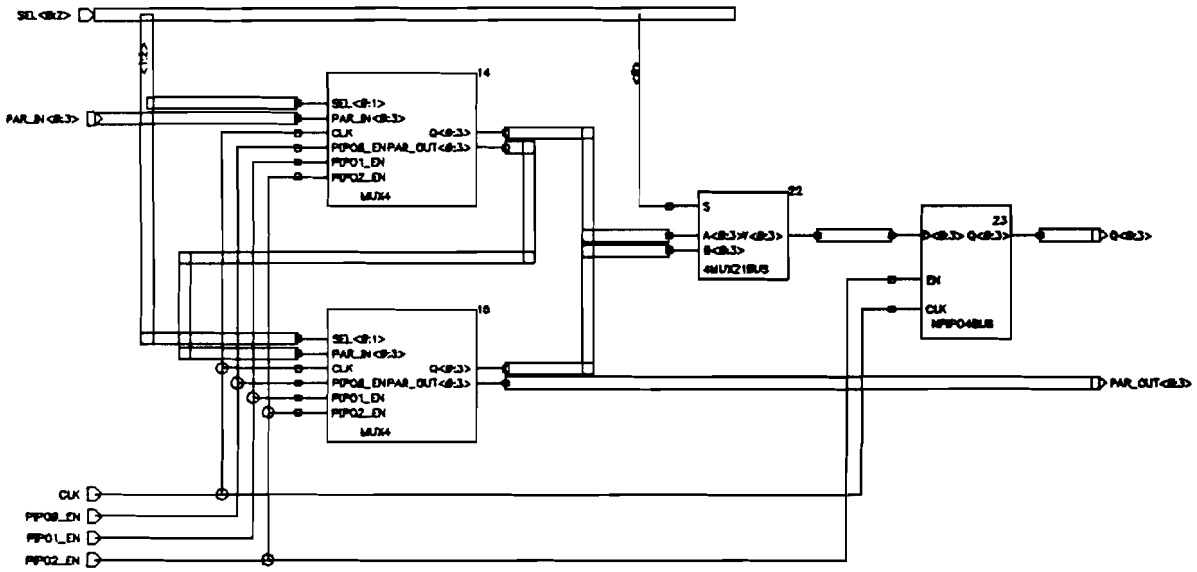


Figure 2.7 *Schematic Mux8*

2.5 Mux4

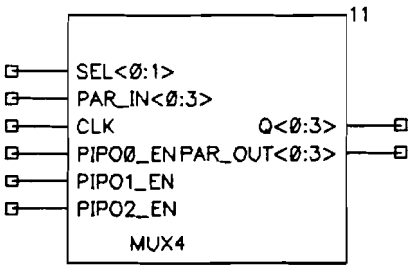


Figure 2.8 *Symbol Mux4*

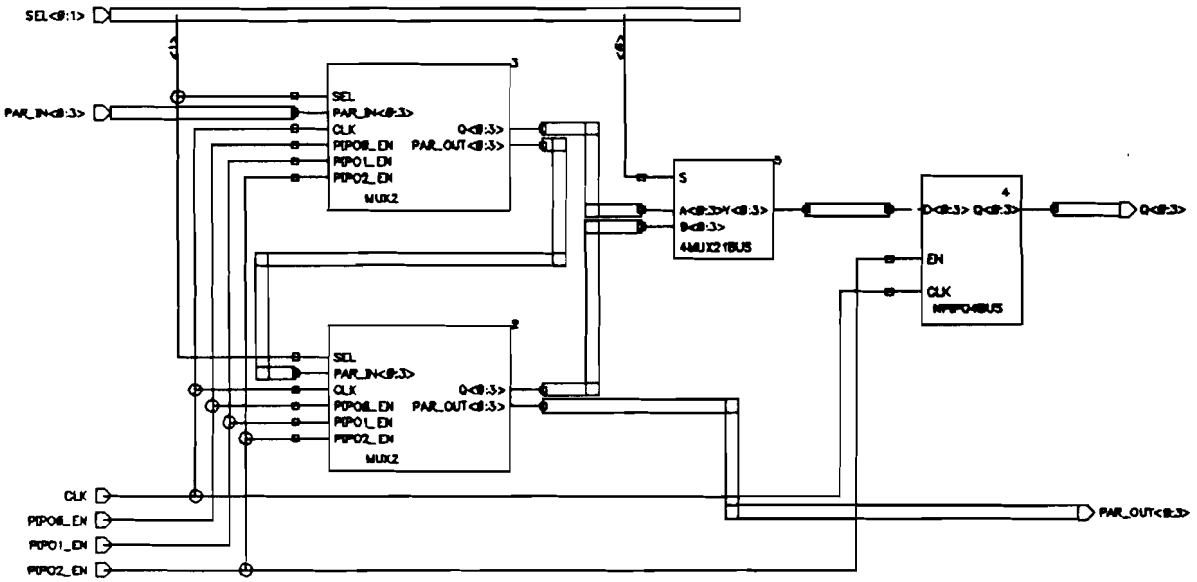


Figure 2.9 *Schematic Mux4*

2.6 Mux2

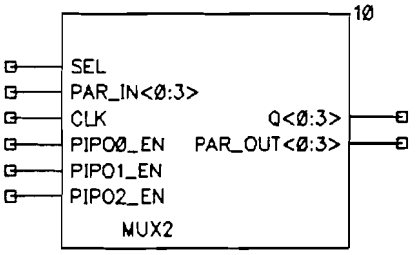


Figure 2.10 *Symbol Mux2*

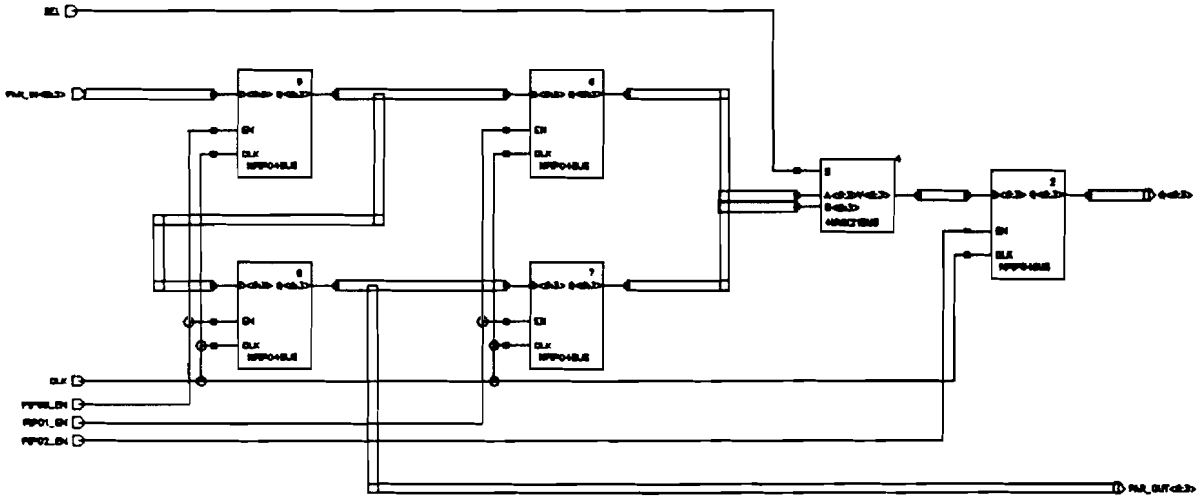


Figure 2.11 *Schematic Mux2*

2.7 Logic Elements

2.7.1 eNable Parallel In Parallel Out 4 bits register with bus (NPIPO4BUS)

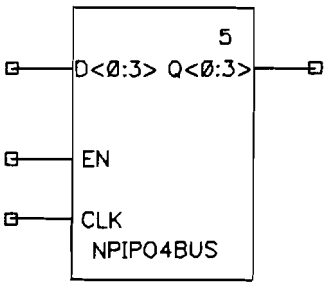


Figure 2.12 *Symbol eNable Parallel In Paralle Out 4 bits register with bus*

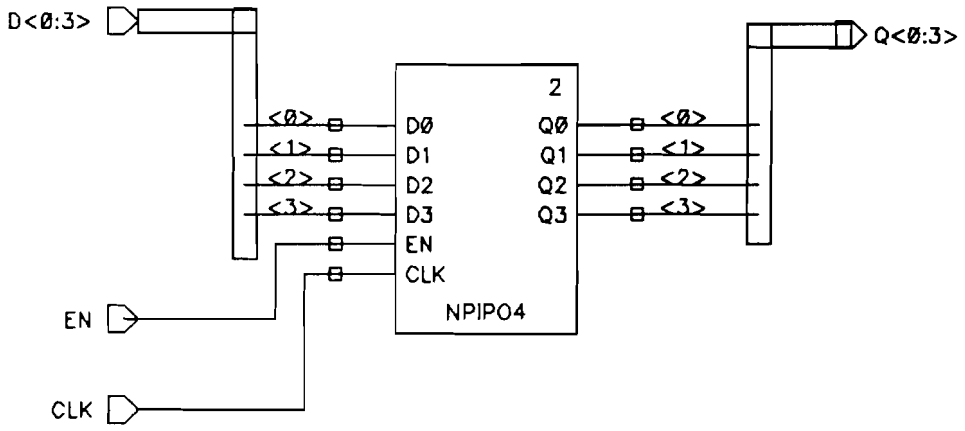


Figure 2.13 *Schematic eNable Parallel In Paralle Out 4 bits register with bus*

2.7.2 eNable Parallel In Parallel Out 4 bits register (NPIPO4)

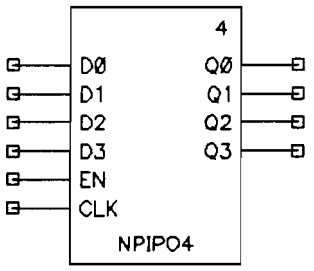


Figure 2.14 *Symbol eNable Parallel In Parallel Out 4 bits register (NPIPO4)*

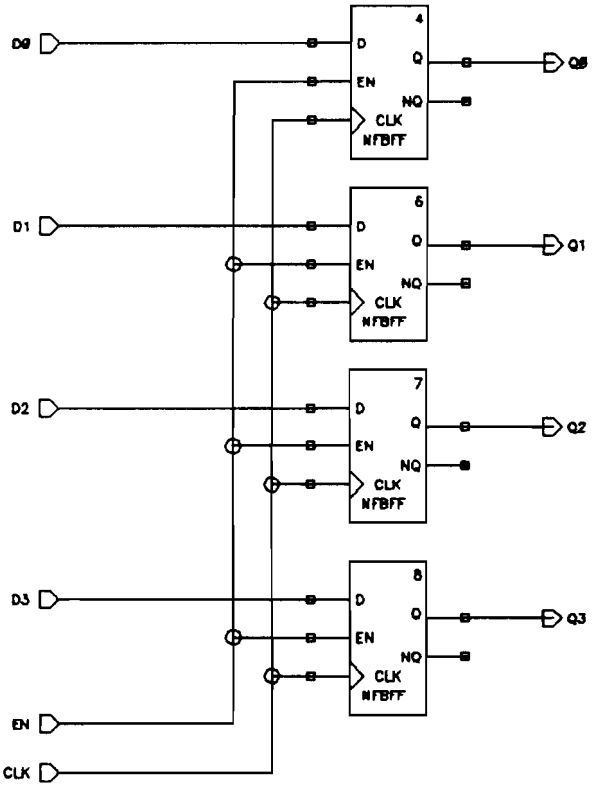


Figure 2.15 *Schematic eNable Parallel In Parallel Out 4 bits register (NPIPO4)*

2.7.3 Four 2-to-1 multiplexers with bus (4MUX21BUS)

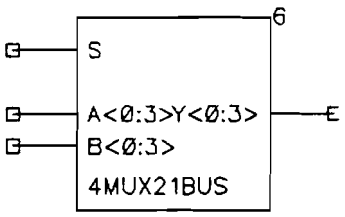


Figure 2.16 *Symbol Four 2-to-1 multiplexers (4MUX21BUS)*

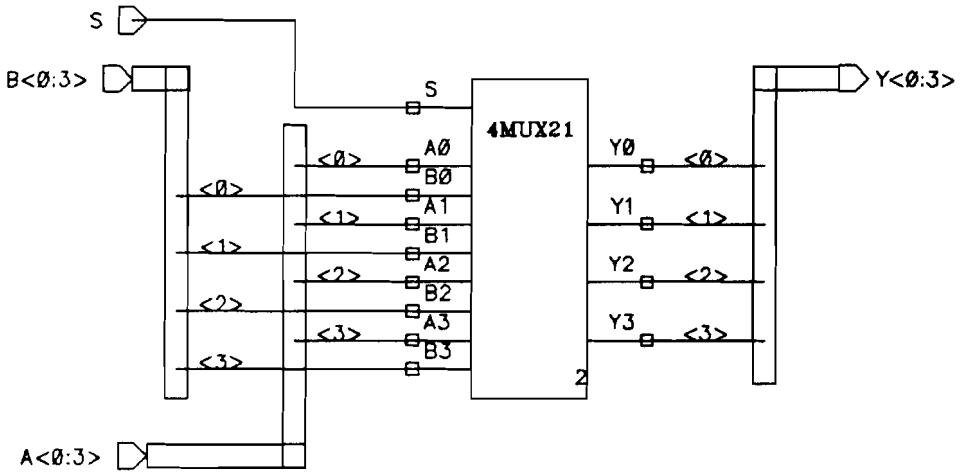


Figure 2.17 *Schematic Four 4-to-1 multiplexers (4MUX21BUS)*

2.7.4 eNable Feedback Flip Flop (NFBFF)

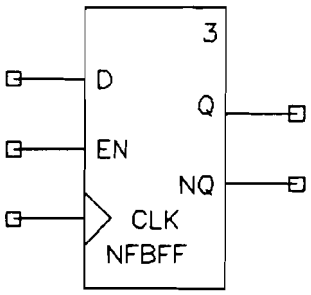


Figure 2.18 *Symbol eNable Feedback Flip Flop (NFBFF)*

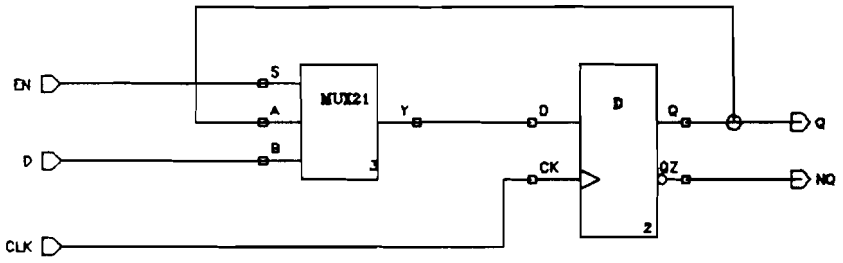


Figure 2.19 *Schematic eNable feedback Flip Flop*

Appendix 3

Simulation Problems Parallel Time Switch Element

Contents

3.1 Simulation Problem Mux32test

```
mulis : fans          Page 1                                    08/03/90  10:32 AM


H370: /3/4/15/4/2/7/3.Y
     ***Warning: fanin greater than 1
    fanin 654128        fanout 1
    load 0.10    (estIntCap 0.04)
    maxload 1.30

H509: /3/4/15/3/8/2/8/3.Y
    ***Warning: fanin greater than 1
    fanin 654272        fanout 1
    load 0.10    (estIntCap 0.04)
    maxload 1.30

H541: /3/4/15/3/7/2/6.HQ
    ***Warning: fanin greater than 1
    fanin 663536        fanout 0
    maxload 1.50

H542: /3/4/15/3/7/2/7.HQ
    ***Warning: fanin greater than 1
    fanin 665504        fanout 0
    maxload 1.50

H543: /3/4/15/3/7/2/8.HQ
    ***Warning: fanin greater than 1
    fanin 663520        fanout 0
    maxload 1.50
```

