DESIGN OF 128 BIT ROUND ROBIN PRIORITY ENCODER

By

KIRAN RAJ JOSHI

Bachelor of Engineering

Institute of Engineering

Kathmandu, Nepal

2000

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
MASTER OF SCIENCE
July, 2004

DESIGN OF 128 BIT ROUND ROBIN PRIORITY ENCODER

Thesis Approved:

[illegible]

Thesis Advisor

[illegible]

[illegible]

Alfred Carlozzi

Dean of the Graduate College

ACKNOWLEDGEMENTS

I would like to express my sincere appreciation to my father and mother. I would also like to thank my brother and my sisters for the moral support they have provided me over the years.

I would like to thank Dr. Daniel Grischkowsky for financially supporting me as well as providing inspiring guidelines throughout my study here in Oklahoma State University.

I would like to express my sincere appreciation to Dr. Louis Johnson for guiding me through this thesis and for his wonderful class lectures. I also like to extend my appreciation to my committee member Dr. Gary Yen for providing the encouragement to carry on the thesis.

I would also like to thank my friends for making my stay in Stillwater memorable.

TABLE OF CONTENTS

Chapter Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

NOMENCLATURE

iSLIP	Iterative Serial Line IP
CAM	Content Addressable Memory
RRPE	Round Robin Priority Encoder
SLPE	Simple Linear Priority Encoder
PPE	Programmable Priority Encoder
BSPE	Barrel Shift Based Priority Encoder
KSG	Kill Signal Generator
ESG	Encoded Signal Generator
PLPE	Programmable Linear Priority Encoder
HDL	Hardware Description Language
MPE	Modular Priority Encoder
IMPE	Improved modular priority encoder
PMPE	Pipelined MPE
CUT	Circuit Under Test

Chapter 1

Introduction and Thesis organization

1.0 Thesis Introduction

This report presents a design of a 128 bit round robin priority encoder (RRPE), with a delay of order O(logN) and also presents an implementation method that greatly minimizes the effect of interconnect delays.

The round robin encoder has the basic structure shown in figure 1.1.

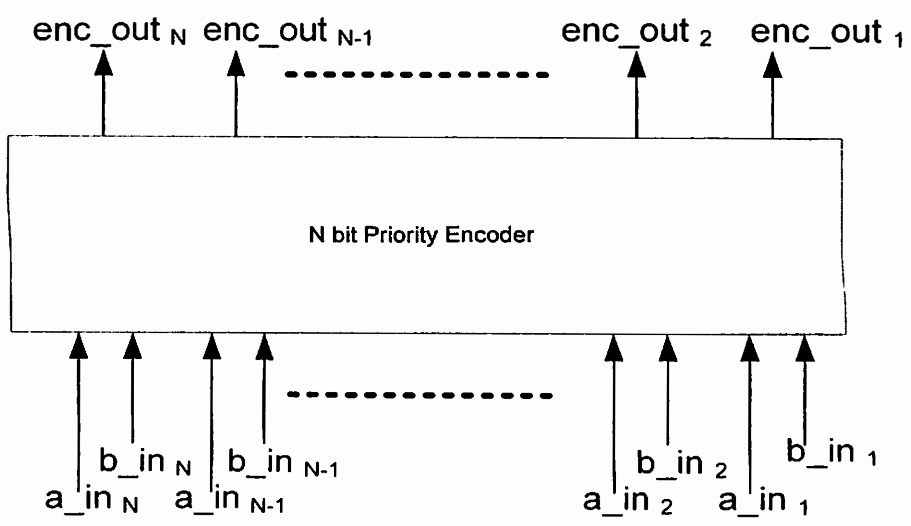


Figure 1.1: N bit Round Robin Priority Encoder

a_in_j , j=N,..., 1 , are the N inputs to the encoder, the inputs can be either one or zero at anytime depending on the request at that particular input. enc_out_i, i= N,..., 1 represents the N outputs from the circuits.

b_in_j, j=N,..., 1, are the N signals that denote which input signal has the highest priority. The input with the highest priority is denoted by resetting the corresponding begin signal to '0'. For example, if the a_in_i input is to be the highest priority, then the b_in_i signal is zero. At any time, only one of the b_in_j inputs can be zero, the rest of them should be one, which implies that only one of the inputs can be the highest priority input at anytime. The priority then decreases gradually from the highest priority input to the lowest in a round robin fashion from right to the left. Thus, if a_in_i has the highest priority, then input $a\_in_{\mathrm{mod}N(i+1)}$ will have the next highest priority and so on. Thus, the input $a\_in_{\mathrm{mod}N(i-1)}$ will have the lowest priority. Figure 1.2 shows the pictorial representation of the priority assignment.

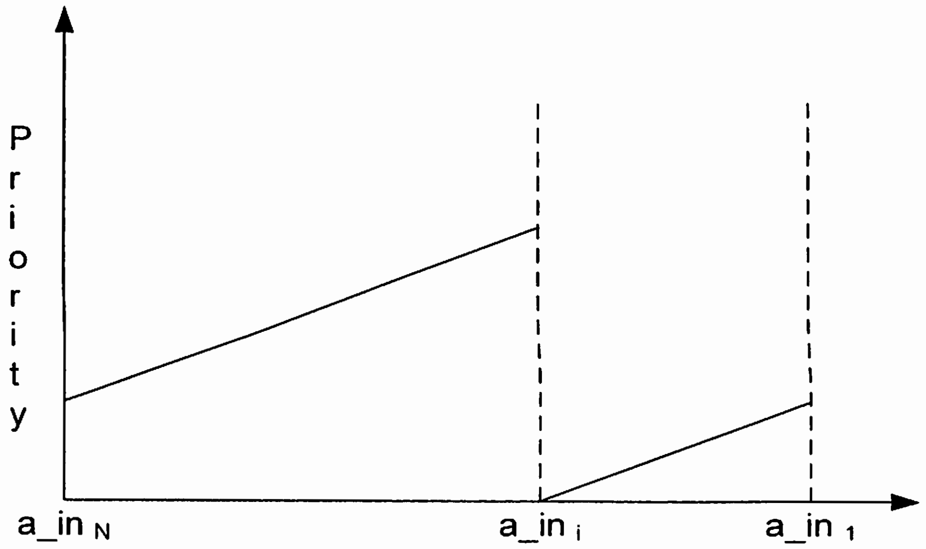


Figure 1.2: Assignment of priority in N bit RRPE with input a_in_i set to highest priority

1.1 Scope and previous studies

Several studies of Round Robin priority encoders with large numbers of inputs have been done in the past [1] [2]. In [1] [2] designs have been presented that break the circuits into smaller modules, which helps to simplify the design complexity. A similar method has been used in this study to design the Round Robin encoders. This method greatly simplifies the design and reduces the design time. In [1] all the proposed circuits have been implemented using Altera's ACEX1K series CPLD. The delay for a 64 input was 25.5 ns. They have also simulated the circuit proposed by [2] for which the delay of 39.9 ns has been shown for the encoder of size 64 bits.

Most of the circuits in the current study makes use of the tree of AND and OR gates. Use of tree structures to minimize the delay through the circuit is well known. They have been used in the design of look ahead carry generators [5] [6] [7]. The concepts of look ahead signals have also been used in the design of fast linear priority encoders [8] [9] [10] [11]. The concept of the look ahead signal has also been utilized in the design of the round robin priority encoder, in order to obtain the delay O(logN), where N denotes the number of bits in the encoder.

All of the circuits that have been designed in this study have been for 128 bit round robin encoders and have been implemented using either the Verilog hardware description language or using the Cadence Virtuoso layout editor. Therefore a direct comparison of area and speed with the previously done work is difficult, since different technologies are used

The proposed round robin encoder finds application in several different fields. They have been used in the design of high speed switches and scheduling [1]. For fast switching and fair control of congested flow in broadband networks [12].

There have been many designs of round robin priority encoders, which have been used in the realization of iterative serial line IP (iSLIP) and in the area of high speed packet switches with input buffers [2] [12]. They have also been used in the Content Addressable Memory (CAM) circuits [14] [15]. Most of these designs have been implemented for a relatively small number of inputs, of the order of 10. Although these designs have delays of order O(logN), if the same designs are expanded to a large number of inputs of the order of 100, the delay will dramatically increase. The main reason for the increase in the delay is because of the increase in the delay of the interconnect network. The use of a tree type structure indicates that the delay increases exponentially with the increase in the number of inputs.

The round robin encoder can be incorporated inside the modern superscalar processors to improve its performance. One of the motivations for the study of the round robin priority encoder was to access if the layout for the encoders with large inputs is reasonable enough to be incorporated inside the superscalar process to improve the performance. At the same time the circuit should be fast enough to perform its operation within a clock period of the processor.

1.2 Design Approaches

Two novel features of the design have been implemented. The first approach is based on the principle of a look-ahead signal, similar to the one used in look ahead carry adder circuits. The look-ahead approach is very easy to implement when the number of inputs to the encoder is small. As the number of inputs to the encoder increases, the delay in the look ahead carry as well as the circuit area are dominated by the cross over network and the wrap around circuits, which makes them impractical for a large number of inputs.

A modular approach in which the circuit is broken down into smaller sub blocks has been designed. The delay for the modular approach is very close to the O(log N).The delay for this design is given by $2\log L + \log M$, where M and L are related by N as $N=M\times L$. There is a large reduction in the area of the chip as compared to the look-ahead approach. Also many of the sub blocks used in the modular approach are similar in structure. Thus, these repeated blocks need to be designed only once, which makes the implementation of the modular approach less time consuming as compared to the look-ahead approach.

1.3 Thesis Organization

This document is divided into seven sections. Chapter 2 discusses the various design approaches available for the design of the round robin priority encoder. At the end of this chapter, a novel approach is also presented. Chapter 3 discusses the improved modular approach of priority encoder design. In Chapter 4, implementation details for the various designs are presented. Chapter 5 presents the implementation detail of the layout of the improved modular priority encoder circuit in 40 pin pad frame. Chapter 6 presents the testing and verification methods and the simulation results. Chapter 7 discusses the results of the simulation and the future work that can be carried on to improve the performance.

Chapter 2

Design Approaches

In this chapter, two different methods of the implementation is described. The main objective of the thesis is to find a design approach which has O(logN) delay. One way to achieve the targeted delay is to use a binary tree structure. Two different ways to implement a tree are given below.

2.0 General Description

The following is the description of the binary tree approach. For the explanation, purpose of a four bit Round Robin Priority Encoder (RRPE) has been described.

There are four inputs to the encoder. Each of these inputs has an associated 'begin' signal. The begin signal is used to indicate which of the input signals is of the highest priority. From the input with the highest priority, the priority gradually decreases towards the left. If the end of the circuit is reached, the priority will wrap around and will still continue to decrease in the right to left order until the input with the highest priority is reached. Thus, in this scheme, the input which is to the immediate right of the input with the highest priority will be the input with the lowest priority.

As opposed to a round robin priority encoder (RRPE) a linear priority encoder does not wrap around from the lowest priority input. Usually the priority order is fixed in the linear priority encoders. Either the leftmost input or the rightmost input will have the highest priority. For the purpose of this thesis we will assume that all the linear priority encoders have its rightmost input at the highest priority level and the leftmost input at the lowest priority level. And these encoders are referred to as the Simple Linear Priority Encoder (SLPE).