3.2 Simulation Problem Mux64test

```
muijs : si.log          Page 1                                    08/03/90  10:30 AM

si version 2.0d Fri Dec 16 21:55:13 PST 1988 (sda64)
si: Loading user defined simulation run control file "/station_a/user/stud/muijs/.simrc".
si: Loading simulation environment file "/station_a/user/stud/muijs/exch/time2/lib/mux64test/silosrun1/si.env".
si: Loading simulation capabilities file "//station_a/prog/sda/etc/si/simcap".
si: The default value of the variable "simControlFile" has been overriden.
si: The default value of the variable "simTimeUnit" has been overriden.
si: The default value of the variable "silosSimRepList" has been overriden.
si: The default value of the variable "silosSimStepList" has been overriden.
si: The default value of the variable "silosBinary" has been overriden.
Running simulation in directory: "/station_a/user/stud/muijs/exch/time2/lib/mux64test/silosrun1".
si: The default value of the variable "SIMOTHERINFO" has been overriden.


Assigning interconnect loading.
Memory fault


Interconnect assignment failed.


Simulation did not complete.
```

Appendix 4

SDA Service Reports

Contents

4.1 Memory Bug Report

```
muijs : sr.repl          Page 1                                  01/05/90  11:26 AM

                          SERVICE REPORT

Title :

Submitted by : J.N.M. Muijselaar
Company      : University of Technology Eindhoven
Telephone    : 040-473397

Hardware Platform : APOLLO DN4000
Unix Version      : BSD4.2 release 9.5
Software Version  : 2.0

Please check (X) one in each list --

  Nature :          Type :           Severity :        Reproducibility :
   _ Software        _ Question       X Work stopped    X Yes
   _ Hardware        X Incident/Bug   _ Work slowed     _ No
   _ Documentation   _ Changes/Req.   _ Inconvenienced  _ Intermittently
   X Unknown         _ Miscellaneous  _ Cosmetic        _ Not applicable

Detail Description:

Simulating a design gives the following results in file si.out.


**********************************************************************************************
si version 2.0d Fri Dec 16 21:55:13 PST 1988 (sda64)
si: Loading user defined simulation run control file "/station_a/user/stud/muijs/.simrc".
si: Loading simulation environment file "/station_a/user/stud/muijs/exch/time2/lib/mux64test/silosrun1/si.env".
si: Loading simulation capabilities file "//station_a/prog/sda/etc/si/simcap".
si: The default value of the variable "simControlFile" has been overriden.
si: The default value of the variable "simTimeUnit" has been overriden.
si: The default value of the variable "silosSimRepList" has been overriden.
si: The default value of the variable "silosSimStopList" has been overriden.
si: The default value of the variable "silosBinary" has been overriden.
Running simulation in directory: "/station_a/user/stud/muijs/exch/time2/lib/mux64test/silosrun1".
si: The default value of the variable "SIMOTHERINFO" has been overriden.


Assigning interconnect loading.
Memory fault   <------------------------------------------------------ !!!!!????


Interconnect assignment failed.


Simulation did not complete.


**********************************************************************************************


The design has a recursive structure, i.e. mux64test consist of 2 mux32, which consist
of 2 mux16 ,...., and so on.
A mux2 consists of a couple of flip flops, multiplexers and gates.
However the size of the netlist is only 424520 bytes.
```

4.2 Plot Bug Report

muijs : sr.rep2 Page 1 01/05/90 11:26 AM

SERVICE REPORT

Title :

Submitted by : J.N.M. Muijselaar
Company : University of Technology Eindhoven
Telephone : 040-473397

Hardware Platform : APOLLO DN4000
Unix Version : BSD4.2 release 9.5
Software Version : 2.0

Please check (X) one in each list --

Nature :	Type :	Severity :	Reproducibility :
_ Software	_ Question	X Work stopped	X Yes
_ Hardware	X Incident/Bug	_ Work slowed	_ No
_ Documentation	_ Changes/Req.	_ Inconvenienced	_ Intermittently
X Unknown	_ Miscellaneous	_ Cosmetic	_ Not applicable

Detail Description:

No plots on paper can be made using our system.

4.3 Log in Bug Report

```
muijs : sr.rep1          Page 1                                  01/05/90  11:26 AM

                          SERVICE REPORT

Title :

Submitted by : J.H.M. Muijselaar
Company      : University of Technology Eindhoven
Telephone    : 040-473397

Hardware Platform : APOLLO DN4000
Unix Version      : BSD4.2 release 9.5
Software Version  : 2.0

Please check (X) one in each list --

  Nature :          Type :            Severity :         Reproducibility :
   X Software        _ Question        X Work stopped     _ Yes
   _ Hardware        X Incident/Bug    _ Work slowed      _ No
   _ Documentation   _ Changes/Req.    _ Inconvenienced   X Intermittently
   _ Unknown         _ Miscellaneous   _ Cosmetic         _ Not applicable

Detail Description:


After installation of the ECPD15 library severe problems occured when logging into
the system.
Intermittently the system crashes after start up and 'running solosetup()'.
```

4.4 Oct-Hex Bug Report

```
muijs : sr.rep5          Page 1                                  02/07/90  2:20 PM

                          SERVICE REPORT


Title :

Submitted by : J.H.H. Huijselaar
Company      : University of Technology Eindhoven
Telephone    : 040-473397

Hardware Platform : APOLLO DN4000
Unix Version      : BSD4.2 release 9.5
Software Version  : 2.0

Please check (X) one in each list --

  Nature :          Type :            Severity :        Reproducibility :
   X Software        _ Question        X Work stopped    X Yes
   _ Hardware        X Incident/Bug    _ Work slowed     _ No
   _ Documentation   _ Changes/Req.    _ Inconvenienced  _ Intermittently
   _ Unknown         _ Miscellaneous   _ Cosmetic        _ Not applicable

Detail Description


Simulating a design gives the following results in si.log:




si.log
.....

Simulating a design
si version 2.04 Fri Dec 16 21:55:13 PST 1988 (sda64)
si: Loading user defined simulation run control file "/station_a/user/stud/muijs/.simrc".
si: Loading simulation environment file "/station_a/user/stud/muijs/exch/time4/lib/mux8/silosrun2/si.env".
si: Loading simulation capabilities file "//station_a/prog/sda/etc/si/simcap".
si: The default value of the variable "simControlFile" has been overriden.
si: The default value of the variable "simTimeUnit" has been overriden.
si: The default value of the variable "silosSimRepList" has been overriden.
si: The default value of the variable "silosSimStopList" has been overriden.
si: The default value of the variable "silosBinary" has been overriden.
Running simulation in directory: "/station_a/user/stud/muijs/exch/time4/lib/mux8/silosrun2".
si: The default value of the variable "SIMOTHERINFO" has been overriden.


Assigning interconnect loading.


Interconnect assignment done.

Running simin

Running runsim with simulator: "silos"
Begin simulation:    Wed Feb  7 13:08:47 MET 1990
si: Silos did not complete without errors.
    Check si.out or SAVE.ERR file in the simulation run directory
    for silos error messages.
End simulation:      Wed Feb  7 13:09:12 MET 1990


Simulation did not complete.
.....


with batch.out :
.....

    ERROR:  BATCH filename "simout.tmp"
            could not be assigned
```

```
            Default filename "batch.out"
            will be used instead


 Ready:  INPUT  NETLIST

    Reading "netlist"
    File "NETLIST" input

 Ready:  INPUT SI.LOADS

    Reading "si.loads"
    File "SI.LOADS" input

 Ready:  INPUT  .TERM
    Enter .TERM Data
```

This is most probably caused by an error of SI generating file si.inp.

The design has terminal pin SEL<2:0>, which is mapped onto an .OCT NSEL2-0 statement in the si.inp file.
Replacing this pin by SEL<3:0> and connecting the introduced net SEL<3> to
a bus retention (i.e. HZPULL) gives a .HEX NSEL3-0 statement in the si.inp file.
Simulating this design terminates correctly.

The only difference between the two si.inp files seems to be the .OCT and .HEX statement.
Apparently SILOS doesn't accept an .OCT statement.

wrong si.inp file

```
.lib .mac=//station_a/prog/ecpd15/utl_cust/si/macrolib
batch simout.tmp
input  netlist
input si.loads
input  .term
.udecay .default=50
.symbol *D=Z

N36  .CLK   0 S1  2000 S0  4500 S1  5000 S1  .REP  0
.OCT  NSEL2-0  N4,   N3,   N2
.HEX  NPAR_IN3-0  N32,   N31,   N30,   N29
.HEX  NQ3-0  N28,   N27,   N26,   N25
.HEX  NPAR_OUT3-0  N24,   N23,   N22,   N21
.PATTERN  NSEL2-0  NPAR_IN3-0  N35 N34
+         N33
0     0 7 1 0 0
5000     0 6 1 0 0
10000     0 5 1 0 0
15000     0 4 1 0 0
20000     0 3 1 0 0
25000     0 2 1 0 0
30000     0 1 1 0 0
35000     0 0 1 0 0
40000     0 F 0 1 0
45000     0 F 0 0 1
50000     4 F 0 0 1
55000     0 F 0 0 1
60000     6 F 0 0 1
65000     2 F 0 0 1
70000     4 F 0 0 1
75000     1 F 0 0 1
80000     7 F 0 0 1
85000     3 F 0 0 1
90000     1 F 0 0 1
.EOP
.TABLE    NQ3-0  NPAR_OUT3-0
.TNODE    N28 N27 N26 N25 N24
+         N23 N22 N21

Itype errors
ISIMUL  0  TO  95000
```

```
!type sizes
!type errors
!exit
```

correct si.inp file

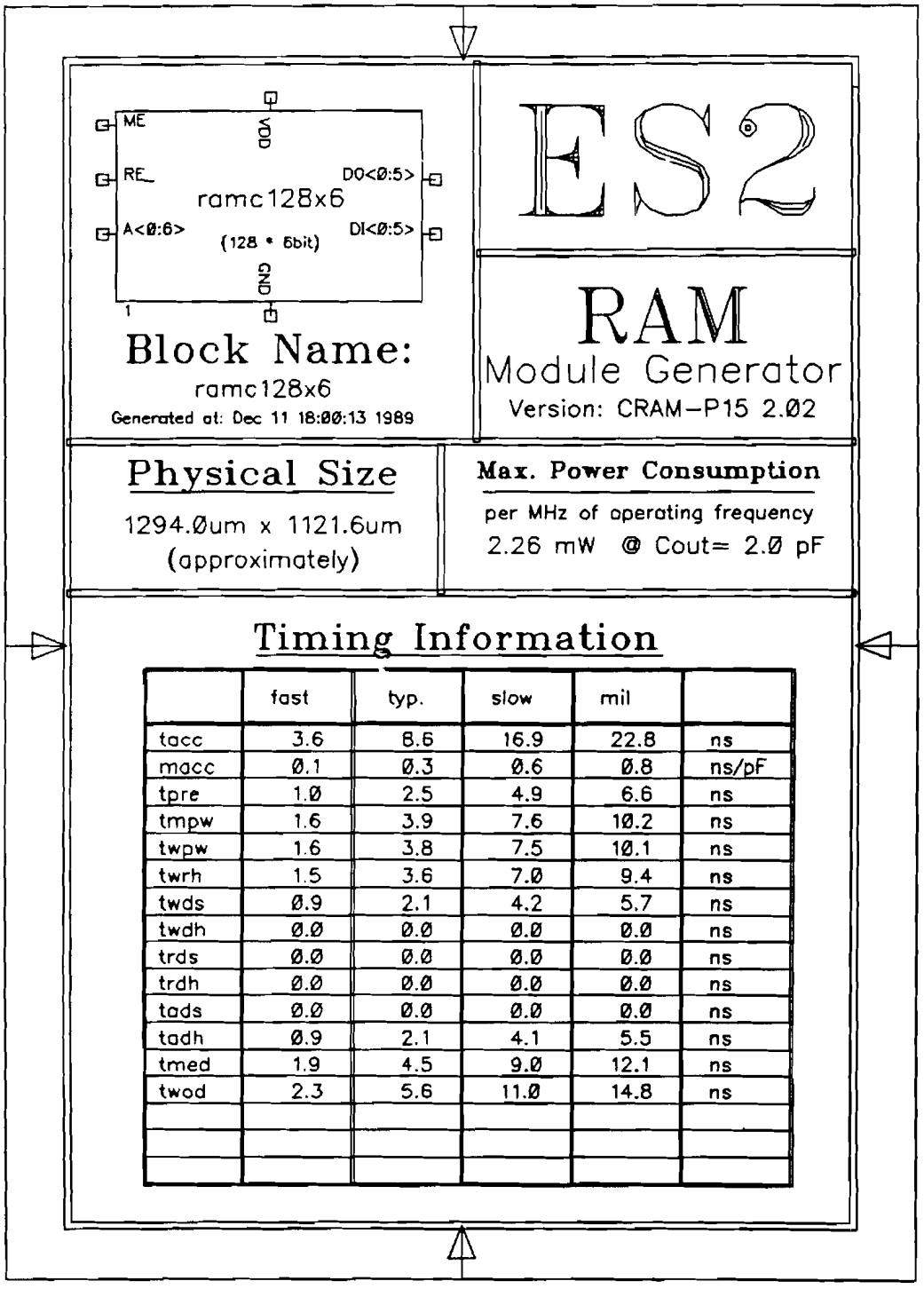
```
.lib .mac=//station_a/prog/ecpd15/utl_cust/si/macrolib
batch simout.tmp
input  netlist
input si.loads
input  .term
.ndecay .default=50
.symbol %D=Z

N10  .CLK   0 S1  2000 S0  4500 S1  5000 S1  .REP  0
.HEX  NSEL3-0  N5,   N4,   N3,   N2
.HEX  NPAR_IN3-0  N9,   N8,   N7,   N6
.HEX  NQ3-0  N17,   N16,   N15,   N14
.HEX  NPAR_OUT3-0  N21,   N20,   N19,   N18
.PATTERN  NSEL3-0  NPAR_IN3-0  N11 N12
+         N13
0    0 7 1 0 0
5000     0 6 1 0 0
10000    0 5 1 0 0
15000    0 4 1 0 0
20000    0 3 1 0 0
25000    0 2 1 0 0
30000    0 1 1 0 0
35000    0 0 1 0 0
40000    0 F 0 1 0
45000    0 F 0 0 1
50000    4 F 0 0 1
55000    0 F 0 0 1
60000    6 F 0 0 1
65000    2 F 0 0 1
70000    4 F 0 0 1
75000    1 F 0 0 1
80000    7 F 0 0 1
85000    3 F 0 0 1
90000    1 F 0 0 1
.EOP
.TABLE    NQ3-0  NPAR_OUT3-0
.THODE    N17 N16 N15 N14 N21
+         N20 N19 N18

!type errors
!SIMUL  0  TO  95000

!type sizes
!type errors
!exit
```

Appendix 5

Datasheet RAM

Contents

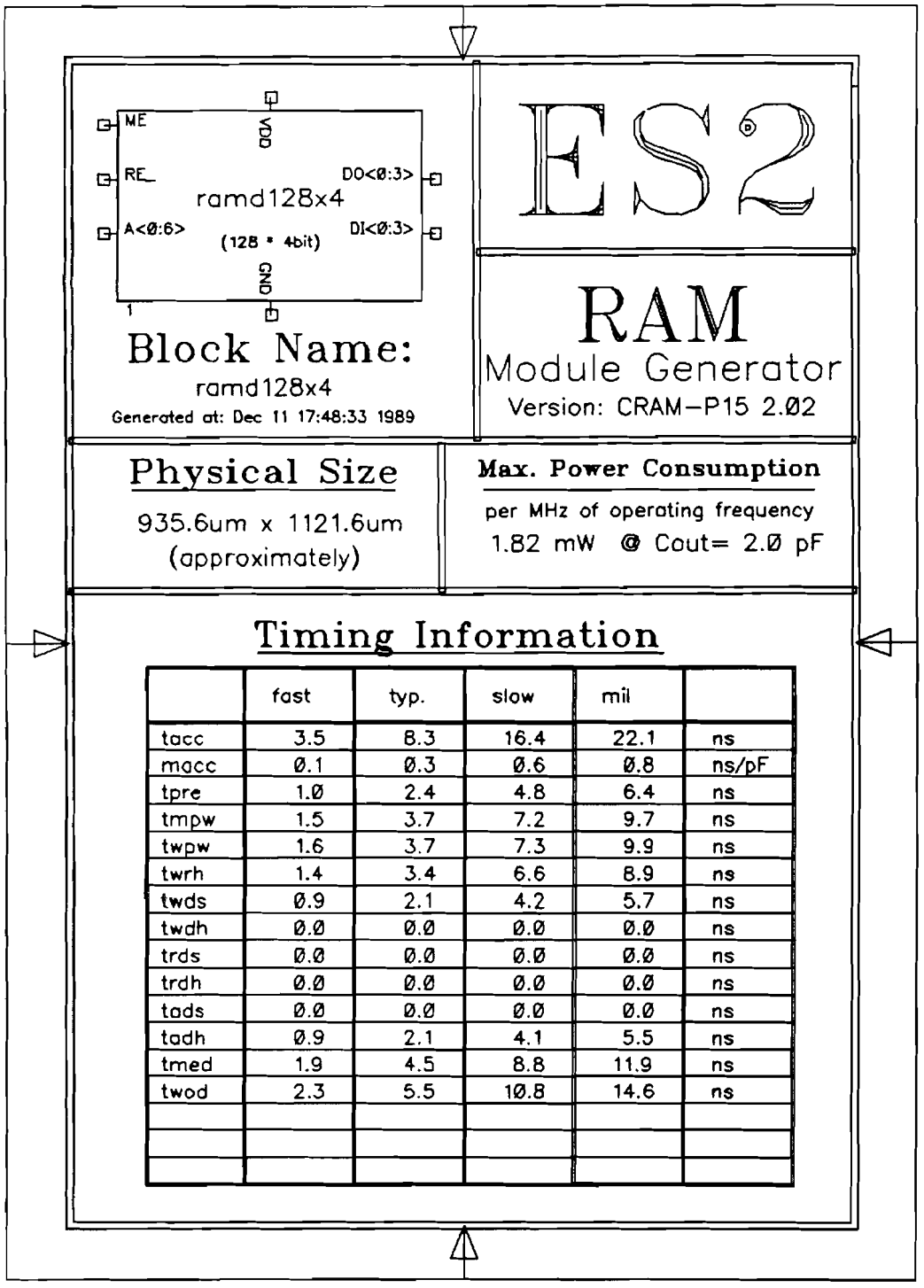
5.1 RAM 128x6



	fast	typ.	slow	mil	
tacc	3.6	8.6	16.9	22.8	ns
macc	0.1	0.3	0.6	0.8	ns/pF
tpre	1.0	2.5	4.9	6.6	ns
tmpw	1.6	3.9	7.6	10.2	ns
twpw	1.6	3.8	7.5	10.1	ns
twrh	1.5	3.6	7.0	9.4	ns
twds	0.9	2.1	4.2	5.7	ns
twdh	0.0	0.0	0.0	0.0	ns
trds	0.0	0.0	0.0	0.0	ns
trdh	0.0	0.0	0.0	0.0	ns
tads	0.0	0.0	0.0	0.0	ns
tadh	0.9	2.1	4.1	5.5	ns
tmed	1.9	4.5	9.0	12.1	ns
twod	2.3	5.6	11.0	14.8	ns

Figure 5.1 *RAM 128X6*

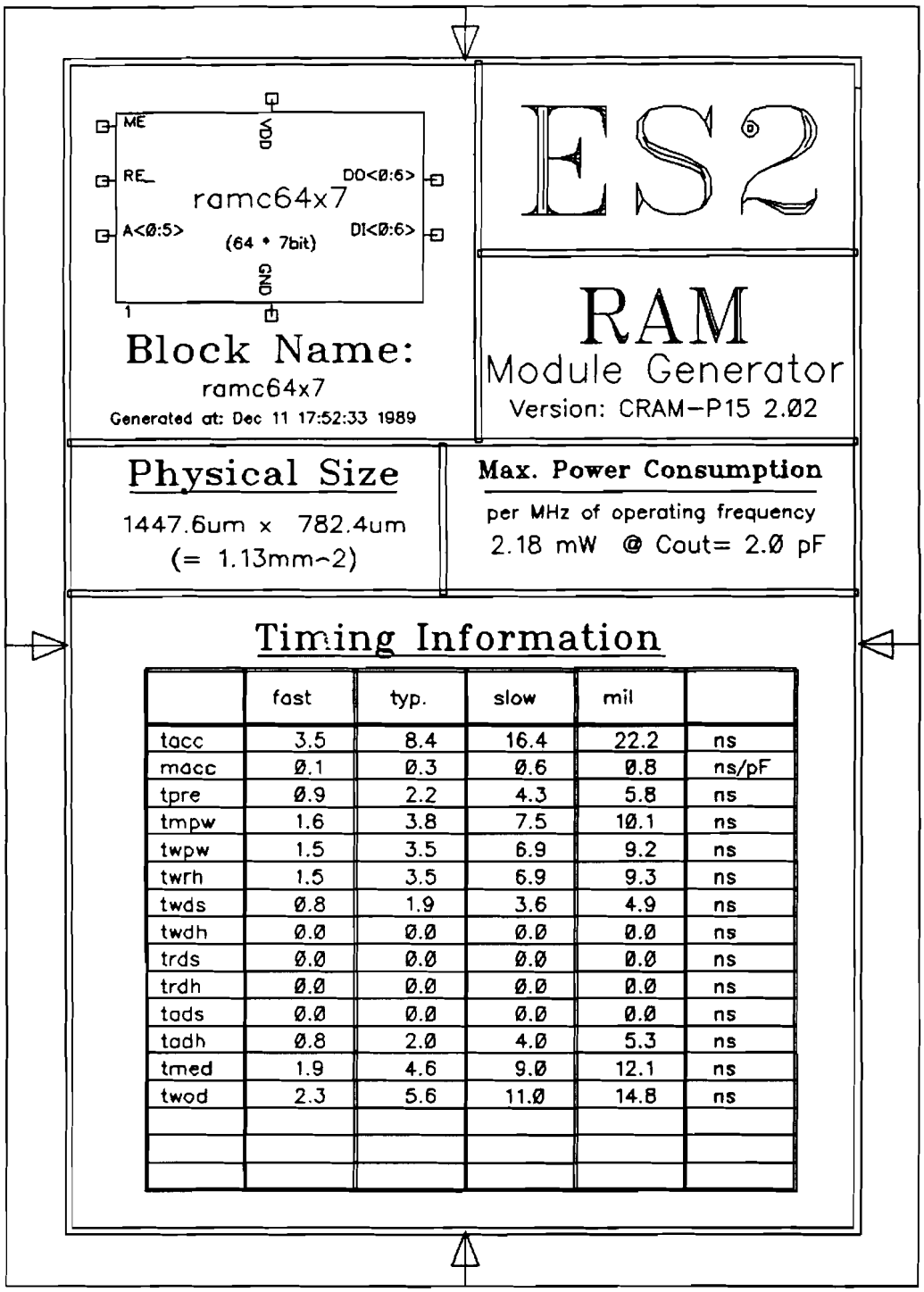
5.2 RAM 128x4



	fast	typ.	slow	mil	
tacc	3.5	8.3	16.4	22.1	ns
macc	0.1	0.3	0.6	0.8	ns/pF
tpre	1.0	2.4	4.8	6.4	ns
tmpw	1.5	3.7	7.2	9.7	ns
twpw	1.6	3.7	7.3	9.9	ns
twrh	1.4	3.4	6.6	8.9	ns
twds	0.9	2.1	4.2	5.7	ns
twdh	0.0	0.0	0.0	0.0	ns
trds	0.0	0.0	0.0	0.0	ns
trdh	0.0	0.0	0.0	0.0	ns
tads	0.0	0.0	0.0	0.0	ns
tadh	0.9	2.1	4.1	5.5	ns
tmed	1.9	4.5	8.8	11.9	ns
twod	2.3	5.5	10.8	14.6	ns

Figure 5.2 *RAM 128x4*

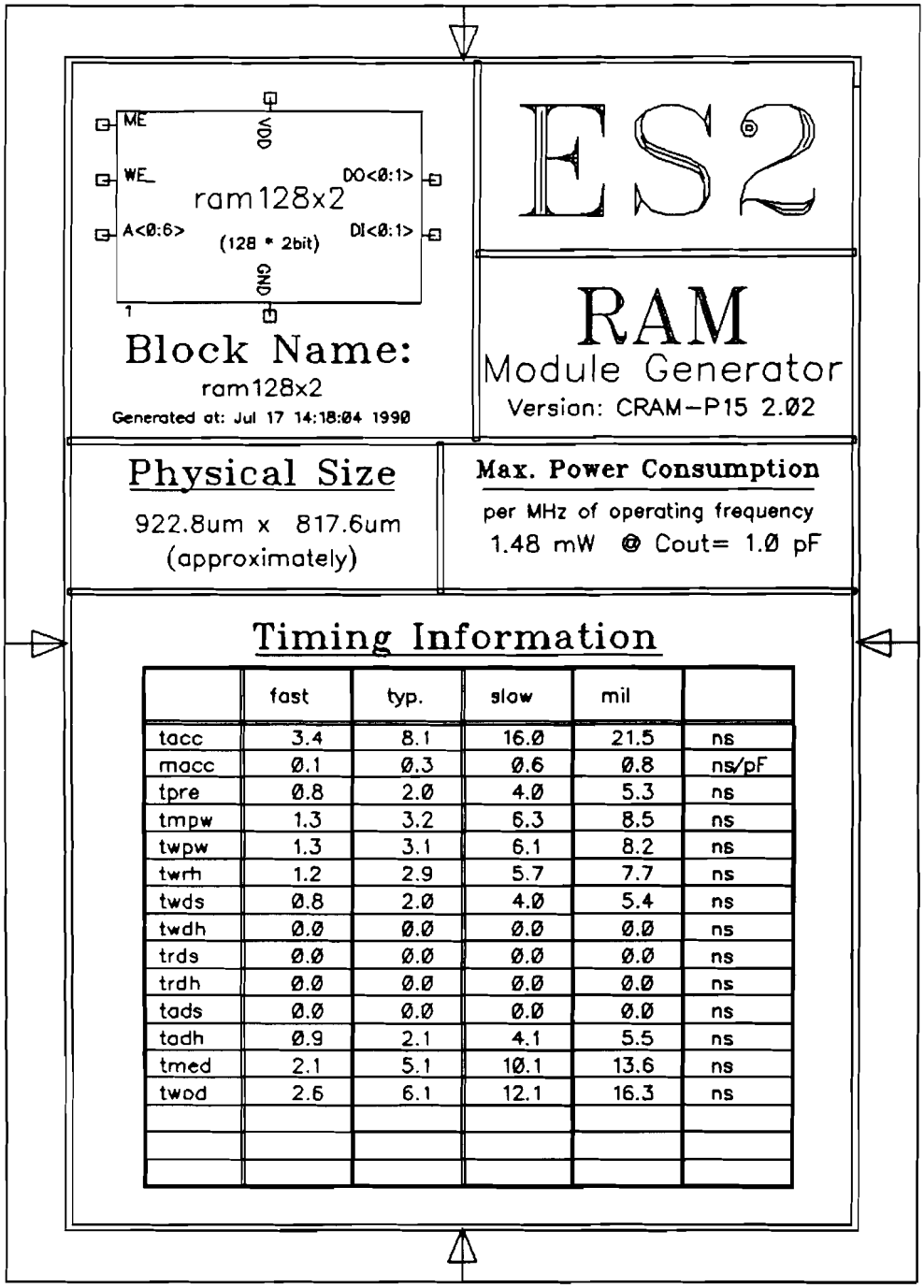
5.3 RAM 64x7



	fast	typ.	slow	mil	
tacc	3.5	8.4	16.4	22.2	ns
macc	Ø.1	Ø.3	Ø.6	Ø.8	ns/pF
tpre	Ø.9	2.2	4.3	5.8	ns
tmpw	1.6	3.8	7.5	10.1	ns
twpw	1.5	3.5	6.9	9.2	ns
twrh	1.5	3.5	6.9	9.3	ns
twds	Ø.8	1.9	3.6	4.9	ns
twdh	Ø.Ø	Ø.Ø	Ø.Ø	Ø.Ø	ns
trds	Ø.Ø	Ø.Ø	Ø.Ø	Ø.Ø	ns
trdh	Ø.Ø	Ø.Ø	Ø.Ø	Ø.Ø	ns
tads	Ø.Ø	Ø.Ø	Ø.Ø	Ø.Ø	ns
tadh	Ø.8	2.Ø	4.Ø	5.3	ns
tmed	1.9	4.6	9.Ø	12.1	ns
twod	2.3	5.6	11.Ø	14.8	ns