2.1 Operation of the encoder

The operation of the encoder can be completely specified by logic equations. For a fixed priority linear encoder with a_in as the input and enc_out as the output, the following equation completely defines the priority encoder. The order of the priority is such that the input with the smallest number has the highest priority. The i-th encoded output can be written as

$$enc\_out_i = a\_in_i \bullet \overline{(a\_in_{i-1} + a\_in_{i-2} + \ldots + a\_in_2 + a\_in_1)}$$
$$enc\_out_i = a\_in_i \bullet \overline{kill}_i$$

where, the ‘kill$_i$’ is the kill signal and is defined as

$$kill_i = a\_in_{i-1} + a\_in_{i-2} + \ldots + a\_in_2 + a\_in_1$$

For the round robin encoder there needs to be extra inputs which determines which of the inputs is of the highest priority. Also the priority wraps around from the circuit. This extra input can be termed as b_in. Only one of the b_in signals can be a t logic low level at any time which indicates the highest priority input, all the remaining b_in inputs must be at logic high. If a kill signal similar to the one in the linear encoder is generated then the logic equation for the kill signal is given by

$$\begin{aligned} kill_i = {} & a\_in_{i-1} \bullet b\_in_i + a\_in_{i-2} \bullet b\_in_i \bullet b\_in_{i-1} + \cdots + a\_in_1 \bullet b\_in_i \bullet b\_in_{i-1} \bullet \\ & \cdots \bullet b\_in_3 \bullet b\_in_2 + a\_in_N \bullet b\_in_i \bullet b\_in_{i-1} \bullet \cdots \bullet b\_in_1 + a\_in_{N-1} \bullet b\_in_i \bullet \\ & \cdots \bullet b\_in_1 \bullet b\_in_N + \cdots + a\_in_{N+1} \bullet b\_in_i \bullet \cdots \bullet b\_in_1 \bullet b\_in_N \bullet \cdots \bullet b\_in_{i+2} + \\ & a\_in_i \bullet b\_in_i \bullet \cdots \bullet b\_in_1 \bullet b\_in_N \bullet \cdots \bullet b\_in_{i+1} \ldots\ldots\ldots\ldots\ldots Eqn1 \end{aligned}$$

The Encoded signal is generated from this kill signal as in the linear priority encoder as

$$enc\_out_i = a\_in_i \bullet \overline{kill_i}$$

Where, N indicated the total number of inputs to the encoder, and I can be varied from N to 1. The simplest implementation of the round robin priority encoder is given in figure 2.1. The description for the encoder is presented below.

2.1.1 Request to the Encoder

Each individual input can request by setting the input, a_in$_i$, to high. Thus at any instance of time there can be several of the inputs to the priority encoder with a value of logic 1.

2.1.2 Indication of the highest priority

The input with the highest priority will have its priority input, b_in_i, as 0 all the remaining inputs will have its priority input, b_in as 1. Thus at any instance of time only one of the priority input can be 0. This ensures that there is only one, a_in input with the highest priority at any instant of time.

2.1.3 Passing of the priority token

At the beginning of the circuit operation, the priority token is with the input with the highest priority. This priority is then passed to the input in the immediate lower priority hierarchy. Thus, in its simples form the RRPE looks like the one shown in figure 2.1 below.

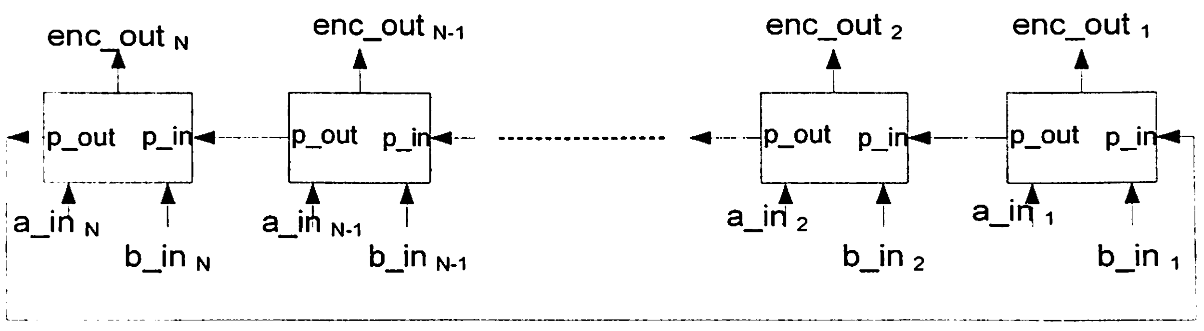


Figure 2.1: General Structure of priority encoder

If there are N inputs to the encoder, there will be N such blocks. Each Block has the following inputs and outputs.

2.1.4 Input p_in

This indicates the priority input to the block. The signal is fed from the block with the next higher priority. If the particular block is the one with the highest priority the p_in signal is ignored. This signal is like acts like a kill signal.

2.1.5 Input a_in

The input is provided by the external circuit, which is using the priority encoder circuit and serves as the request to the priority encoder.

2.1.6 Input b_in

This input is also provided by the external circuit. This is used to indicate which input has the highest priority. The assignment of the priority is according to the scheme described above. Thus only one of these b_in signals will be low, which makes that particular input signal the highest priority one. All the other b_in must be set to high so as to prevent more than one input from becoming the highest priority one at any instance of time.

2.1.7 Output p_out

Each of the blocks can send out a priority signal to the next block which is in the immediate lower priority level than itself. This signal acts as the kill signal to the next block.

2.1.8 Output enc_out

This is the output encoded signal. The signal is computed from the inputs a_in, b_in and the priority in signal, p_in.

2.1.9 Circuit Function

This type of priority encoder is called programmable priority encoder (PPE) [1]. The PPE differs from the simple priority encoder in the sense that the priority for each of the input signals can be set by an external circuitry [1]. All the encoders designed in this thesis fall in the general category of PPE as the b_in inputs can be used to indicate the priority for the input signals.

In order to understand the circuit operation, we start from the input with the highest priority and then move down toward the lower priority one.

$enc\_out_i = a\_in_i \cdot p\_int_i$

$p\_int_i = b\_in_i \cdot p\_in_i$

$p\_out_i = \sim a\_in_i \cdot p\_int_i$

Where, the operators are defined as follows

$\cdot$ Logical AND

$+$ Logical OR

$\sim$ Logical NOT

And i = N,...,1, N being the total number of the inputs to the encoder.

$p\_in_i = p\_out_{\mathrm{modN}\,(i-1)}$

Thus the priority signal acts like a token. If the input with the higher priority uses the token, the input with the lower priority cannot be served even if it requests. If on the other hand, the one with the higher priority doesn't use the token, then it passes the token to its immediate lower priority. Then the process is repeated in a similar manner for the other inputs. Thus, if there are N inputs and the request is made by only the input with the lowest priority, then, in order for the input to be granted by the scheme described above, the priority token will have to pass through each of the inputs sequentially, until it reaches the last input. Thus the worst case delay of the circuit is O(N). This delay will not be significant as long as N is small, but if N is large, then the delay will render this type of circuit unsuitable for most modern high speed applications. While the delay for the last input will take N gate delays to get served, the input with the highest priority will take only one gate delay to get served. Thus, there is a huge imbalance in the delay from the first input to the last input in the priority hierarchy.

One of the approaches to reduce this imbalance in delay is to spread the delay among the inputs so that the delay for all the inputs is approximately equal. One of the most common methods to implement this approach is to use a tree circuit which has look-ahead signals to improve the performance. Several different approaches have been proposed.

Some of the Round Robin priority encoder design is presented in the next section. The different aspects like delay, complexity and the chip area are compared for each of the design implementations.

2.2 Barrel Shift Based Priority Encoder (BSPE)

The block diagram for the BSPE is given in figure 2.2. There have been other designs of the barrel shifter based priority encoder in the past [13]. The design consists of four major blocks. The first block is a left shifter, the second block is a kill signal generator (KSG), the third block is the right shifter, and the fourth block is the encoded signal generator (ESG). The circuit functions as follows, the requests are fed to the inputs a_in and the grants are assigned to the output enc_out. The pointer b_in indicates which input is currently the highest priority. The shifters are such that the amount of shift can be varied. The pointer, b_in, is connected to the shift amount input of both shifters. Thus, the first barrel shifter shifts the inputs such that the highest priority input to the circuit is rearranged to the highest priority input of the KSG. The KSG operates on the request and produces the kill output. Since the input to the KSG is presented by the first barrel shifter such that the highest priority request always lies in its highest priority input, the programmable nature of the circuit is transparent to the KSG. The output of the KSG is in the wrong order with respect to the input requests. Thus, the process by which the order of the inputs was changed has to be reversed for the output of the KSG. Since the input was shifted left by the amount specified by the pointer b_in, the output of the KSG also has to be shifted right by the same amount. Thus, the third block, right shifter, shifts the output of the KSG to the correct order. The output of the third stage is the rearranged kill signal, from which the actual encoded signal is generated. This is done by the fourth block.

There have been several designs of the KSG, with O(log N) delay. The most commonly used shifter with O(log N) delay is the barrel shifter. Each of the blocks has been described in detail in the following section.

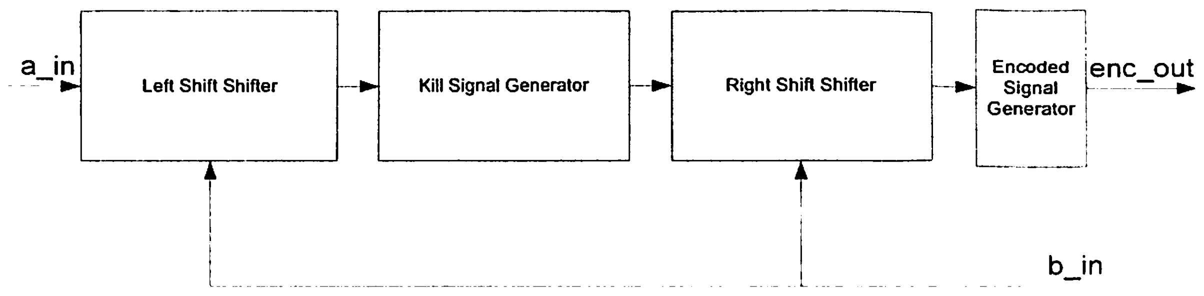


Figure 2.2: Basic Structure of a BSPE

2.2.1 Structure of Left Shift Barrel Shifter

The barrel shifter can be designed by using an overlapped tree structure as shown in figure 2.3. The extra shift_in input to the shifter is used to supply the value to the shifter which would otherwise be undefined after the shift operation. Since the shifter is being used for the round robin encoder, the shift signals should be wrapped around and connected to the left inputs as shown in figure 2.3. For an N bit input encoder, an N bit shifter is required. For an $N=2^n$ bit shifter, there are n stages of nodes. Each of these nodes is implemented with a 2 to1 multiplexer as shown in the figure 2.4. From figure 2.3, it is seen that the number of wrap arounds increase exponentially with the number of the stages. There is only one wrap around connection in stage 1, while there are two wrap around connections for the second stage. Thus for the p-th stage, there are 2^{p-1} wrap around connections. Also the number of wrap around also increases exponentially for larger inputs, which make them very difficult to layout.

Also the size of the cross over network in each stage increases exponentially. The number of bits crossed over at i-th stage is $2^{(i-1)}$. As the number of stage increases the cross over network becomes too large, and thus, becomes the major cause of delay contribution to the circuit. In order to reduce these delays, additional buffers are needed. These buffers increase the area requirement of the circuit.

The a_in input to this encoder circuit is applied to the inputs of the left shift barrel shifter. The b_in input to the encoder are decoded and are applied to the control inputs of the shifter. The Db_in signals in figure 2.3 represent the decoded signal of b_in inputs.

The inputs a_in are shifted by the amount specified by the Db_in inputs and appear at the output of the shifter in rearranged form. These rearranged inputs are termed as ra_in as shown in figure 2.3.

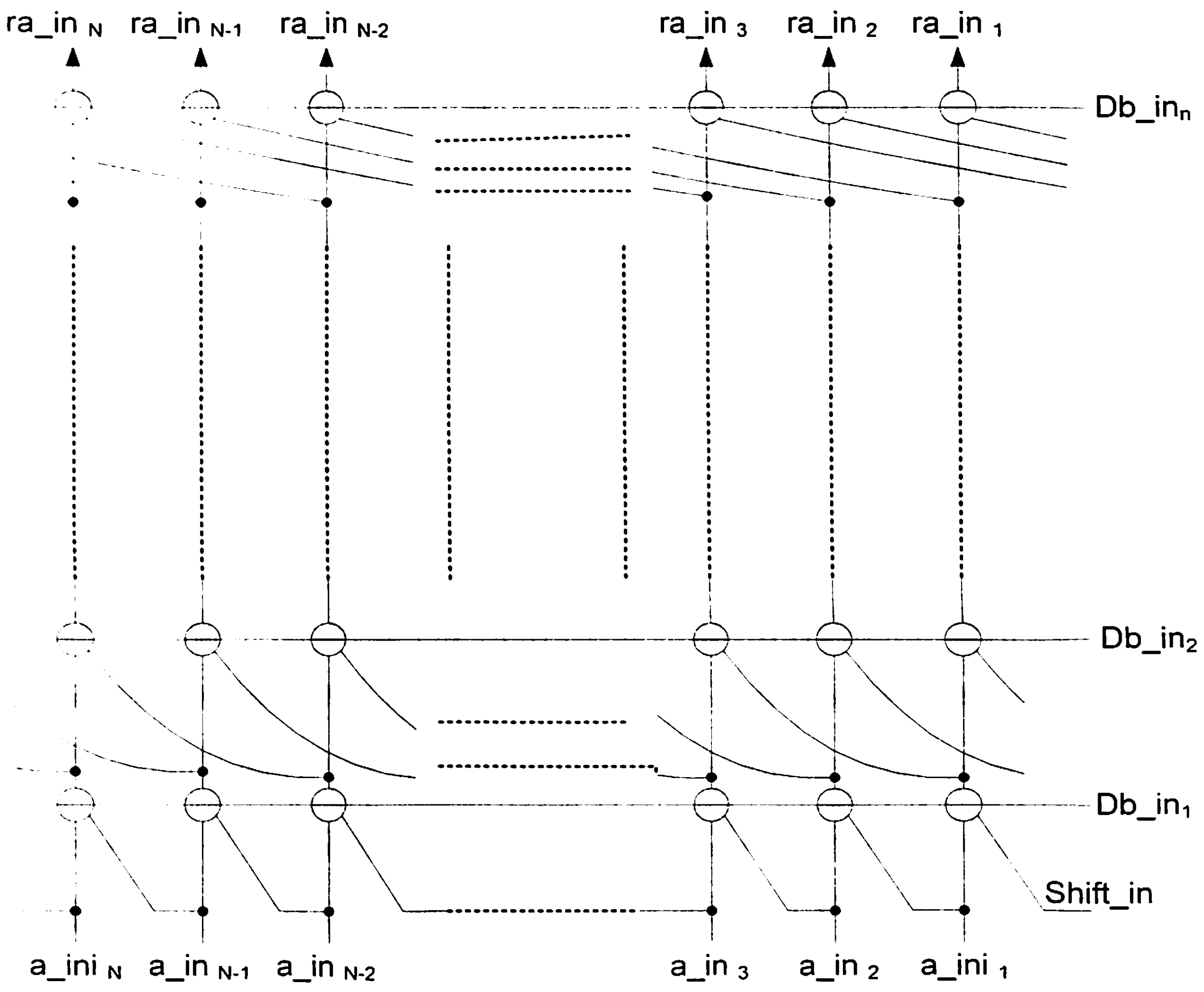


Figure 2.3: Structure of the Left Shift Barrel Shifter

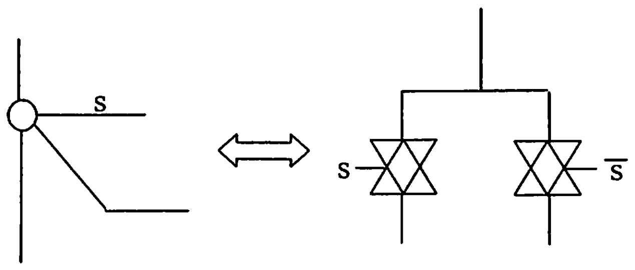


Figure 2.4: Node structure for the left shift barrel shifter

2.2.2 Kill Signal Generator (KSG)

Figure 2.5 shows the circuit for the KSG. The circuit is similar in construction to the barrel shifter described above. The KSG is similar to a fixed priority encoder, with the input on the right most corner having highest priority. The priority then decreases subsequently towards the left. Thus the leftmost input in the KSG is the one with the lowest priority. The KSG can be implemented by using a simple tree of OR gates. As shown in figure 2.5 some of the inputs to the OR gates in the right side doesn't have any connection. These inputs must be connected to ground for the proper functioning of the circuit. For a two input OR gate, if one of the input is connected to the ground, then the output will be same as the other input. Thus those OR gates in the KSG circuits can be removed. But leaving them there serves two purposes. First of all it makes the circuit symmetrical, the second is that it serves as a buffer to drive the crossover networks. As in the case of the barrel shifter, as the number of stages increases the crossover network also increases exponentially, and eventually dictate the overall delay of the circuit. The cross over network also increases the complexity in the layout of the circuit as the number of inputs increases.

The KSG receives it input from the left shift register, which is the rearranged for of the input a_in to the encoder. From figure 2 it is seen that if the right most input, ra_in 1 is at logic 1 then all the outputs, k_1, k_2, k_3 and k_4 will be at logic 1, irrespective of the state of the other inputs, ra_in $_4$, ra_in $_3$ or ra_in $_2$. If ra_in $_1$ is at low and ra_in $_2$ is at high then k1 will be at low but k_2, k_3 and k_4 will be at logic high irrespective of the logic levels of the other two inputs ra_in $_4$ or ra_in $_3$. The output signals are the kill signals. Kill signals start at the position where the highest priority input signal is at logic 1 and then propagate all the way to the lowest priority position.

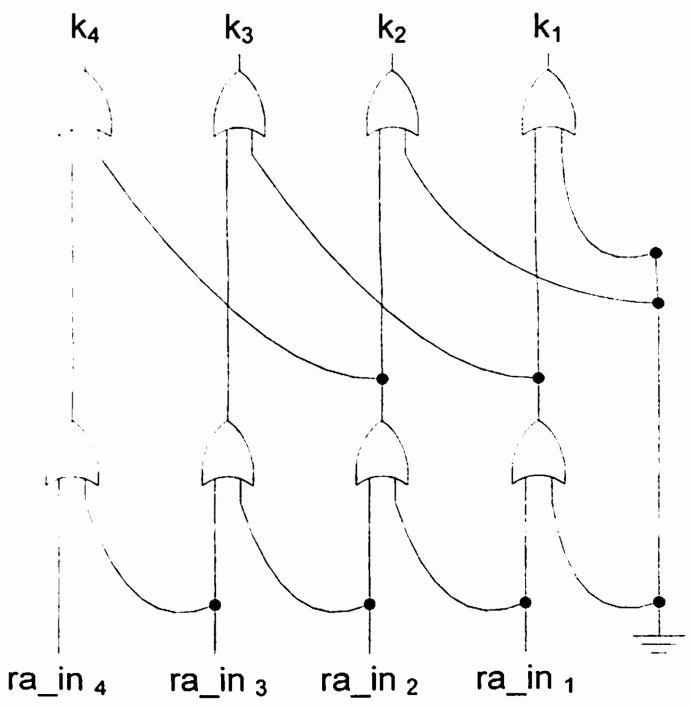


Figure 2.5: Structure for a four input KSG

2.2.3 Right Shift Barrel Shifter

Figure 2.6 shows the diagram for the right shift barrel shifter. The barrel shifter is similar to the left shift barrel shifter shown in figure 2.4. The only difference is in the connection of each of the nodes, so that the circuit shown in figure 2.6 will perform a right shift of the inputs by the amount specified by the input S. All the problems associated with the left shifter will also be present in the design and implementation of the right shift barrel shifter.

The input to the stage provided by the KSG, which are the k signals. These are the kill signals generated for the rearranged input signals ra_in. The right shifter again shift it by the amount specified by the amount by which the a_in was shifted but in opposite direction to get the correct pattern of the kill signals. Thus the output from the right shifter is the kill signal which corresponds to the input signal a_in to the encoder. The control signal Db_in to the right shifter is the decoded version of the b_in signals.

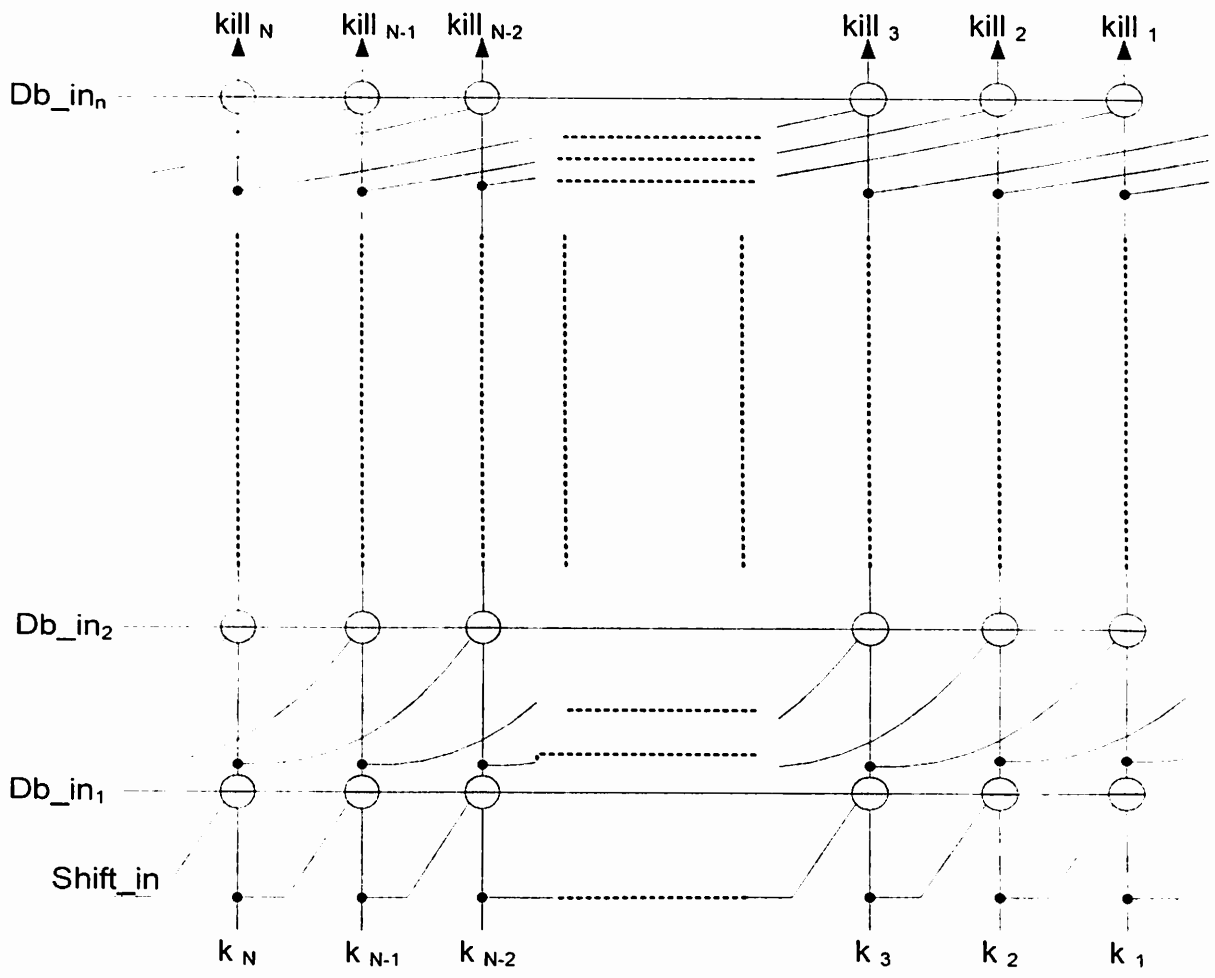


Figure 2.6: Structure for a right shift barrel Shifter

2.2.4 Encoded Signal Generator (ESG)

This block simply takes the kill signals that has been rearranged by the right shift buffer and then combines with a_in and b_in signal to generate encoded output, en_out.

The following logic operations are performed for each of the kill signals.

$k\_int_i = kill_i . b\_in_i$

$enc\_out_i = \sim k\_int_i \cdot a\_in_i$

where, i represents the i-th input position, and i runs from N to 1 for an N bit encoder.

2.2.5 Performance evaluation

The delay for the above circuit can be estimated by adding the delay through the three blocks. As mentioned above, all three of the blocks have approximately log N gate delays, where N signifies the number of the inputs. Thus the total delay of the circuit is roughly 3 log N gate delays thus limiting the speed at which the BSPE can operate. As the number of inputs to the circuit increases, the cross over network in each of the blocks increases exponentially. For the higher levels in each block, the cross over network will present a large capacitive load to the driving node, which significantly increases the delay. In order to reduce the delay, buffer circuits have to be inserted at carefully chosen places in the cross over networks. This need for the insertion of the buffer greatly increases the complexity of the circuit. Also, the area greatly increases as there are three identical blocks where these buffers have to be added at several points. Also, since each of the level in each block is different than any other, if additional inputs are required, the circuit has to have a new layout. This makes this approach less suitable for applications with a large number of inputs.