Figure 5.3 *RAM 64x7*

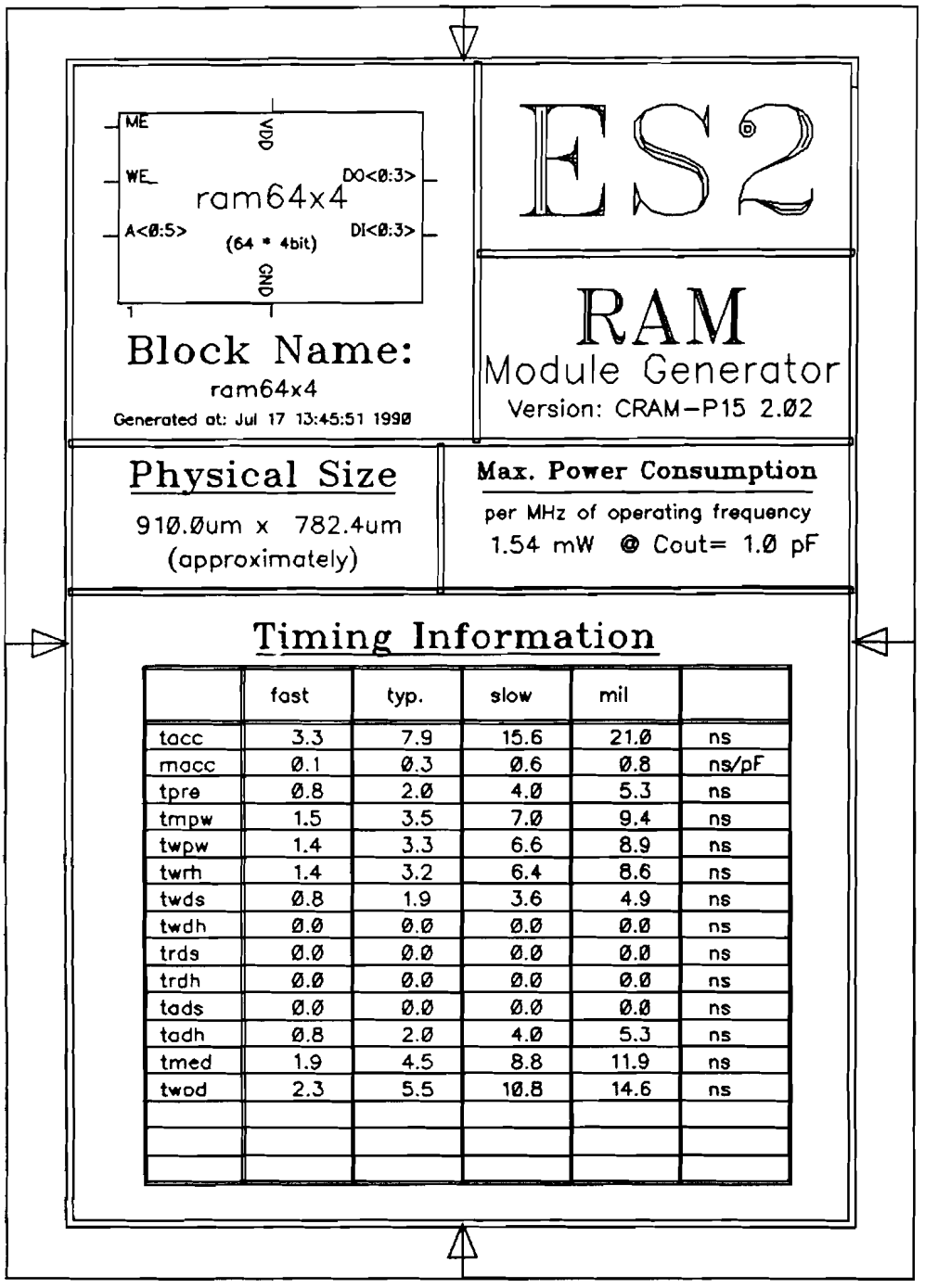
5.4 RAM 128x2



	fast	typ.	slow	mil	
tacc	3.4	8.1	16.0	21.5	ns
macc	0.1	0.3	0.6	0.8	ns/pF
tpre	0.8	2.0	4.0	5.3	ns
tmpw	1.3	3.2	6.3	8.5	ns
twpw	1.3	3.1	6.1	8.2	ns
twrh	1.2	2.9	5.7	7.7	ns
twds	0.8	2.0	4.0	5.4	ns
twdh	0.0	0.0	0.0	0.0	ns
trds	0.0	0.0	0.0	0.0	ns
trdh	0.0	0.0	0.0	0.0	ns
tads	0.0	0.0	0.0	0.0	ns
tadh	0.9	2.1	4.1	5.5	ns
tmed	2.1	5.1	10.1	13.6	ns
twod	2.6	6.1	12.1	16.3	ns

Figure 5.4 *RAM 128x2*

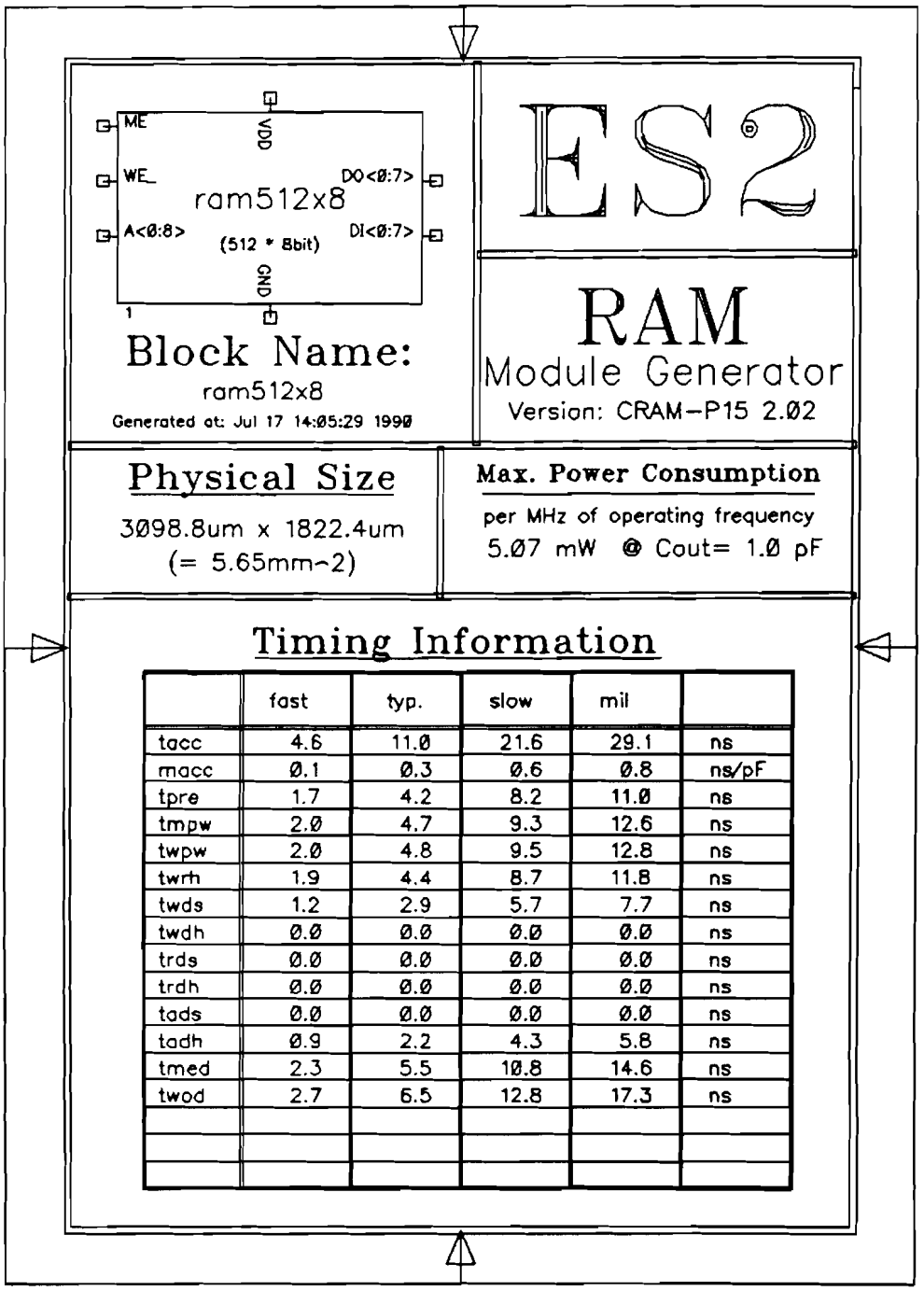
5.5 RAM 64x4



	fast	typ.	slow	mil	
tacc	3.3	7.9	15.6	21.0	ns
macc	0.1	0.3	0.6	0.8	ns/pF
tpre	0.8	2.0	4.0	5.3	ns
tmpw	1.5	3.5	7.0	9.4	ns
twpw	1.4	3.3	6.6	8.9	ns
twrh	1.4	3.2	6.4	8.6	ns
twds	0.8	1.9	3.6	4.9	ns
twdh	0.0	0.0	0.0	0.0	ns
trds	0.0	0.0	0.0	0.0	ns
trdh	0.0	0.0	0.0	0.0	ns
tads	0.0	0.0	0.0	0.0	ns
tadh	0.8	2.0	4.0	5.3	ns
tmed	1.9	4.5	8.8	11.9	ns
twod	2.3	5.5	10.8	14.6	ns

Figure 5.5 *RAM 64x4*

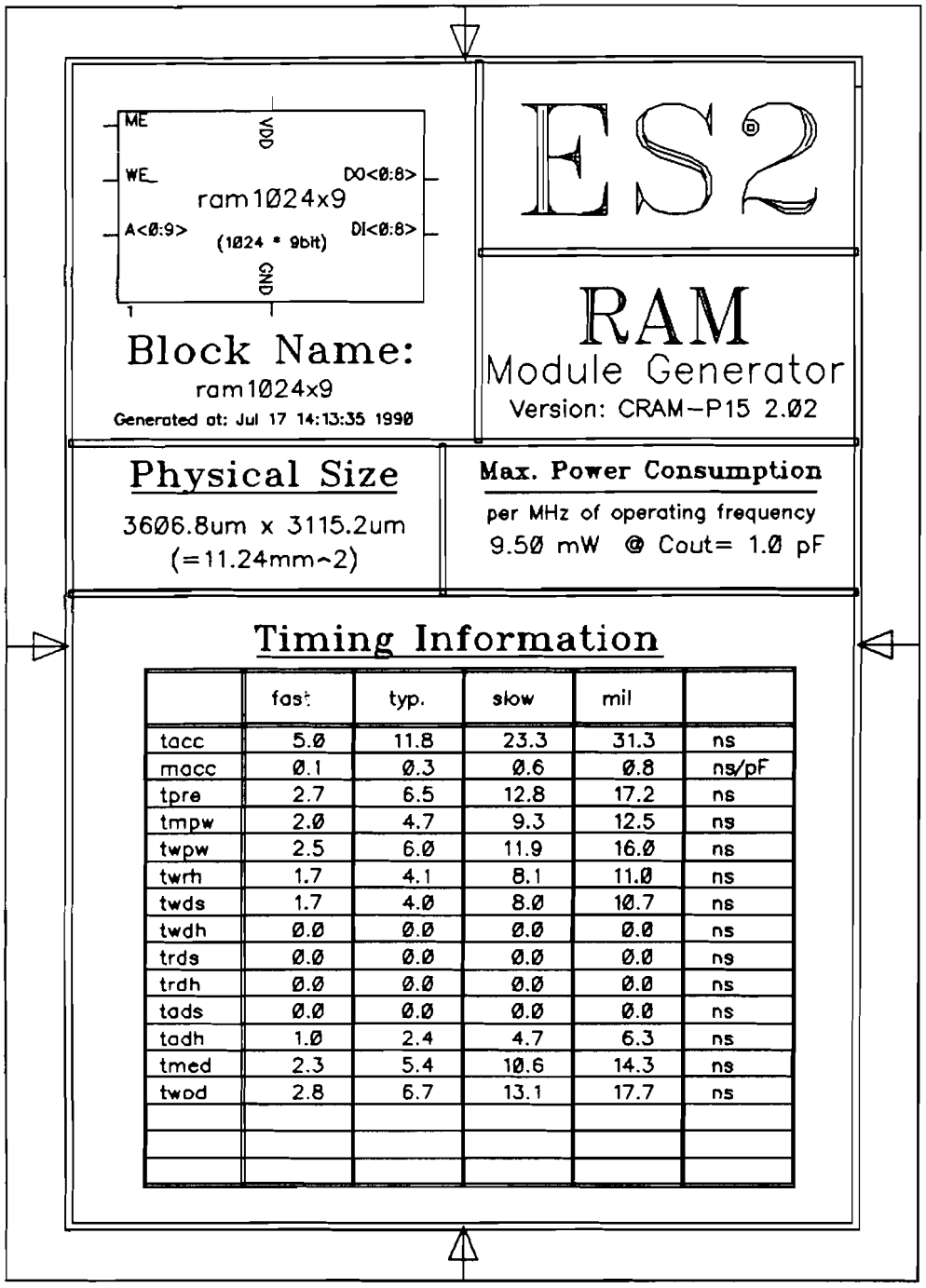
5.6 RAM 512x8



	fast	typ.	slow	mil	
tacc	4.6	11.0	21.6	29.1	ns
macc	0.1	0.3	0.6	0.8	ns/pF
tpre	1.7	4.2	8.2	11.0	ns
tmpw	2.0	4.7	9.3	12.6	ns
twpw	2.0	4.8	9.5	12.8	ns
twrh	1.9	4.4	8.7	11.8	ns
twds	1.2	2.9	5.7	7.7	ns
twdh	0.0	0.0	0.0	0.0	ns
trds	0.0	0.0	0.0	0.0	ns
trdh	0.0	0.0	0.0	0.0	ns
tads	0.0	0.0	0.0	0.0	ns
tadh	0.9	2.2	4.3	5.8	ns
tmed	2.3	5.5	10.8	14.6	ns
twod	2.7	6.5	12.8	17.3	ns

Figure 5.6 *RAM 512x8*

5.7 RAM 1024x9



	fast	typ.	slow	mil	
tacc	5.Ø	11.8	23.3	31.3	ns
macc	Ø.1	Ø.3	Ø.6	Ø.8	ns/pF
tpre	2.7	6.5	12.8	17.2	ns
tmpw	2.Ø	4.7	9.3	12.5	ns
twpw	2.5	6.Ø	11.9	16.Ø	ns
twrh	1.7	4.1	8.1	11.Ø	ns
twds	1.7	4.Ø	8.Ø	1Ø.7	ns
twdh	Ø.Ø	Ø.Ø	Ø.Ø	Ø.Ø	ns
trds	Ø.Ø	Ø.Ø	Ø.Ø	Ø.Ø	ns
trdh	Ø.Ø	Ø.Ø	Ø.Ø	Ø.Ø	ns
tads	Ø.Ø	Ø.Ø	Ø.Ø	Ø.Ø	ns
tadh	1.Ø	2.4	4.7	6.3	ns
tmed	2.3	5.4	1Ø.6	14.3	ns
twod	2.8	6.7	13.1	17.7	ns

Figure 5.7 *RAM 1024x9*

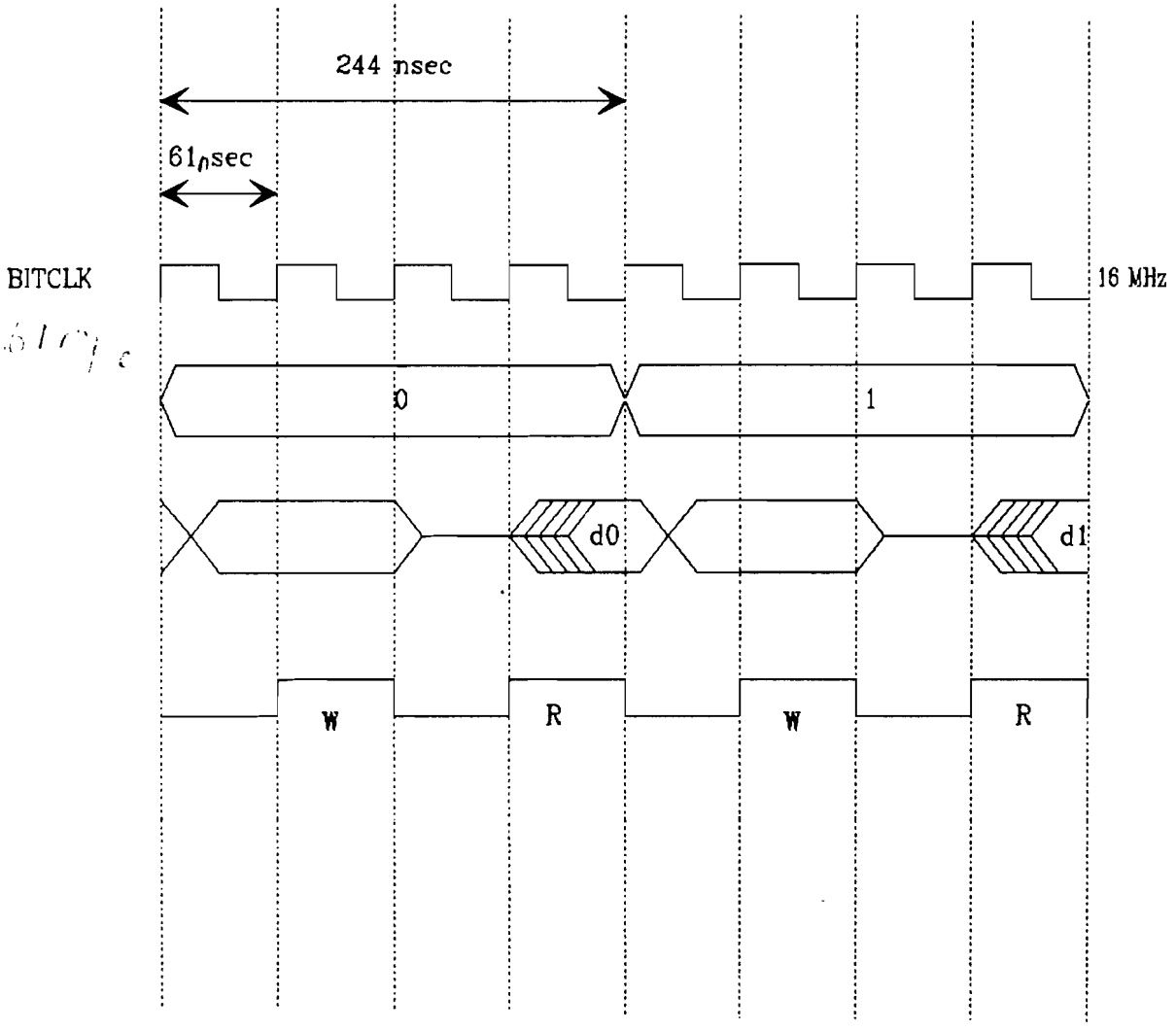
Appendix 6

Timing Diagrams Time Shared Bus Space Switch Element

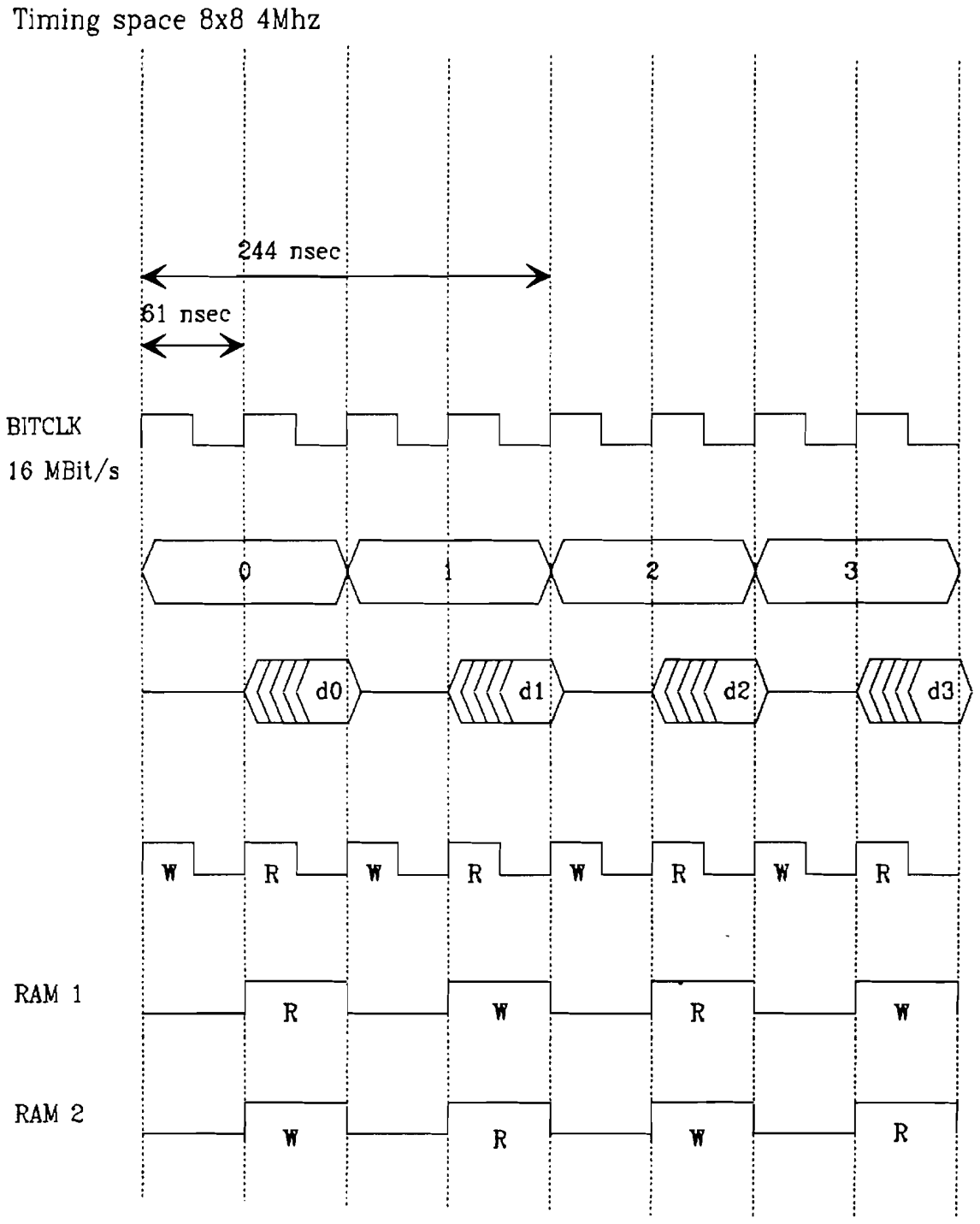
Contents

6.1 Time Shared Bus Space Switch Element 4x4 4 MBit/sec

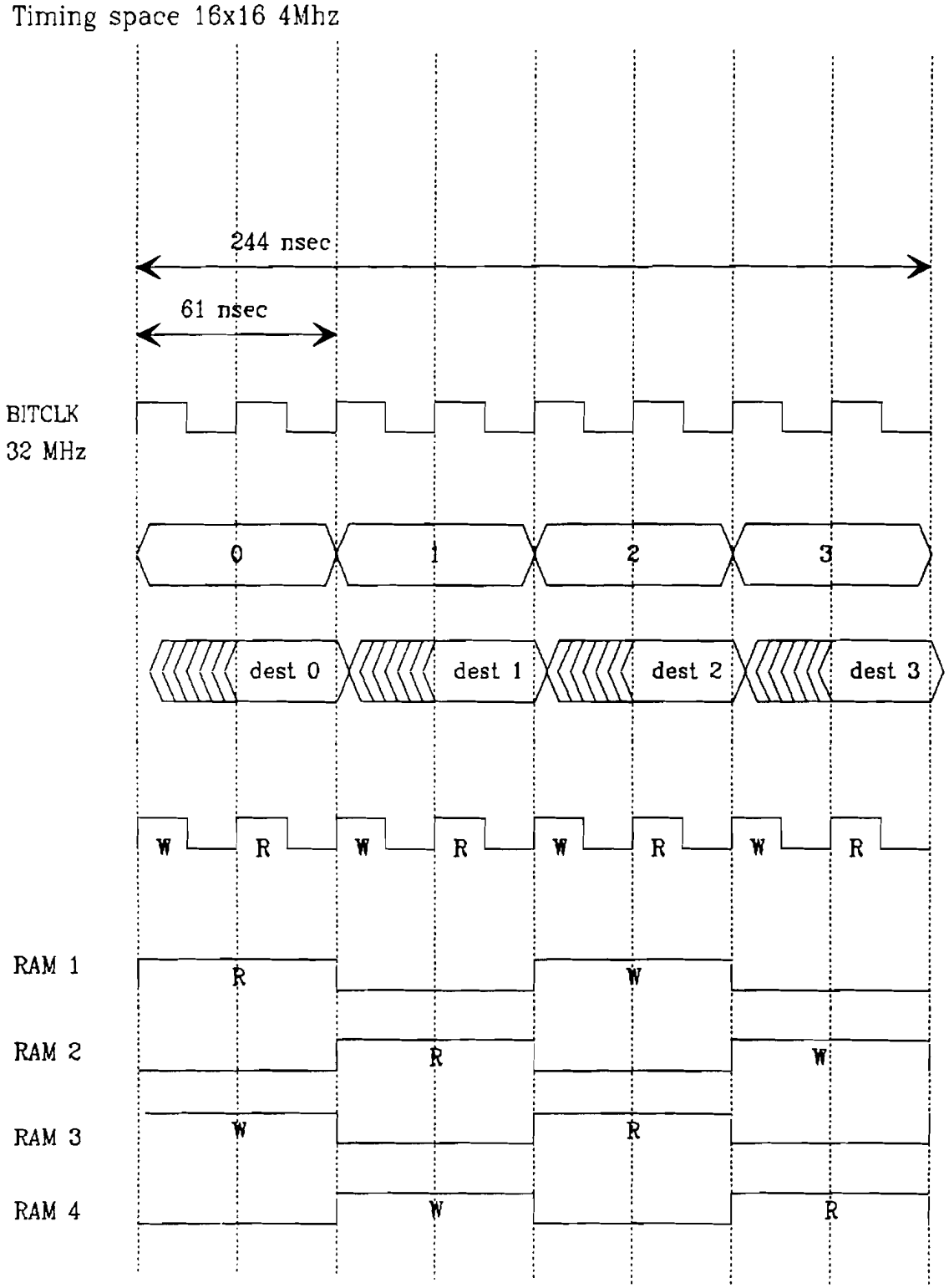
Timing space 4x4, 4MHz



6.2 Time Shared Bus Space Switch Element 8x8 4 MBit/sec

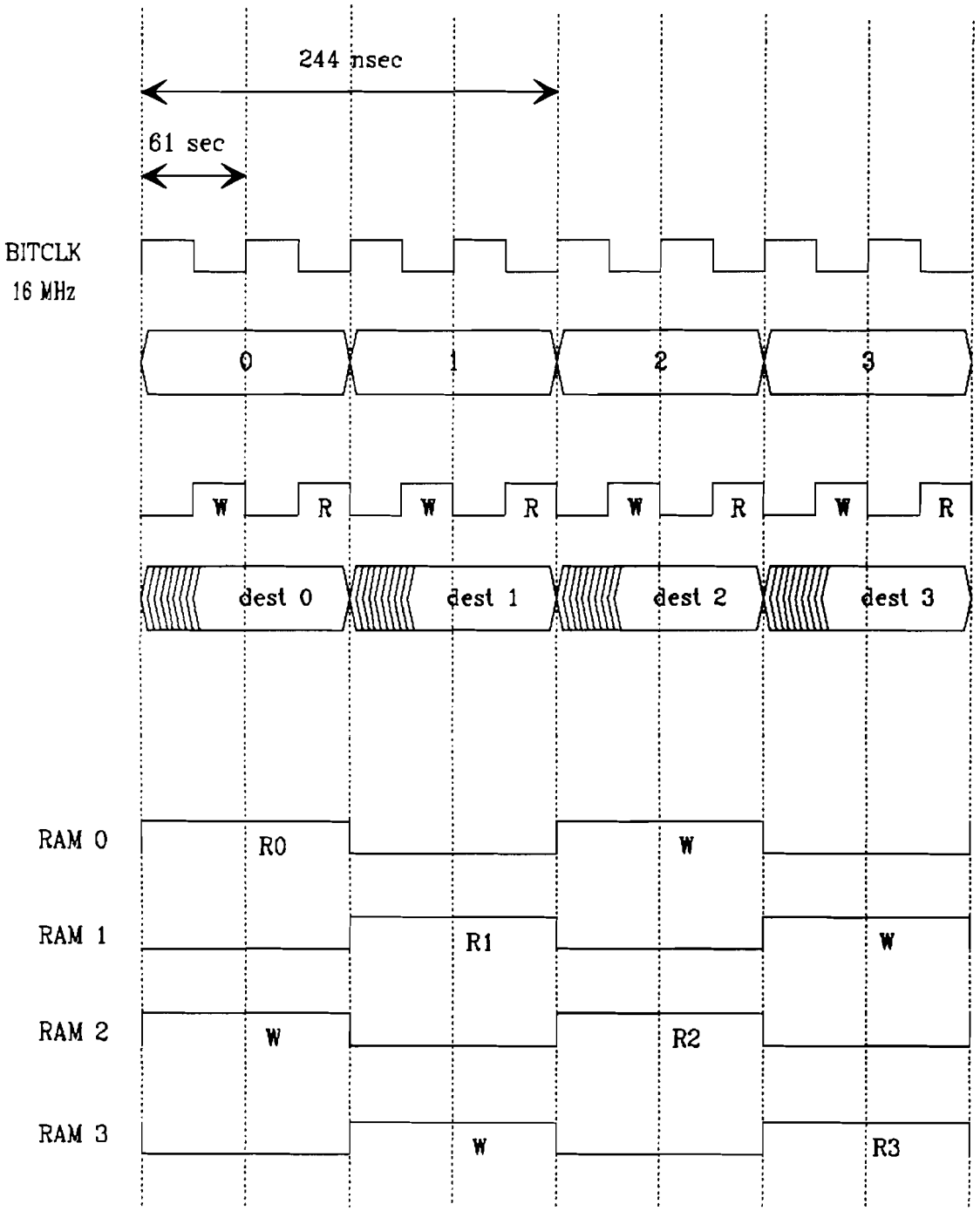


6.3 Time Shared Bus Space Switch Element 16x16 4 MBit/sec

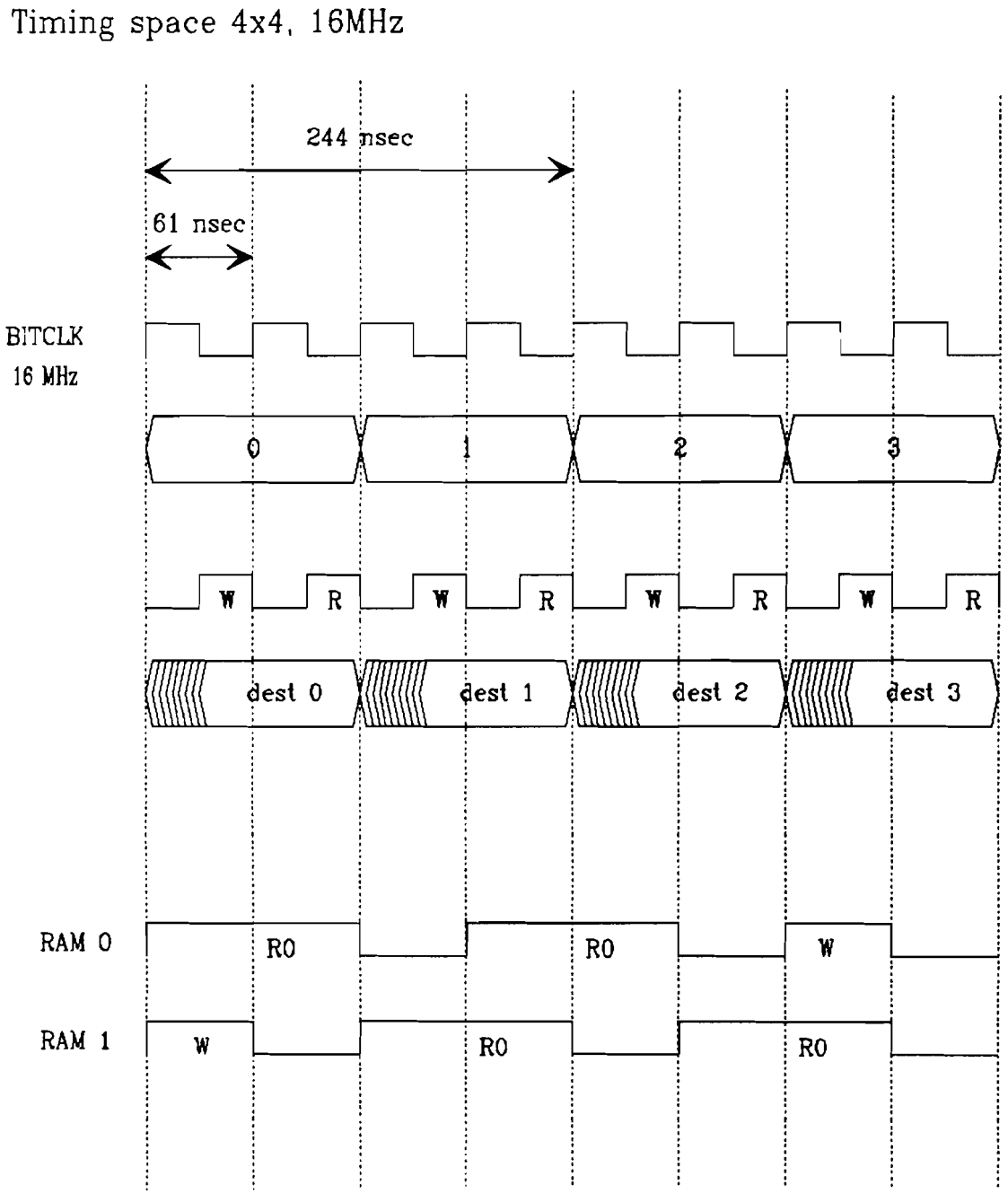


6.4 Time Shared Bus Space Switch Element 4x4 16 MBit/sec, 4 interleaved RAM blocks

Timing space 4x4, 16MHz

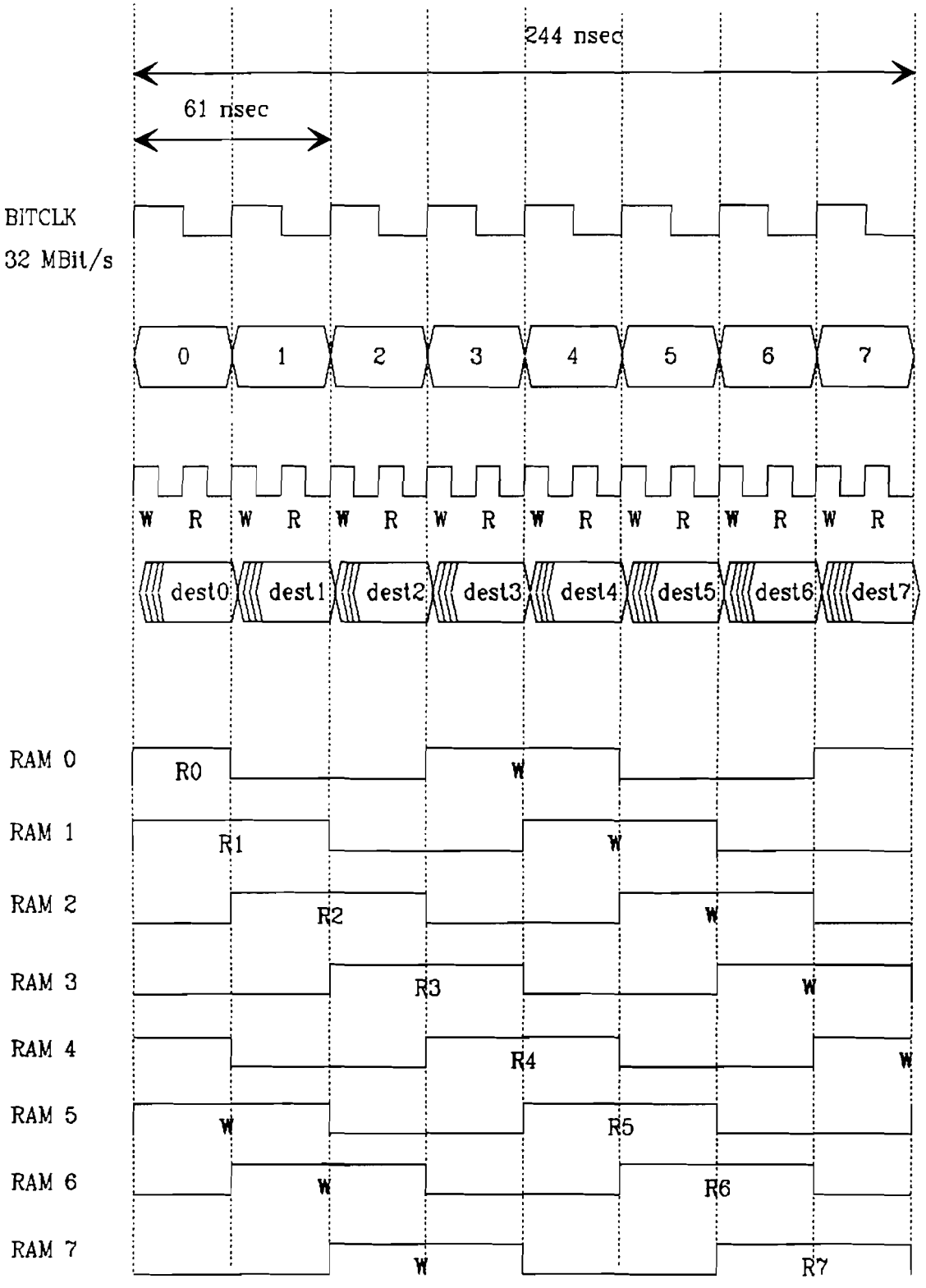


6.5 Time Shared Bus Space Switch Element 4x4 16 MBit/sec, 2 interlaeved RAM blocks



6.6 Time Shared Bus Space Switch Element 8x8 16 MBit/sec

Timing space 8x8 16Mhz