2.3 Look Ahead Tree Priority Encoder (LATPE)

The general equation to generate the kill signal for a round robin priority encoder is given in section 2.1. It is seen from the equation that to generate the kill signal two logic operations are involved. One is the AND operation and the next is the OR operation. These two operations are interlaced as given by equation 1 in section 2.1. One method to compute this type of interlaced AND-OR operation has been used in the construction of the look ahead carry generator [5] [6] [7]. Also the concept of lookahead signals have been implemented in the design of large linear priority encoders [8] [9] [10] [11]. A tree implementation of the circular priority encoder, which is same as round robin priority encoder in functionality is also presented in reference [12].

In order for the circuit to perform at logN delay a novel circuit configuration based on the binary tree structure has been designed. The basic structure of the circuit is shown in figure 2.8. The circuit can be divided into two different planes: the AND plane and the OR plane. The OR plane and the AND plane are very similar in construction. Both of these networks have a tree structure. The OR plane is exactly similar to the KSG shown in figure 2.5. While the AND plane is similar to the Right Shift barrel shifter shown in figure 2.6, with each node being replaced by an AND gate instead of the multiplexer. Figure 2.7(a) shows the structure for the OR plane and figure 2.7(b) for the AND plane. The LATPE circuit can be considered as the interlaced combination of the AND and the OR plane. These two planes when interlaced will generate the kill signal.

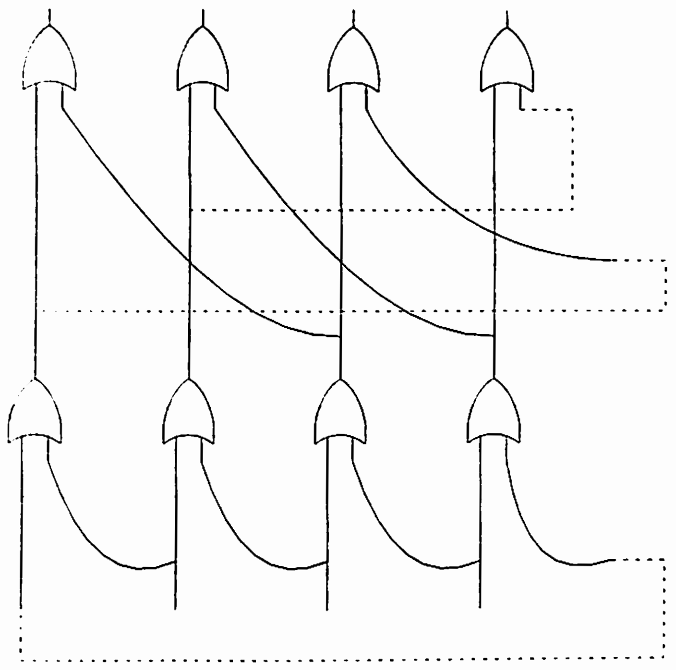


Figure 2.7 (a): OR Plane

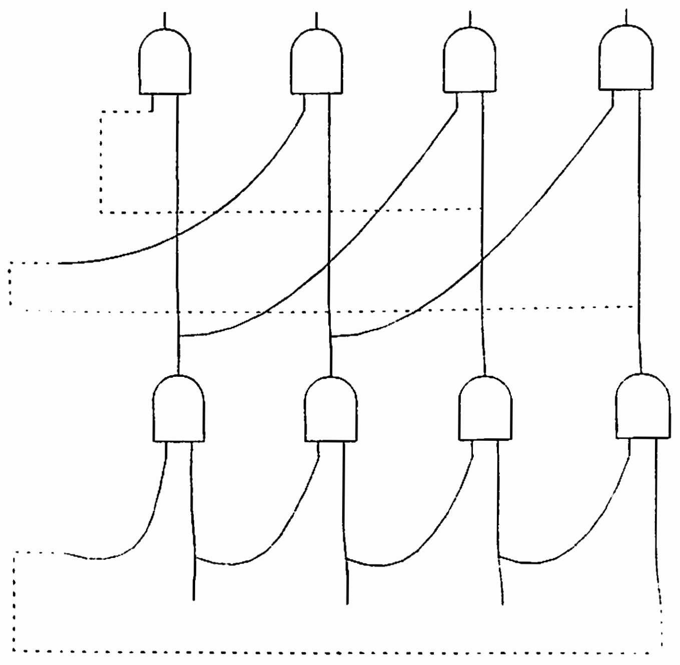


Figure 2.7 (b): AND plane

For an N input LATPE, where $N=2^n$, there are n+1 levels. Each of the nodes in the encoder is divided into several categories. Type 1 nodes are the nodes in level 1. The nodes in level 2 through level n are called type 2. The nodes in level n+1 are named type 3. The inputs to the OR plane are called 'a_in' and the inputs to the AND plane are called 'b_in'. Any external circuit which uses the encoder will choose the pattern for the a_in. For example if there are N different inputs vying for a common resource, then the request to the resource is made by setting the corresponding a_in inputs to 1. The b_in inputs define the priority of the inputs at any instance of time. For an N input encoder there are N b_in signals, each associated with the corresponding a_in input. The highest priority input is indicated with its corresponding b_in input as logic 0. Thus at any instance of time, one and only one of the b_in inputs can be 0. All the rest of the b_in inputs should be set at logic 1.

The grant to the resource for a particular input is indicated by turning its corresponding 'enc_out' signal high. Since at a time only one of the inputs can have access to the resource, only one of the N enc_out signals can be at logic 1 at any instant of time. All the remaining outputs should be at logic 0.

The inputs to type 2 nodes in the i-th level and p-th column are made from three other nodes.

- nodes of (i-1)-th level and p-th column
- nodes of (i-1)-th level in $\mathrm{mod}(p - 2^{i-1})_N$-th column
- nodes of (i-1)-th level in $\mathrm{mod}(p + 2^{i-1})_N$-th column

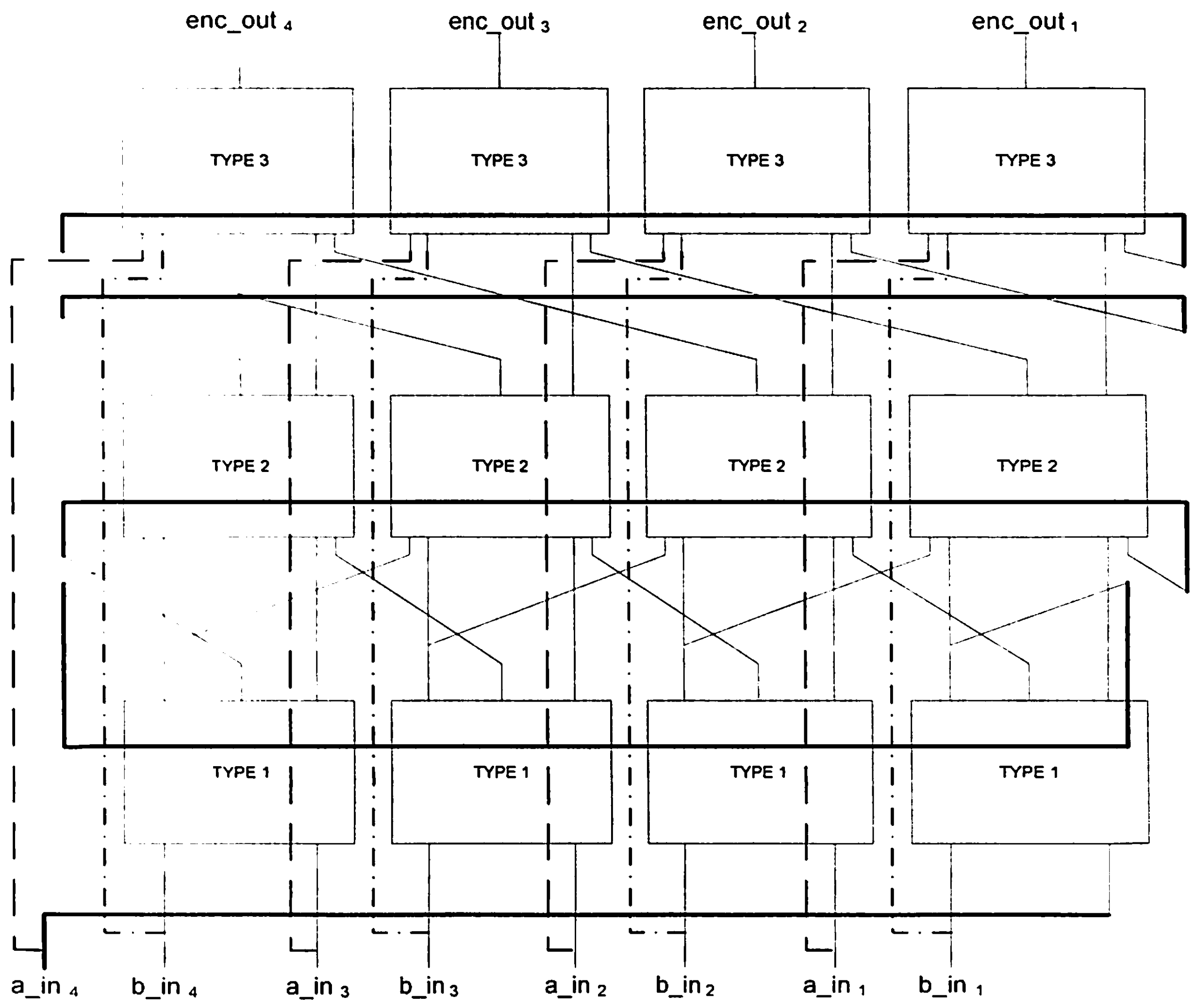


Figure 2.8: Four-input LATPE

Thus at any k-th level of the LAPTE there are 2k-1 cross over from the left end which wrap around to the right side and there are 2k-1 crossover from the right which wraparound to the left of the circuit. As the number of inputs increases, the number of levels also increases, and also the number of crossovers and wrap arounds. For each higher level, the length of the crossover also doubles as compared to the crossover in the preceding level. Thus, part of the loading for each node increases exponentially for each additional level of nodes. This will have two implications on the overall circuit performance. For an encoder with a large number of inputs, the circuit area for the higher level is mostly dominated by the cross over networks alone. Appendix B shows the layout of the LATPE for the 7th stage where the interconnect networks are 15 times larger than the circuit elements. Also the delay of the circuit increases dramatically because of the large capacitive load these cross over network presents to the nodes driving it. In order to obtain optimum delay through the circuits, extra buffer circuits have to be added to the driving nodes. The buffer has the general structure as shown in figure 2.9.

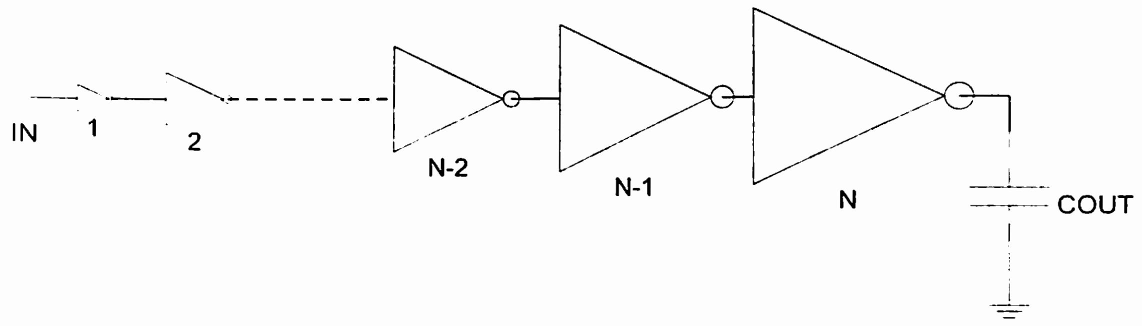


Figure 2.9: N Stage Buffer circuit

The general form of the equations to determine the number of buffer stages needed and the ratio of the buffer size between consecutive stages can be formed. Solving these equations provides the number of stages required for optimum delay. The ratio of buffer size between the consecutive stages is usually not same between stages, but the solution for such a case will be very complicated and often requires a computer program for the solution. A simpler case which is suitable for the hand calculation can be obtained for the case when the size ratio between the consecutive stages is fixed. Such a result is presented below.

$$a^N = \frac{COUT}{\left(\frac{C_G}{A}\right)W_1 L} \equiv Y$$

$$N = \frac{\ln Y}{\ln a}$$

Where,

'N' is the number of buffer stages

'a' is the fixed size ratio between the consecutive stages

$\left(\frac{C_G}{A}\right)$ is the capacitance per unit area for the poly gate

'W_1' is the width of the transistor in the first buffer stage

'L' is the channel length of the transistor used in the buffer

'COUT' represents the load being driven by the buffer stages

Usually 'a' is a fixed number with a value between 3 and 5 for a close to optimum delay through the buffer stages.