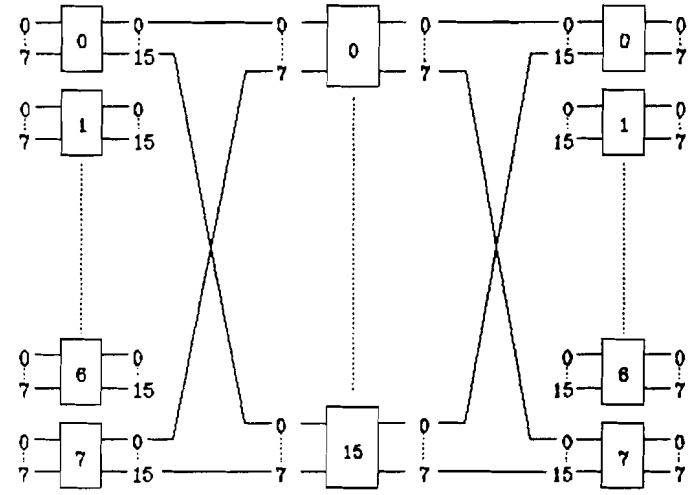


Appendix 7

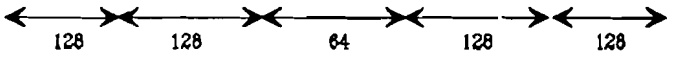
Top Down Design Space Switch Element

Contents

7.1 SSS network



Bedrading :



RAM		
8 x 128 x 32	16 x 128 x 24	8 x 128 x 48
43.52 mm2	66.88 mm2	65.52 mm2
Transistors		
17.536	37.584	32.488

Totaal

176 mm2 RAM +

87.608 transistors +

bedrading +

controle logica

Figure 7.1 *SSS Network*

7.2 Space Switch Element 8x16

7.2.1 Space8x16tot

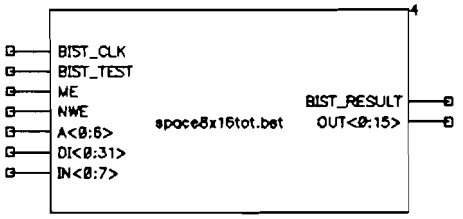


Figure 7.2 *Symbol Space Switch Element 8x16*

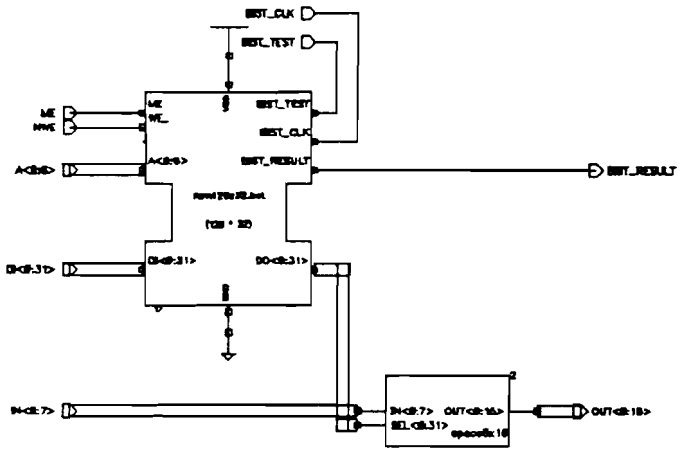


Figure 7.3 *Schematic Space Switch Element 8x16*

7.2.2 Space8x16

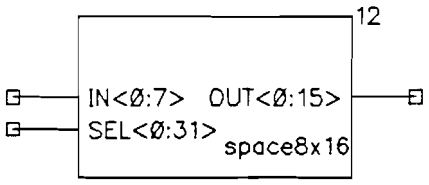


Figure 7.4 *Symbol Space Switch Element 8x16*

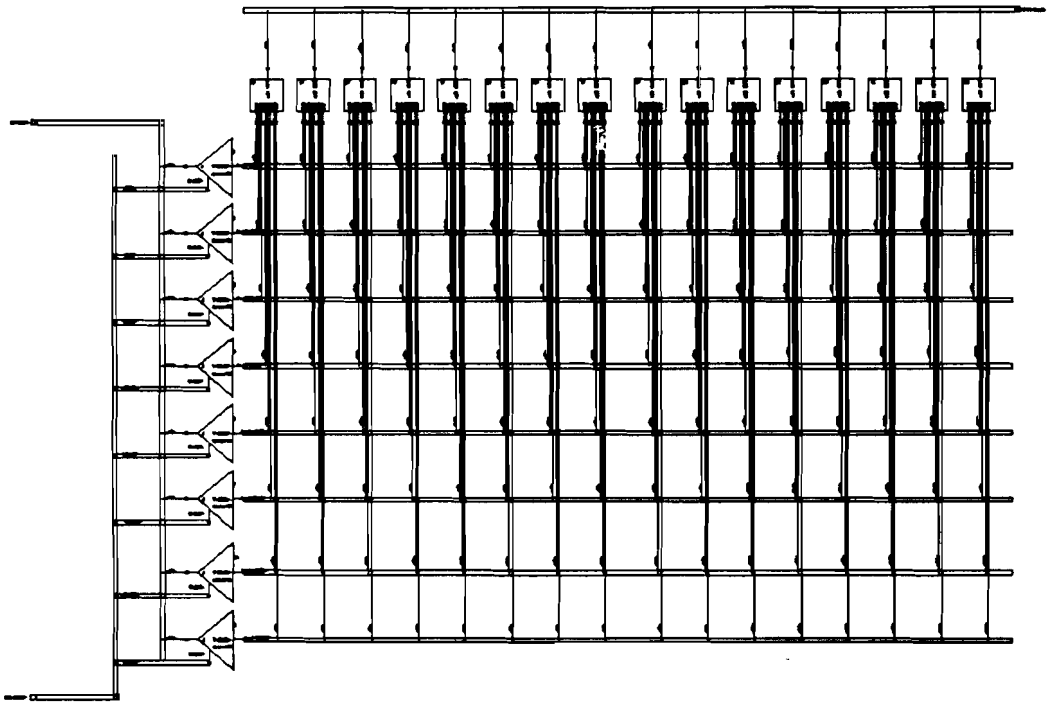
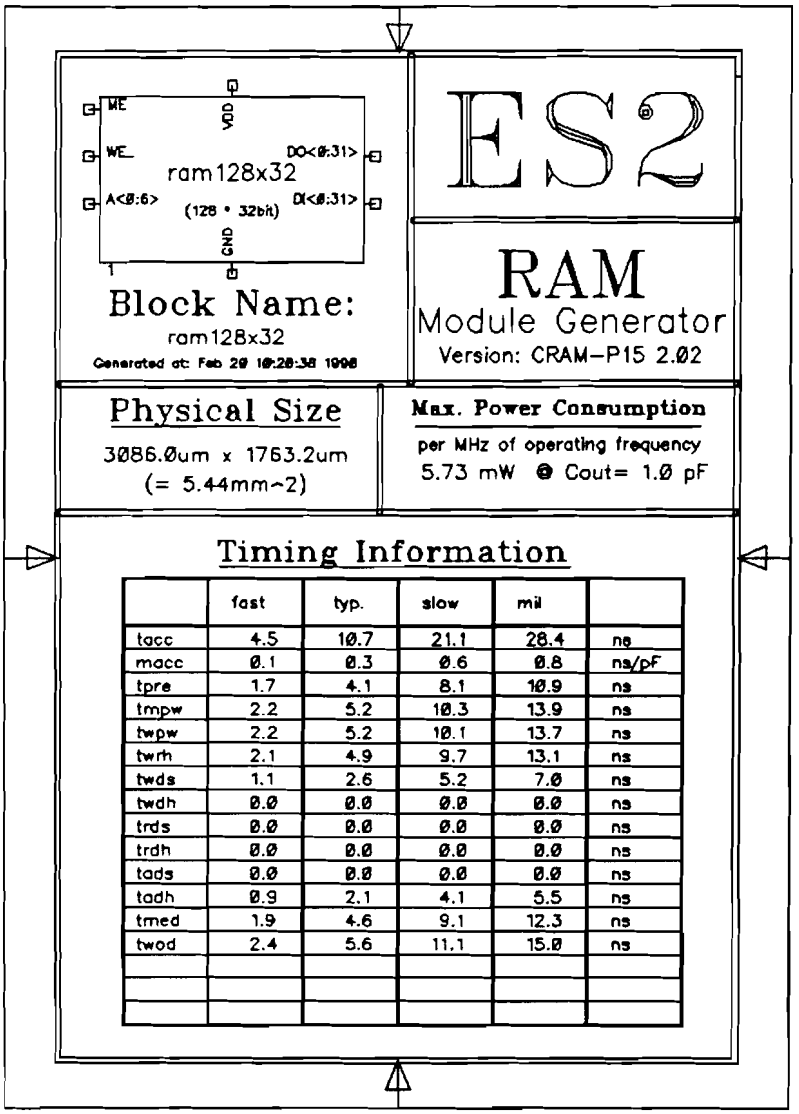


Figure 7.5 *Schematic Space Switch Element 8x16*

7.2.3 RAM 128x32



	fast	typ.	slow	mil	
tacc	4.5	10.7	21.1	28.4	ns
macc	0.1	0.3	0.6	0.8	ns/pF
tpre	1.7	4.1	8.1	10.9	ns
tmpw	2.2	5.2	10.3	13.9	ns
twpw	2.2	5.2	10.1	13.7	ns
twrh	2.1	4.9	9.7	13.1	ns
twds	1.1	2.6	5.2	7.0	ns
twdh	0.0	0.0	0.0	0.0	ns
trds	0.0	0.0	0.0	0.0	ns
trdh	0.0	0.0	0.0	0.0	ns
tads	0.0	0.0	0.0	0.0	ns
tadh	0.9	2.1	4.1	5.5	ns
tmed	1.9	4.6	9.1	12.3	ns
twod	2.4	5.6	11.1	15.0	ns

Figure 7.6 *RAM 128x32*

7.3 Space Switch Element 8x8

7.3.1 Space 8x8tot

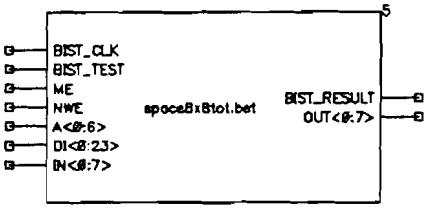


Figure 7.7 *Symbol Space Switch Element 8x8*

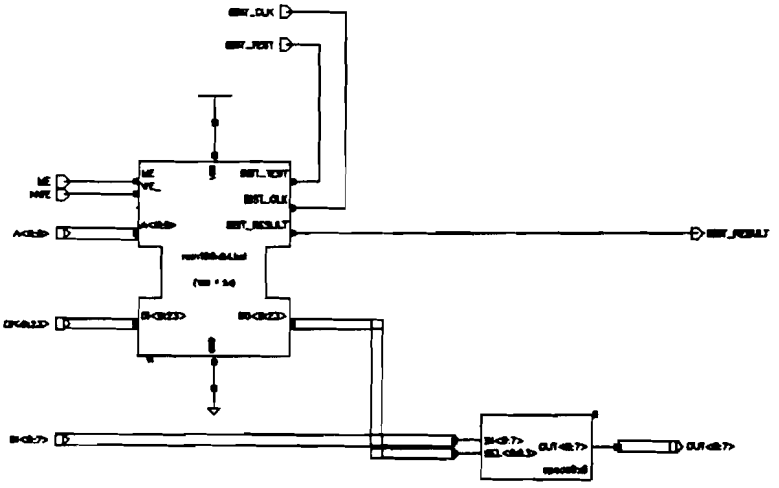


Figure 7.8 *Schematic Space Switch Element 8x8*

7.3.2 Space 8x8

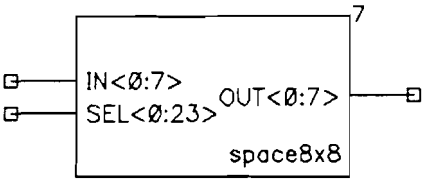


Figure 7.9 *Symbol Space Switch Element 8x8*

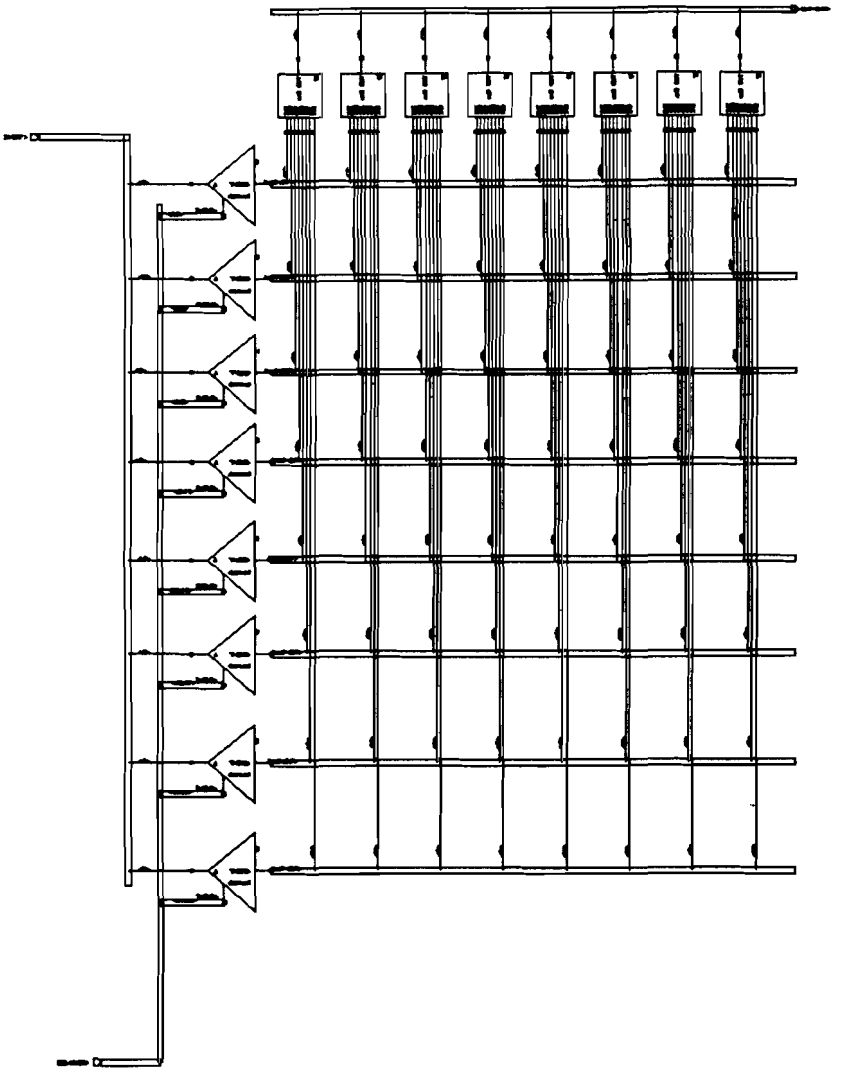
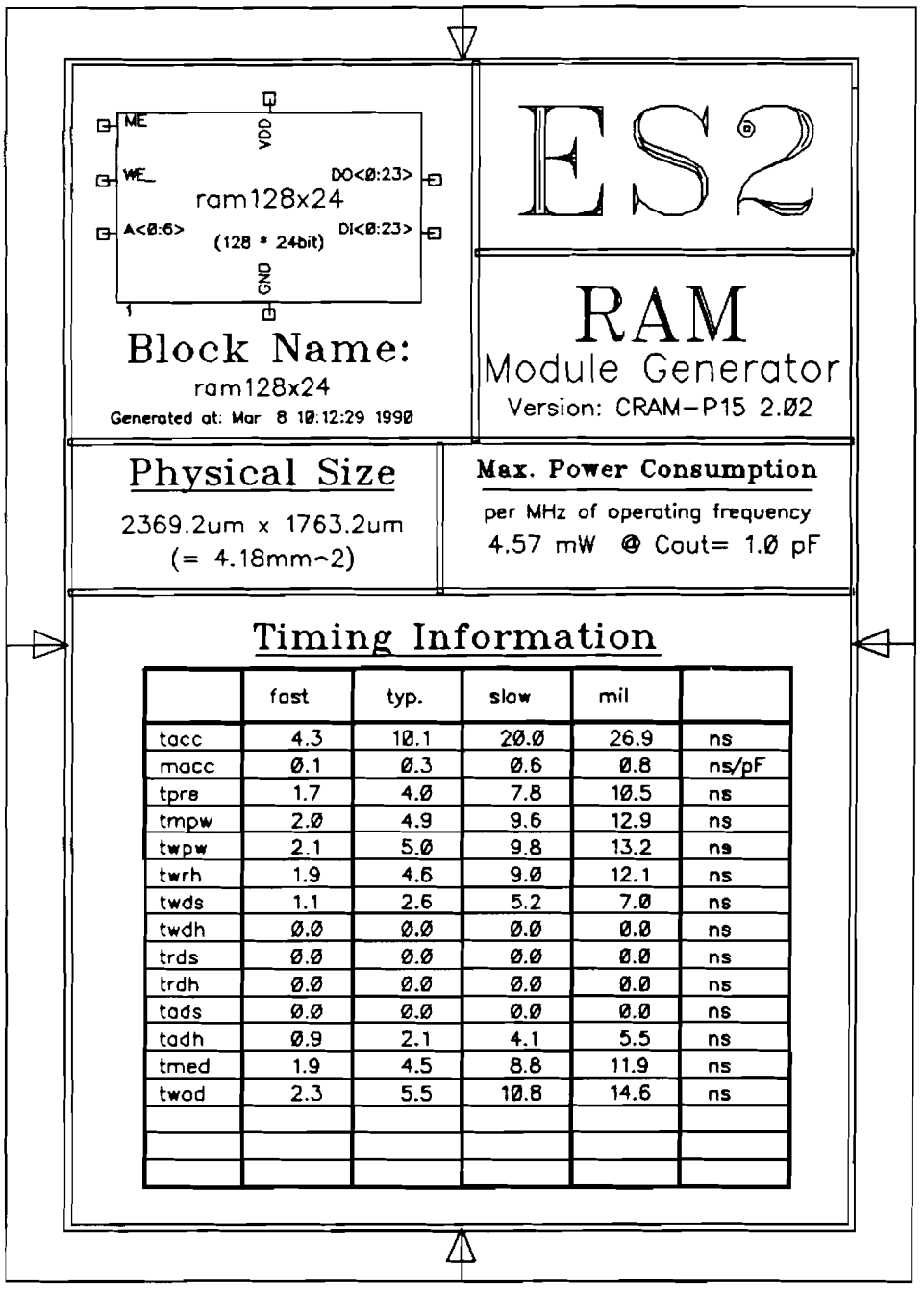


Figure 7.10 *Schematic Space Switch Element*

7.3.3 RAM 128x24



	fast	typ.	slow	mil	
tacc	4.3	10.1	20.Ø	26.9	ns
macc	Ø.1	Ø.3	Ø.6	Ø.8	ns/pF
tpre	1.7	4.Ø	7.8	10.5	ns
tmpw	2.Ø	4.9	9.6	12.9	ns
twpw	2.1	5.Ø	9.8	13.2	ns
twrh	1.9	4.6	9.Ø	12.1	ns
twds	1.1	2.6	5.2	7.Ø	ns
twdh	Ø.Ø	Ø.Ø	Ø.Ø	Ø.Ø	ns
trds	Ø.Ø	Ø.Ø	Ø.Ø	Ø.Ø	ns
trdh	Ø.Ø	Ø.Ø	Ø.Ø	Ø.Ø	ns
tads	Ø.Ø	Ø.Ø	Ø.Ø	Ø.Ø	ns
tadh	Ø.9	2.1	4.1	5.5	ns
tmed	1.9	4.5	8.8	11.9	ns
twod	2.3	5.5	10.8	14.6	ns