The structure for the Type 1, type 2 and the type 3 nodes are given in the following sections.

2.3.1 Type 1 Node

Type 1 nodes are the simplest of all the nodes. They have only one AND gate in them. None of the type 1 nodes are directly connected to each other. The input to the type 1 nodes comes from the external circuitry, and the output of the node always goes to the input of the type 2 node in level 2. Figure 2.10 shows the structure for the type 1 nodes for a four input encoder.

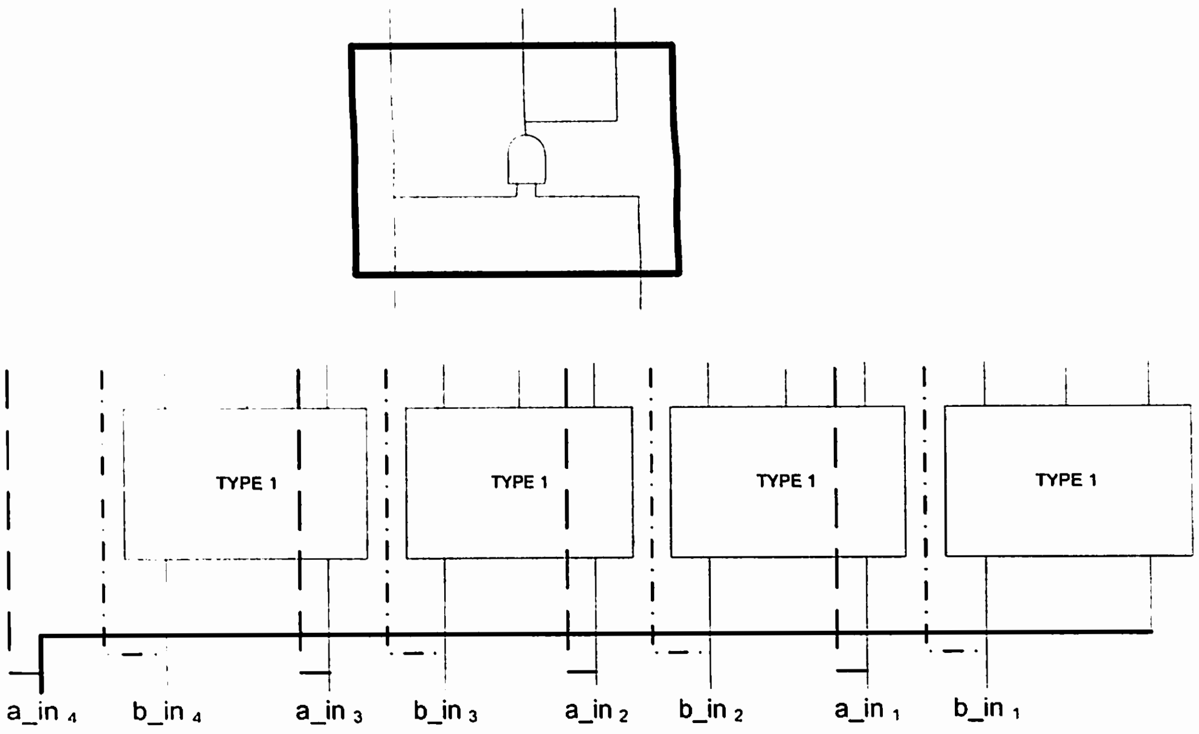


Figure 2.10: Type 1 node structure

2.3.2 Type 2 Node

Type 2 nodes are the most common nodes in the encoder circuit. The structure of the type 2 nodes is shown in figure 2.11. The input to the type 2 nodes comes from either the output from the type 1 nodes or from the outputs of the other type 2 nodes. The output of the type 2 nodes is fed to the input of the type2 nodes in the higher level or to the input of the type 3 nodes. Each type 2 node has three gates, one AND gate and two OR gates. The largest delay experienced by any signal passing through any type 2 nodes is equivalent to two gate delays.

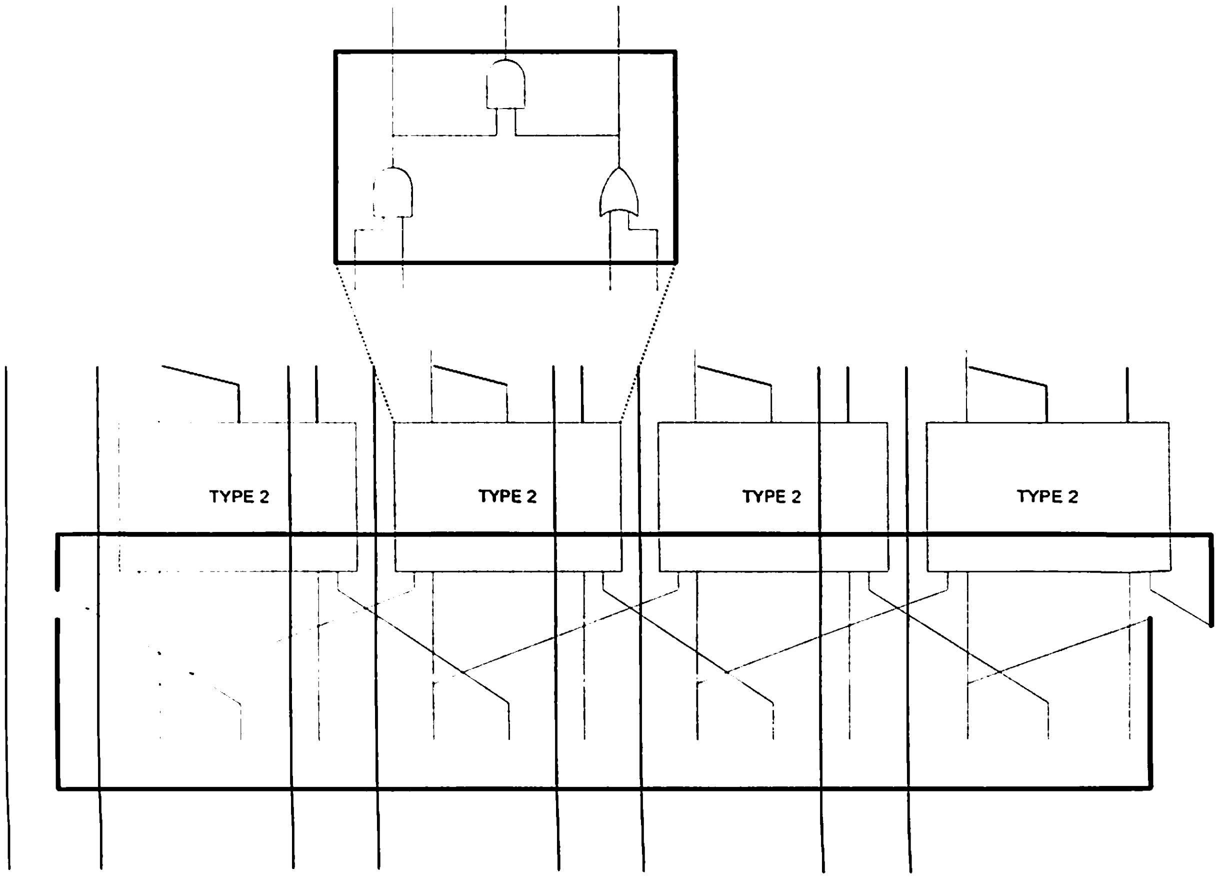


Figure 2.11: Type 2 node structure

2.3.3 Type 3 Node

Type 3 nodes are used in the last stage of the encoder. Thus the input to the type 3 nodes always comes from the output of the type 2 nodes. The structure of the type 3 nodes is shown in figure 2.12. It consists of four gates, one OR gate and two AND gates and one inverter.

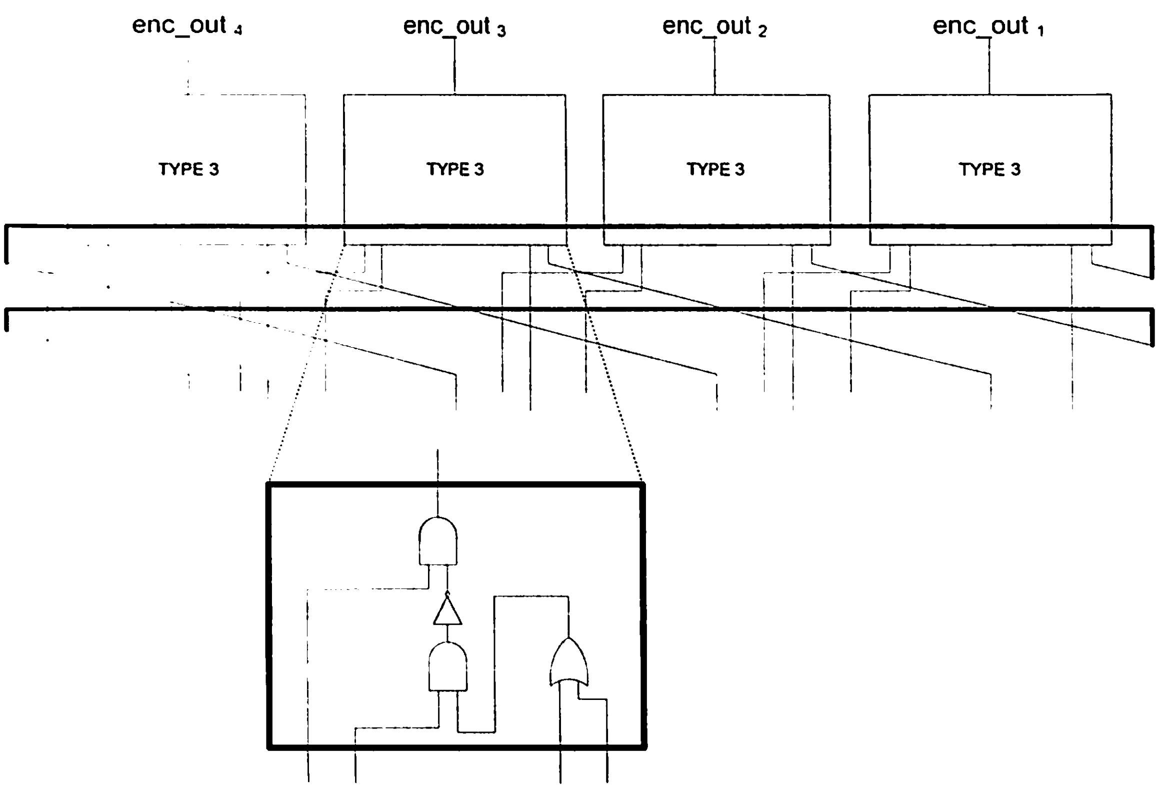


Figure 2.12: Type 3 node structure

2.3.4 Circuit Operation

First it is verified that the circuit given in the figure 2.13 indeed provides the kill signal and the enc_out signal as given by equation 1 presented in section 2.2. For this purpose the kill 3 signal is evaluated from figure 2.13.