Figure 7.11 *Datasheet RAM 128x24*

7.4 Space Switch Element 16x8

7.4.1 Space16x8tot

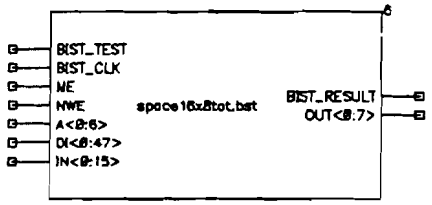


Figure 7.12 *Symbol Space Switch Element 16x8*

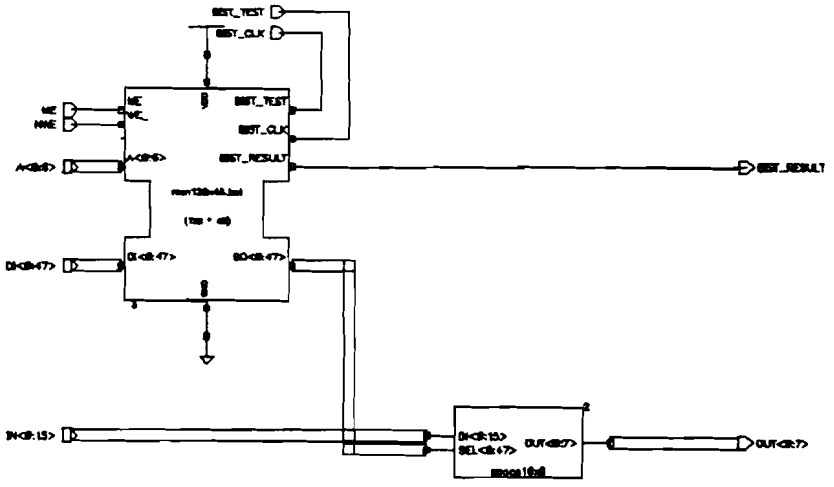


Figure 7.13 *Schematic Space Switch Element*

7.4.2 Space16x8

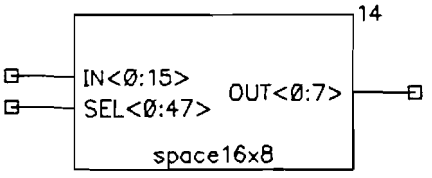


Figure 7.14 *Symbol Space Switch Element 16x8*

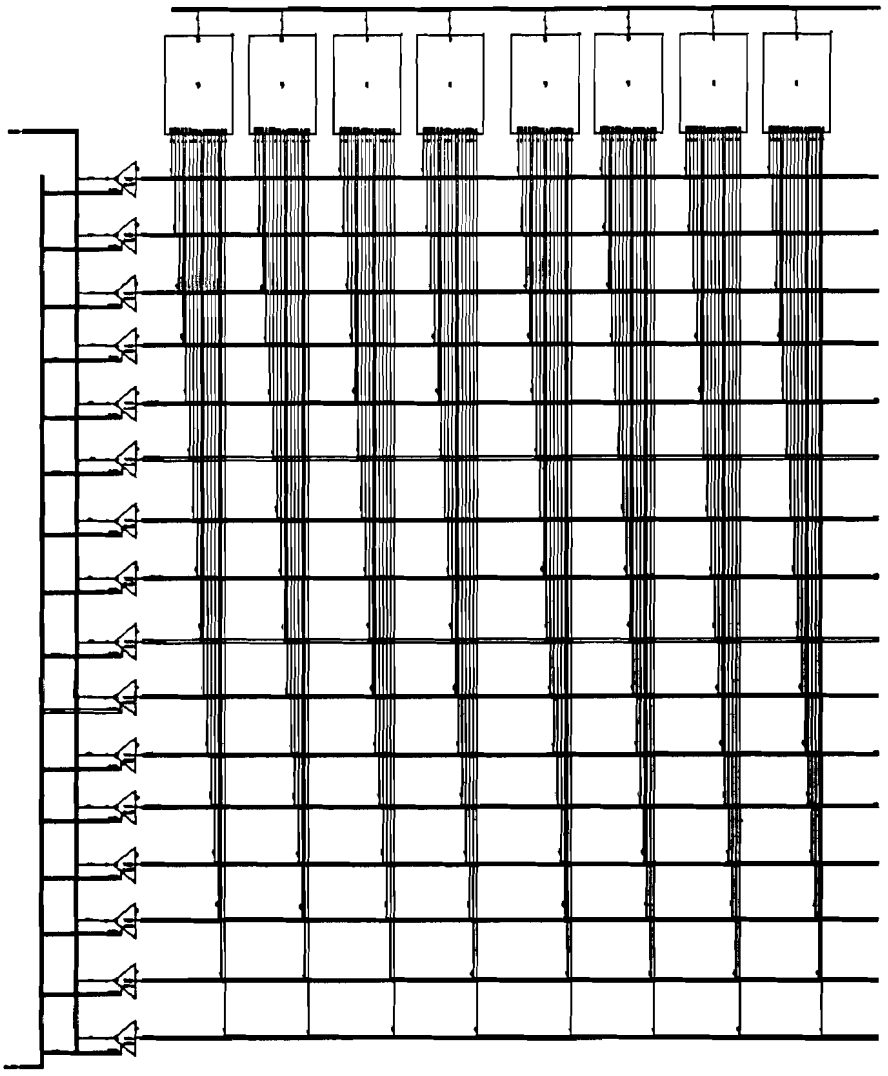
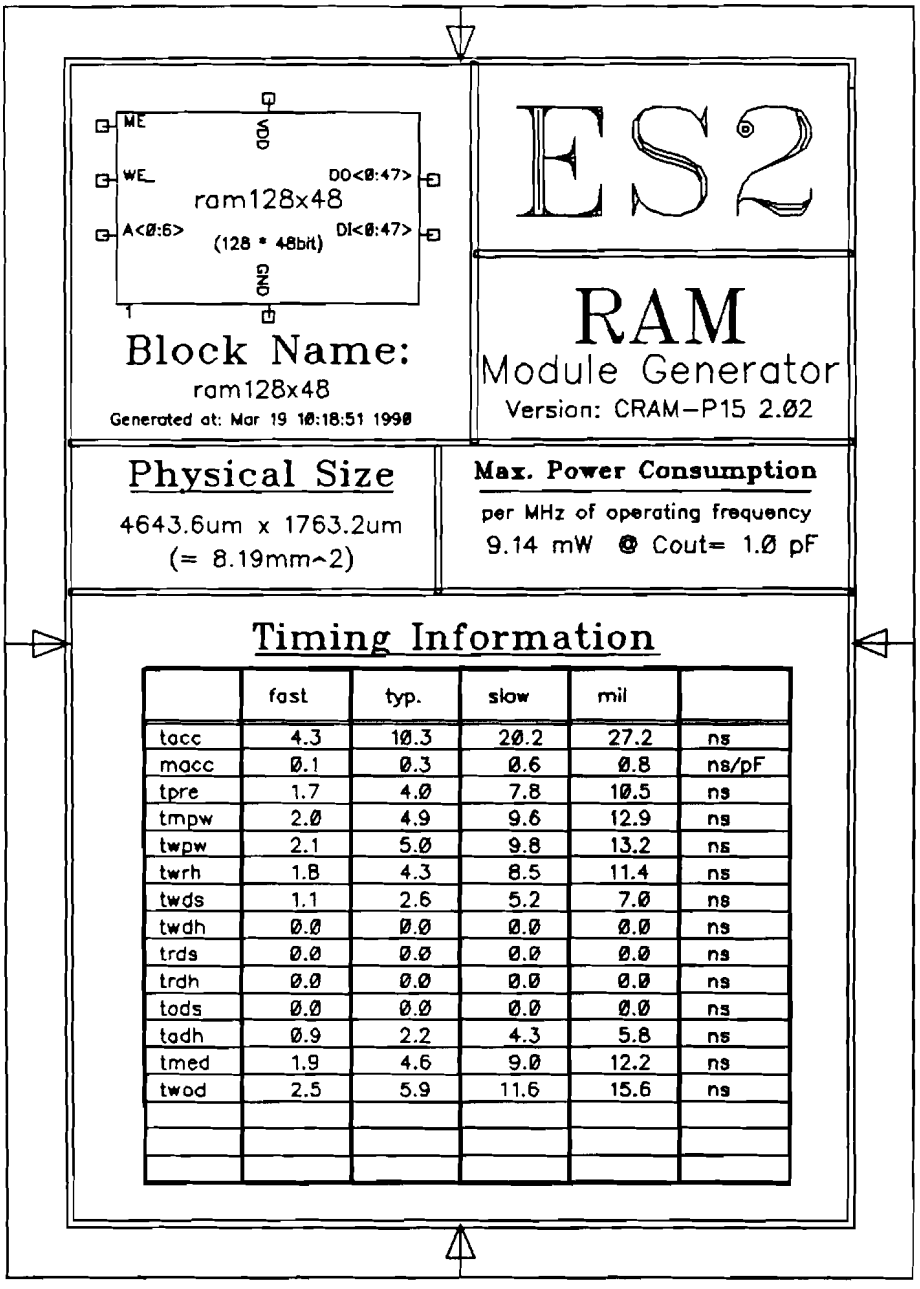


Figure 7.15 *Schematic Space Switch Element 16x8*

7.4.3 RAM 128x48



	fast	typ.	slow	mil	
tacc	4.3	10.3	20.2	27.2	ns
macc	Ø.1	Ø.3	Ø.6	Ø.8	ns/pF
tpre	1.7	4.Ø	7.8	10.5	ns
tmpw	2.Ø	4.9	9.6	12.9	ns
twpw	2.1	5.Ø	9.8	13.2	ns
twrh	1.8	4.3	8.5	11.4	ns
twds	1.1	2.6	5.2	7.Ø	ns
twdh	Ø.Ø	Ø.Ø	Ø.Ø	Ø.Ø	ns
trds	Ø.Ø	Ø.Ø	Ø.Ø	Ø.Ø	ns
trdh	Ø.Ø	Ø.Ø	Ø.Ø	Ø.Ø	ns
tads	Ø.Ø	Ø.Ø	Ø.Ø	Ø.Ø	ns
tadh	Ø.9	2.2	4.3	5.8	ns
tmed	1.9	4.6	9.Ø	12.2	ns
twod	2.5	5.9	11.6	15.6	ns

Figure 7.16 *Datasheet RAM 128x48*

7.5 General Library Elements

7.5.1 Demux 1 to 16

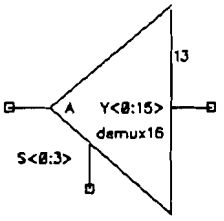


Figure 7.16 *Symbol Demultiplexer 1 to 16*

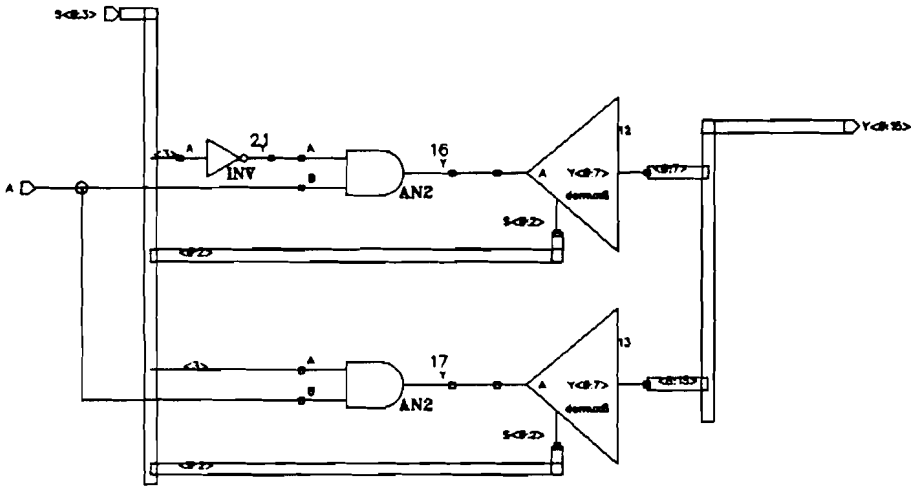


Figure 7.17 *Schematic Demultiplexer 1 to 16*

7.5.2 Demux 1 to 8

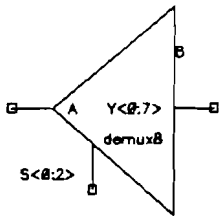


Figure 7.18 *Symbol Demultiplexer 1 to 8*

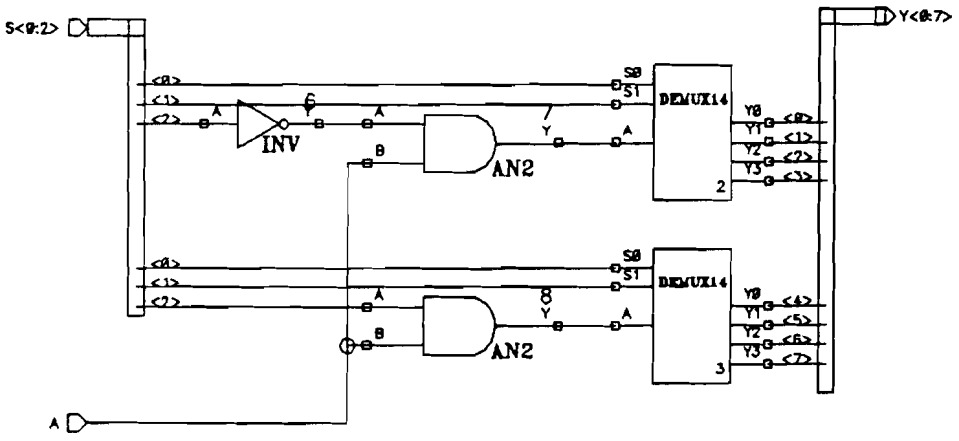


Figure 7.19 *Schematic Demultiplexer 1 to 8*

7.5.3 Or16

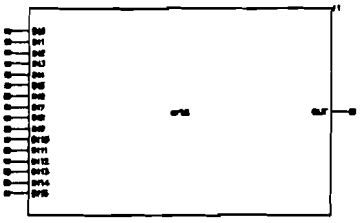


Figure 7.20 *Symbol Or 16*

Figure 7.21 *Schematic Or 16*

7.5.4 Or8

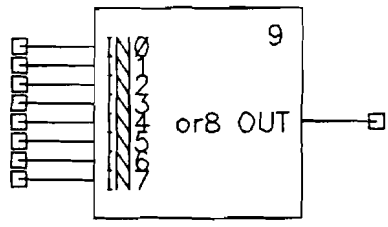


Figure 7.22 *Symbol Or 8*

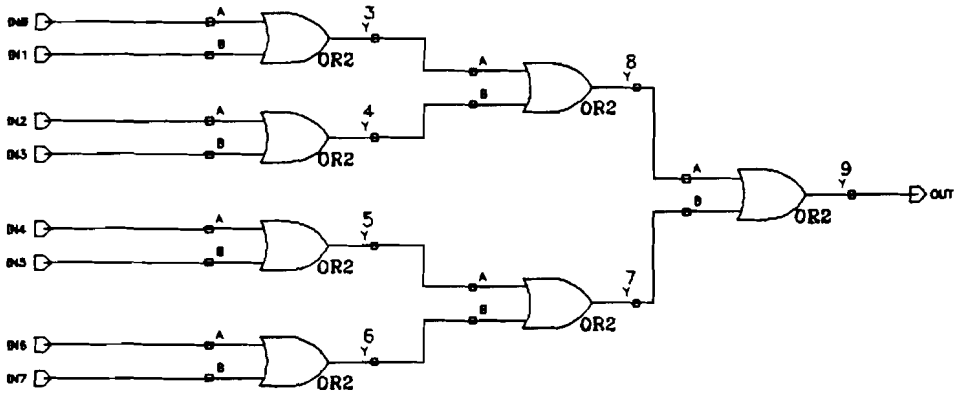


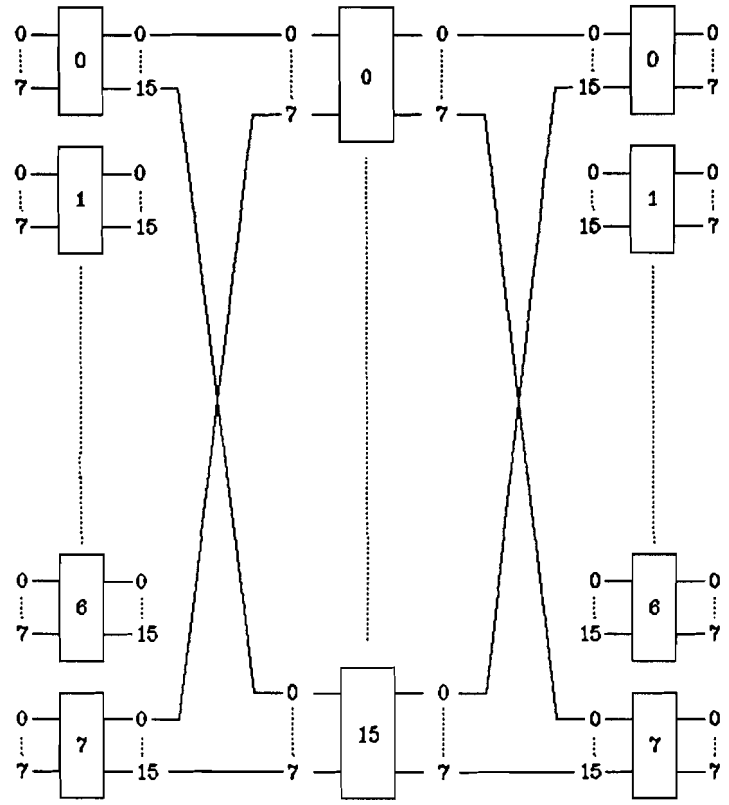
Figure 7.23 *Schematic Or 8*

Appendix 8

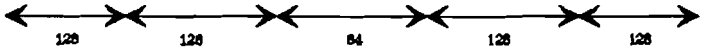
Calculation Size of SSS Network

Contents

8.1 SSS Network Schematic



Bedrading :



RAM :

8 x 128 x 8 x 4	16 x 128 x 8 x 3	8 x 128 x 16 x 3
8 x 128 x 32	16 x 128 x 24	8 x 128 x 48
8 x 4096	16 x 3072	8 x 6144
32.768	49.152	49.152

Figure 8.1 *SSS Network Schematic*

8.2 SSS Network Size

Number of transistors

space 8x16	8 x 2192	17.536
space 8x8	16 x 2349	37.584
space 16x8	8 x 4061	32.488
total		87.608

Size of the RAM

space 8x16	8 x 128x32 bits	8 x 5.44 mm^2	43.5 mm^2
space 8x8	16 x 128x24 bits	8 x 4.18 mm^2	66.8 mm^2
space 16x8	8 x 128x48 bits	8 x 8.2 mm^2	65.2 mm^2
total			175 mm^2

total size

transistors	87.609 x 2.7 10^{-10} m^2
RAM	175 mm^2
total	200 mm^2