$C = A + B$, where, $A = b\_in_3 \,.\, a\_in_2$ and $B = b\_in_2 \,.\, a\_in_1$

$F = G + H$, where $G = b\_in_4 \,.\, a\_in_3$ and $H = b\_in_1 \,.\, a\_in_4$

$E = b\_in_2 \,.\, b\_in_1$

$D = E \,.\, F$

$K = D + C$

$kill_3 = K \,.\, b\_in_3$

$= (D + C) \,.\, b\_in_3 = (E \,.\, F + A + B) \,.\, b\_in_3$

$= (b\_in_2 .\, b\_in_1(G + H) + b\_in_3 \,.\, a\_in_2 + b\_in_2 \,.\, a\_in_1) \,.\, b\_in_3$

$= (b\_in_2 \,.\, b\_in_1 \,(b\_in_4 \,.\, a\_in_3 + b\_in_1 \,.\, a\_in_4) + b\_in_3 .\, a\_in_2 + b\_in_2 .\, a\_in_1)$
$.\, b\_in_3$

$= (a\_in_3 .\, b\_in_4 \,.\, b\_in_2 \,.\, b\_in_1 + a\_in_4 \,.\, b\_in_2 \,.\, b\_in_1 + a\_in_2 \,.\, b\_in_3 + a\_in_1$
$.\, b\_in_2) \,.\, b\_in_3$

$= a\_in_2 .\, b\_in_3 + a\_in_1 \,.\, b\_in_3 \,.\, b\_in_2 + a\_in_4 \,.\, b\_in_3 .\, b\_in_2 \,.\, b\_in_1 + a\_in_3 .$
$b\_in_3 \,.\, b\_in_2 .\, b\_in_1 .\, b\_in_4$

This expression is same as given by the general equation (1) of section 2.2. The enc_out signal is generated from the kill signal as seen from figure 2.13 as

$enc\_out_3 = a\_in_3 \,.\, \sim kill_3$

If all but one of the b_in signals is set at logic high then the LATPE will work exactly as the KSG. All the b_in signals are fed to the AND plane, which consists of two input AND gates interconnected together. Therefore whenever one of the input is at logic one, the output is same as the other input. The whole AND plane will function as a simple wire network if all the b_in signals are set at logic high. Thus in such a case the circuit will functionally look as shown in figure 2.7(a). This circuit is similar to the KSG shown in figure 2.5, except for the wrap around network. Thus if all the b_in signals are set to logic one, a loop circuit is formed. If any one of the inputs to the AND plane is set at logic zero, with all the rest of them at logic one, 0's are formed on all of the inputs to the vertical column in the OR plane. This effectively breaks the loop created in the circuit shown in figure 2.7 (a). Thus a zero at a particular b_in input will effectively act as a shield for the signals being fed to the OR plane from the right side of that particular input. This effectively opens the closed loop form of the circuit and the equivalent circuit acts exactly as the SLPE.

As an example a typical situation is shown in figure 2.13. Here the input b_in$_3$ is set at logic low and all the rest of the b_in inputs are set at logic high. The AND gates that are turned permanently off by this particular input combination are shown in the figure as grayed. The crossover network flowing towards the left, which carry low signals are shown as grayed in the figure. Thus the input b_in$_3$ effectively has separated the circuit into two parts. The right part of the circuit with the inputs a_in$_4$ and a_in$_3$, and the left part with the inputs a_in$_2$ and a_in$_1$. Thus any input combination of the inputs a_in$_2$ and a_in$_1$ will not show in the outputs enc_out$_4$ and enc_out$_3$. While if any of the inputs a_in$_4$ or a_in$_3$ are at logic high level, then this shows up in the output enc_out$_2$ and enc_out$_1$ through the wrap-around networks. If the right and the left part are taken individually without the wrap-around network, they can be considered as two independent SLPE's. The inputs a_in$_3$ and a_in$_4$ forms the first SLPE which can be named as SLPE1. Since for the SLPE the rightmost input will have the highest priority, the input a_in$_3$ will be the highest priority input in the SLPE1. The inputs a_in$_2$ and a_in$_1$ form the second SLPE named SLPE2. Here since a_in$_1$ is the rightmost input, this input will have the highest priority. The wrap around network serves as the connection between SLPE2 and SLPE1. Since the a_in inputs are propagated from right to left in the circuit because of the orientation of the crossover network in the OR plane, SLPE1 will have higher priority than SLPE2. Thus a combined SLPE can be assumed with SLPE1 on the right side of SLPE2. This representational diagram is shown in figure 2.14. Here the wrap around circuit can be assumed to supply a signal, kill_in, to SLPE2. The kill_in signal is generated by SLPE1, if none of the inputs in the SLPE1 are at logic high state, then the kill_in signal will have no effect on the operation of SLPE2. While if any one of the inputs in SLPE1 is at logic high then the kill_in signal will be such that the output of the SLPE1 is always at logic low independent of the inputs a_in$_1$ and a_in$_2$. Thus the entire LATPE can be represented as a simple SLPE with the input order rearranged as shown in step 4 of figure 2.14. Thus the new priority order from highest to lowest priority is a_in$_3$, a_in$_4$, a_in$_1$ and a_in$_2$. In this way for each of the new combination for the b_in input, the circuit can be represented by a simple SLPE but with a different order of input signals a_in. The dark band in the step 1 of figure 2.14 represents the permanently turned off AND gates shown in figure 2.13. Thus the closed loop can be broken at that point as shown in step 2. Step 3 shows how the two separated parts can be rearranged so that they appear as the cascaded combination of the two simple linear encoders.

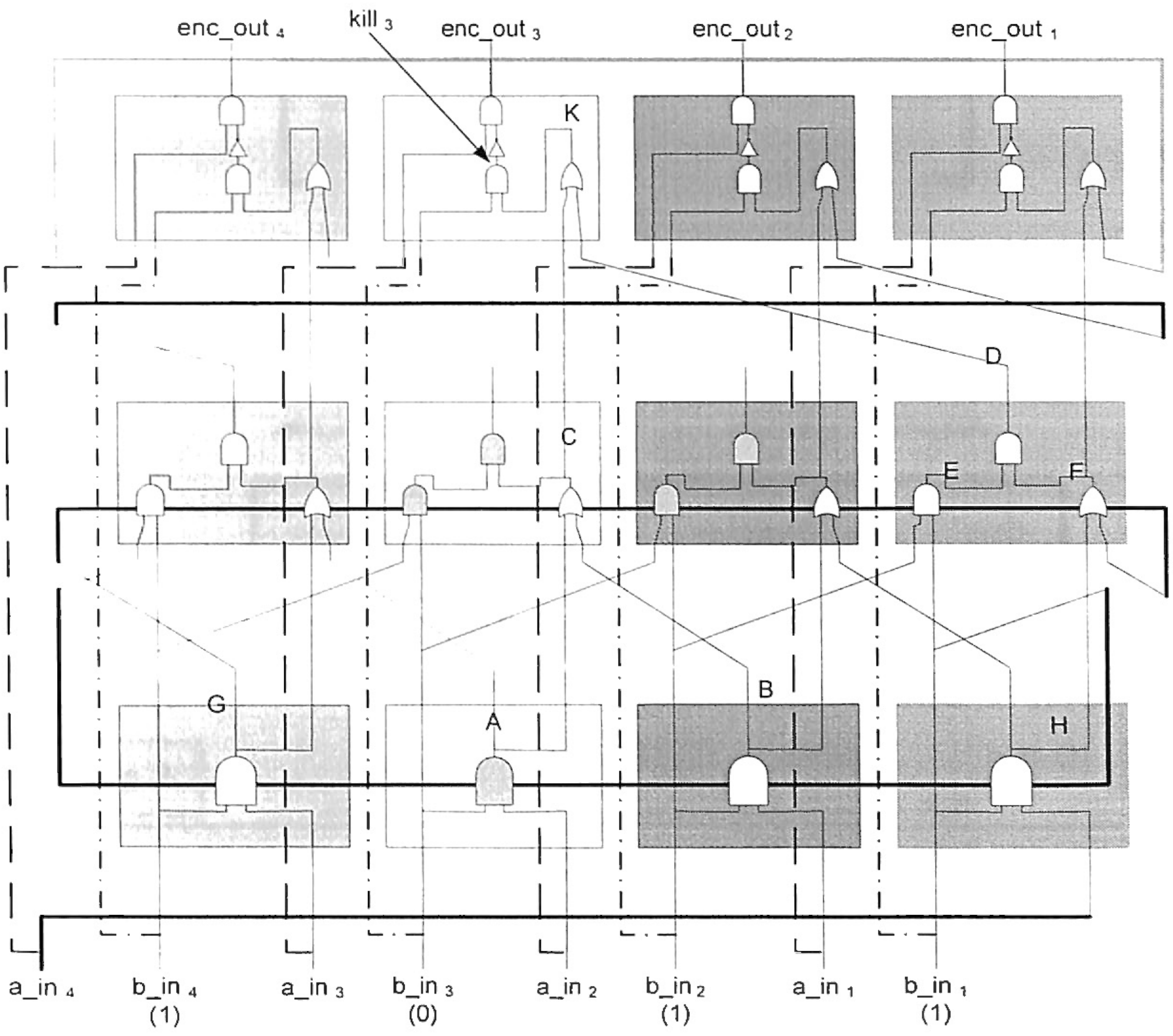


Figure 2.13: Operation for the 4-input LATPE when b_in3 = 0

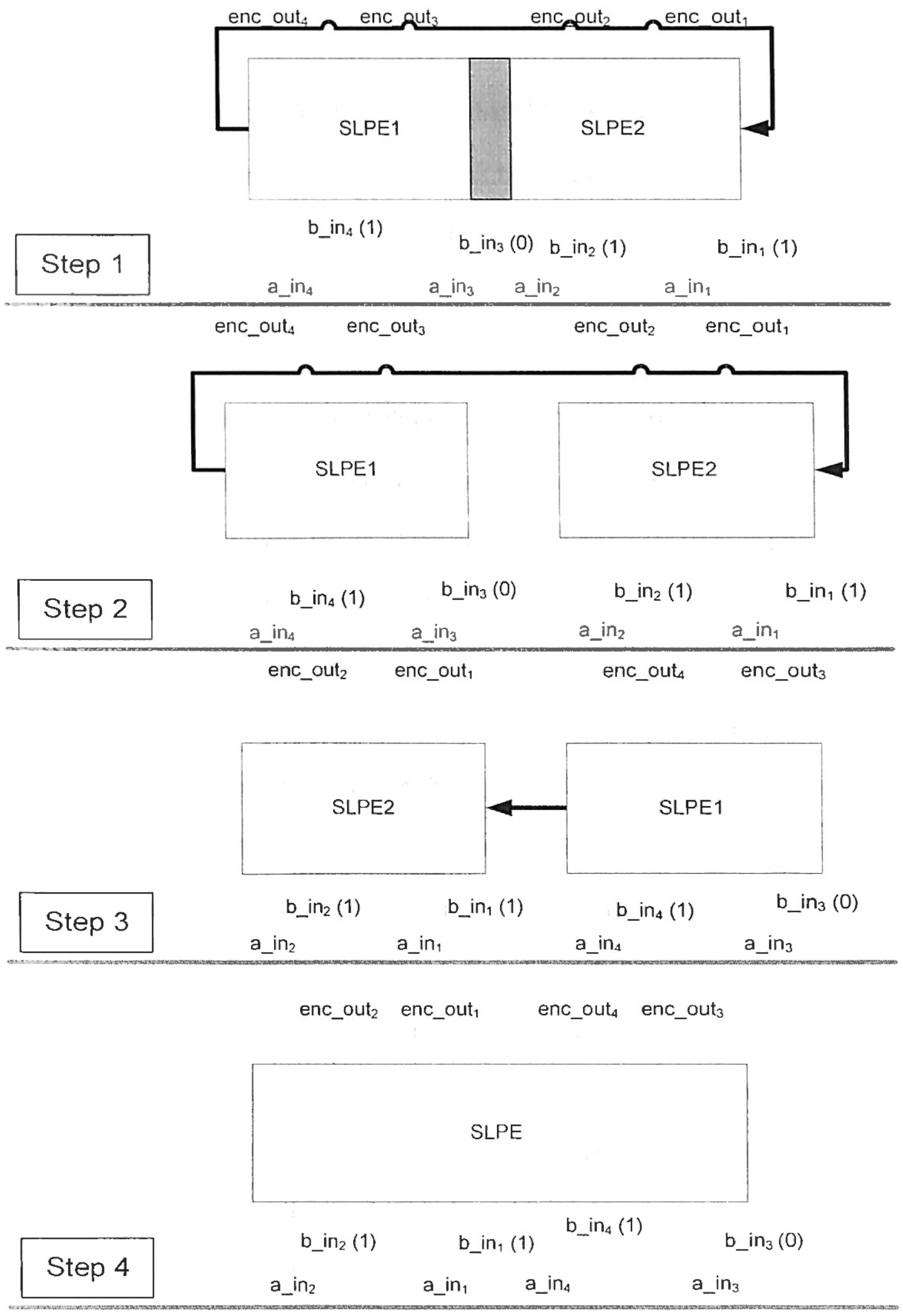


Figure 2.14: Representational steps for a 4 bit LATPE

2.4 Modular Priority Encoder (MPE)

From the analysis of LATPE it is clear that although the circuit offers an O(log N) delay, the practical implementation of the circuit becomes very difficult with the increase in the number of input signals. The overall design of the 128 bit LATPE shown in Appendix B shows that a large area is taken by the interconnect networks alone. This is because of the increase in the size of the cross over networks in the higher levels of the circuit, which dominates the delay. Although a buffer circuit as shown in figure 2.9 can be used to minimize the delay, the use of such a buffer greatly increases the area requirement of the circuit. The layout of the circuit also becomes very challenging, as the buffer circuit has to be redesigned for each new stage. Thus the amount of circuitry that can be reused in the layout process is very minimal. Another major delay contribution is caused by the wrap around network. These wrap around network along with the cross over network take up most of the area in the higher stages of the LATPE (see the layout of the 7th stage in Appendix B). While a SLPE is similar to the LATPE, the fact that there is no wrap around circuit makes the SLPE circuit much faster than the LATPE. Moreover it is shown in section 2.3.2 that a LATPE can be represented as a cascade combination of two SLPE's, with one of the SLPE providing kill_in signal to another. Thus if the same principle can be physically implemented, then a round robin encoder can be implemented by a simple combination of smaller SLPE's. Similar hierarchical methods have been used to design linear encoders in the past [14] [15]. A 32×32 switch has been implemented in reference [16]. Thus a method to break the total encoder circuit into smaller units can be found such that the units are independent of each other. Each of these smaller units can be a SLPE of smaller order such that the crossover networks are not too large to require a buffer circuit. Appendix B shows the layout for the 3rd stage of the 128 bit LATPE where the interconnect networks are in similar proportion to the circuit elements. This significantly reduces the circuit area and delay. Each of these units is controlled by another circuit which should be in a round robin configuration. The purpose of this circuit is to provide appropriate kill signals to the individual units. Since all the SLPE units are identical in construction, only a single unit is needed to be implemented which acts as a template. The other units are just instantiated from the constructed template. This greatly reduces the layout complexity of the circuit. By using such a method the LATPE circuit can be subdivided into several modules and a detailed analysis is presented in the next section.

Chapter 3

Modular Approach of Design

3.0 Introduction

As mentioned in the section 2.4 a modular approach of design should have several smaller independent units which are controlled by means of a common circuit which supplies them with kill_in signals. A block diagram representation of the circuit is shown in figure 3.1.

As shown in figure 3.1, the overall circuit can be divided into three stages. Stage 1 is the initial stage of the MPE. The inputs to this stage are the a_in and b_in inputs to the encoder circuit. This stage provides signals to the second stage. The structure for the second stage is similar to a LATPE circuit. This second stage has a much smaller number of inputs compared to the encoder circuit itself. This reduction in the number of inputs helps to keep the second stage much simpler to implement. The output of the second stage provides kill_in signals to the third stage. The third stage consists of several independent units. These units are similar to the simple linear priority encoder. The number of inputs to these units is much smaller than the number of inputs to the encoder circuit. Thus, the crossover network in these units is not large enough to require a buffer stage. This greatly simplifies the design of the circuit and also saves considerable chip area. Also only one of the units needs to be designed, since all of the units in stage 3 are similar in structure.

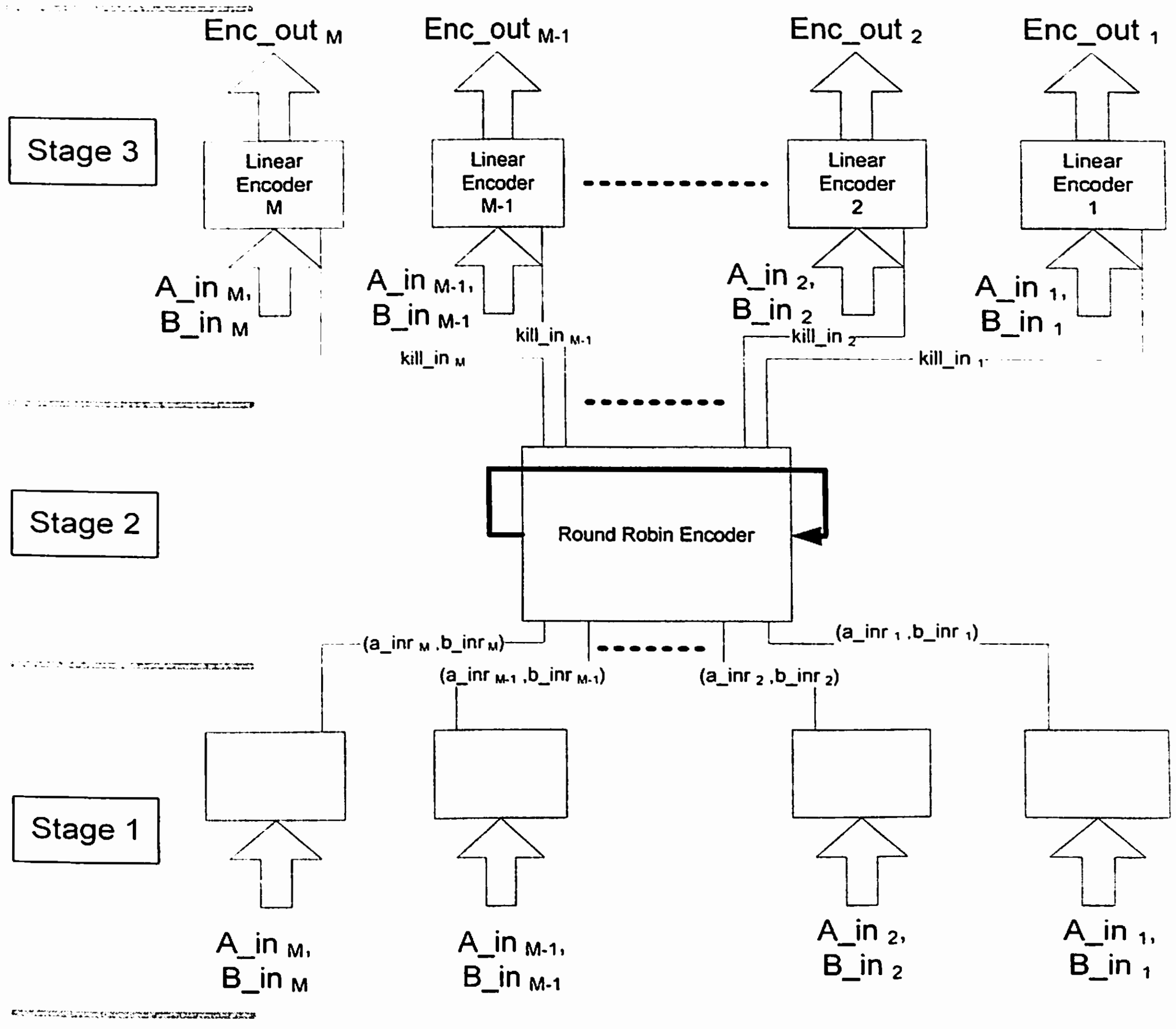


Figure 3.1: Structure for the modular approach of Encoder design

3.1 Implementation Detail

For the design of an N input priority encoder, the inputs can be subdivided into smaller groups. In figure 3.1, the N input signals have been grouped into M groups. For ease of analysis as well as to make maximum reuse of the circuit, these groups can have an equal number of inputs. If the N inputs are divided into M equal groups then each of the groups consists of L inputs, where N = L × M. If L happens to be a fractional number while dividing N by M, then either the number of groups or the number of inputs can be varied such that all the groups have an equal number of inputs except one of them which will have the remaining number of inputs. The choice of the number of groups, M, and the number of inputs within each group L must be done judiciously. The number of groups, M, determines the size of the second stage, while the number of inputs within each stage gives the size of the units used in stage 1 and stage 3. From the layout of the circuit, the number of inputs for which the circuit does not require extra buffer stages for the crossover networks was found. The result is given in table 3.1.

	Round Robin **(Stage 2)**	**Linear** **(Stage 1 & 3)**
Maximum no of inputs	8	16

Table 3.1: The maximum number of inputs without the use of buffers

Thus if the above scheme is used, then the encoder with 128 inputs can be constructed without using any buffer stages for the interconnect network. Also only two basic blocks need to be designed. One of them is a modified 8-input LATPE circuit used in stage 2, and the other is the modified SLPE circuits used in stage 1 and stage 3. Thus the modified SLPE unit is used 8 times in stage 1 and 8 times in stage 3. Since the modified LATPE has only 8 and the modified SLPE has only 16 inputs, the circuit area is not dominated by the wrap around or the cross over network. This results in substantial saving of area as compared to implementing the whole 128 bit encoder circuit using the LTAPE approach.

3.1.1 Stage 1

The input signals are bundled into M groups. If all the groups are assumed to have L inputs, then each unit in stage 1 will have L of the a_in inputs and L of the b_in inputs. These signals are bundled into combined signal (A_in $_k$, B_in $_k$) in figure 3.1, where k denotes the group number for the signals. Each unit in stage 1 provides only two single bit outputs to stage 2. These two outputs are a_inr $_k$ and b_inr $_k$, where k denotes the group number. If any of the L b_in inputs to a particular k-th stage 1 unit is at logic zero, then the output b_inr $_k$ must be zero, otherwise it is at logic one. This means that the b_inr signal is used to indicate to stage 2 in which of the M groups of input signals does the highest priority input fall. The a_inr $_k$ signal indicates whether any one of the priority inputs in any k-th group is at logic high. Thus if all of the L a_in inputs for any particular k-th group is at logic zero, a_inr $_k$ will be at logic zero. For the case when one or more of the L a_in inputs are at logic one, the output a_inr $_k$ depends upon the value of the L b_in inputs. Two different cases for the L b_in inputs can be identified, the first is the case when all the L b_in inputs are at logic one. That means none of the L inputs in that group is the highest priority input. In this case if any one of the L a_in signals is at logic 1, then the output a_inr $_k$ will be at logic 1. If on the other hand any of the L b_in signals within that group is at logic zero, indicating that it is the highest priority input, then the value of the output a_inr $_k$ will depend upon the relative position of the a_in signals that are at logic one with respect to the highest priority signal b_in. If any of the a_in inputs that are at logic one are to the left of the highest priority input then in that case the output a_inr $_k$ will be at logic one. If on the other hand none of the a_in inputs to the left of the highest priority input are at logic one, then the output a_inr $_k$ is at logic zero.

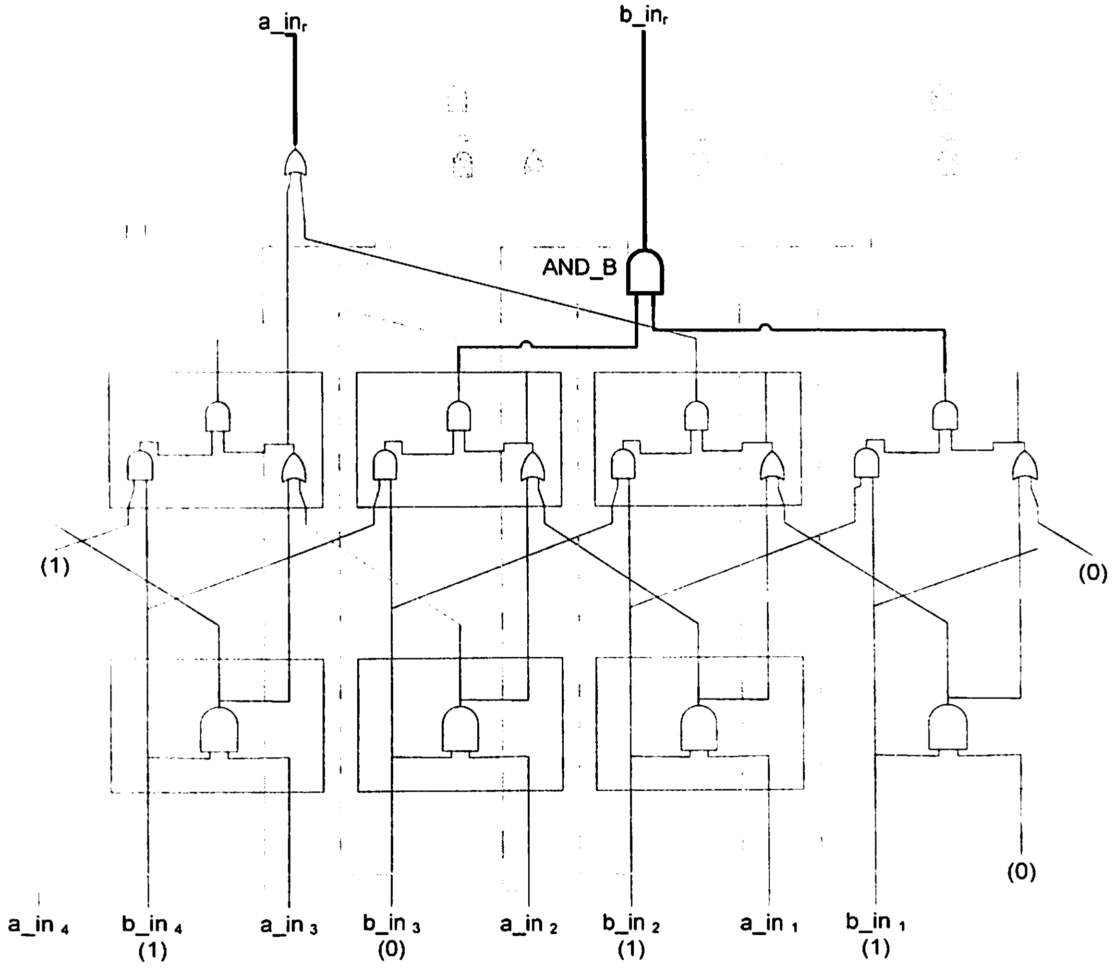


Figure 3.2: Structure of unit cell in stage 1 with four inputs

Thus based one the above explanation of the operation of the basic units of stage 1, it is obvious that the structure for the basic unit of the stage 1 must look like the SLPE. The only difference is, for an L input SLPE, there are L outputs, where as for the L input basic unit of stage 1, there are only two outputs. The structure for the 4-input basic unit block for stage 1 is shown in figure 3.2.

The circuit elements that are presented in figure 3.2 in a grey color represent the elements that are not needed in stage 1. They are needed for stage 3 which will be discussed later.

3.1.2 Stage 2

Stage 2 takes the inputs from the stage 1 and then provides the output signal to stage 3. Stage 2 does not have any unit elements. The structure for the stage 2 is very similar to the structure of the LATPE. The exact structure of stage 2 is shown in figure 3.3. The dark connection in figure 3.1 going from the left to the right of stage 2 represents the wrap around network. This is the only stage where the wrap around network is needed. The number of the wrap around signals depends on the number of inputs to stage 2. Since the number of the inputs to the second stage is the number of unit elements in stage 1. The stage 2 doesn't require a large number of inputs even for the case when the total numbers of inputs to the encoder circuit are high. For example, for a 128 bit input encoder if each unit of stage 1 has 16 inputs, then the second stage requires only 8 inputs. This reduction keeps the wrap around network as well as the cross over network small enough such that no extra buffer stages are required. Since the wrap around networks is small, the circuit also functions much faster.

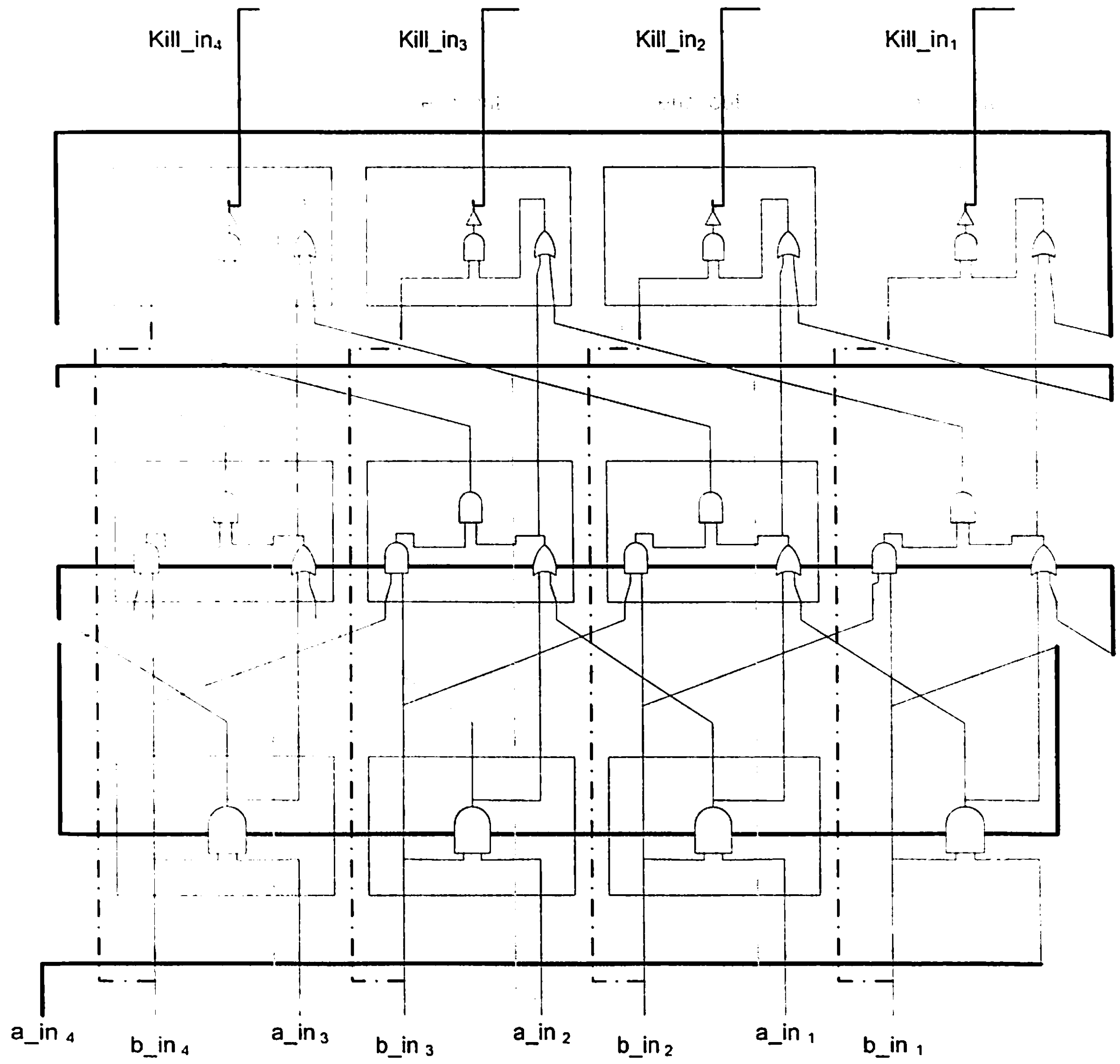


Figure 3.3: Structure for a four input unit in stage 2

Stage 2 gets inputs from stage 1 which indicates which of the M groups of stage 1 has the highest priority input and also which of the inputs in the M groups of L inputs are at logic high. The operation of the second stage is exactly the same as that of the LATPE except for the fact that it generates the kill_in signal rather than the enc_out signals of the LATPE circuit. Generating the kill_in signal requires one less stage than to generate the enc_out output. In figure 3.3 the grayed elements represent the components that need to be removed from the LATPE circuit. The kill_in signal always starts at logic high from the first highest input at logic high all the way to the lowest level input signal in the round robin fashion. Thus at any instance of time, there can be more than one kill_in signal at logic high. Stage 2 provides the kill_in signals to the stage 3.

3.1.3 Stage 3

Stage 3 has the same number of unit as stage 1. Each units of stage 3 has a kill_in input from the stage 2 as well as L a_in and the b_in inputs. The output of the units of stage 3 will produce the output enc_out. The structure for this unit is shown in figure 3.4. Figure 3.4 shows that the circuit is very similar to the unit elements of stage 1 shown in figure 3.2. The grayed circuit elements in figure 3.2 represent the additional circuit elements that are present in the unit elements of stage 3. The leftmost input in figure 3.2 is permanently connected to logic low, whereas in figure 3.4 the same input is connected to the kill_in signal provided by stage 2. This kill_in signal if at logic low indicates that the priority token has not yet been used in any other higher priority group. Thus the unit behaves like a SLPE. On the other hand if the kill_in signal is at logic high then it indicates that the priority token has already been used in the higher priority blocks. At this time all the stage 3 units which do not have the highest priority inputs can be lumped into one category, and the single unit block which has the highest priority input can be placed into a second category. If the kill_in signals to the first category units are high, then all of the enc_out signals of those blocks must be at logic low, irrespective of the input bit combinations. On the other hand if the kill_in signal is high to the second category unit with L inputs where the highest priority input is at the k-th position from the right, then none of the rightmost enc_out outputs up to the k-1-th position will be at 1. Of the remaining L-k+1 leftmost enc_out signals, one and only one can be at logic high, if in fact at least one of the leftmost L-k+1 input are at logic high. Since each stage 3 units can be programmed to have any one of its input as the highest priority input and the operation of the circuit is linear without the wraparound networks, this circuit can be called a programmable linear priority encoder (PLPE).

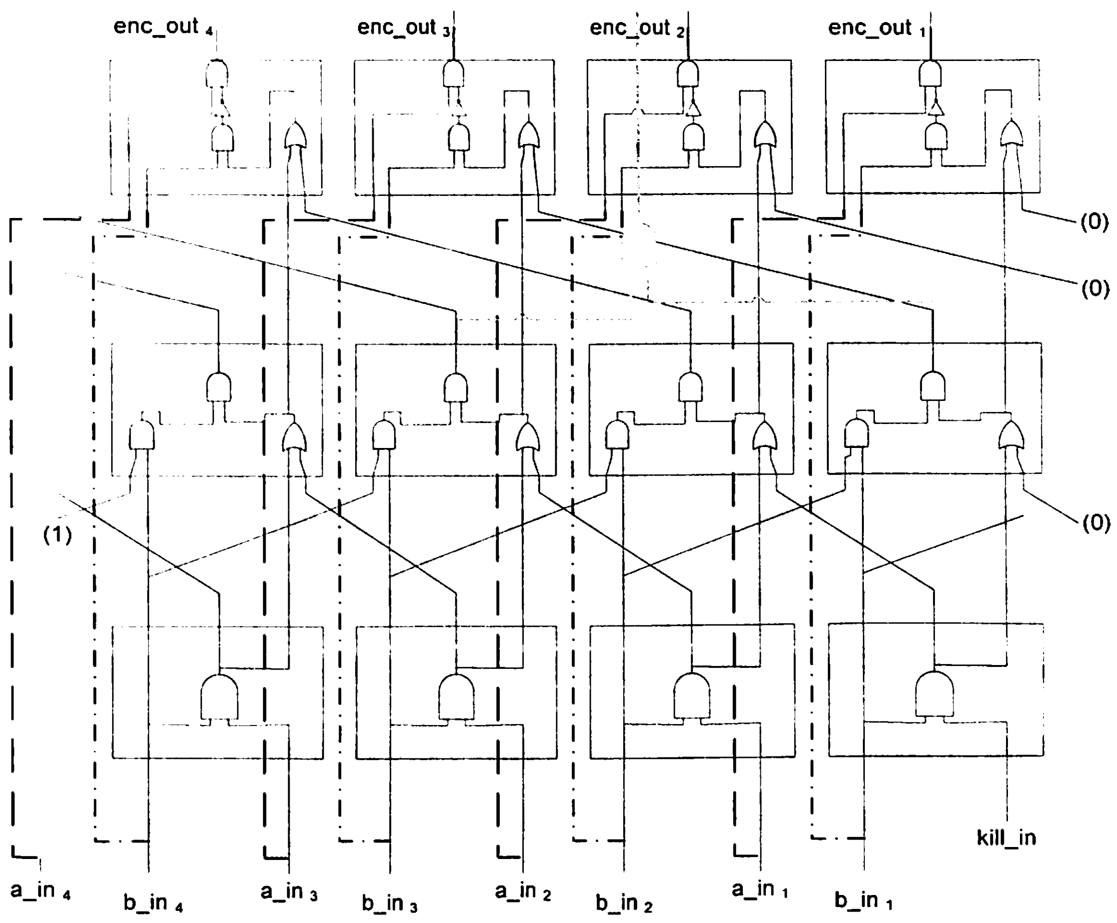


Figure 3.4: Unit cell of Stage 3 with four inputs

The grayed circuit elements in figure 3.4 show the extra elements as well as the network that are necessary to form the unit elements of stage 1. As explained earlier in the circuit operation of the LATPE circuit, the begin signal breaks the PLPE into two independent SLPEs. But in this case as the wrap around network is not present, the two separated encoders will act independently. Thus if the kill_in signal is connected to logic low, then it is possible to have more than one enc_out signal high at the same time, depending on the input bit combinations, if at least two high inputs fall in the two separated SLPEs. Thus the kill_in signal prevents this situation from happening.

3.2 Improved Modular Priority Encoder (IMPE)

From the operation of the MPE it is clear that the three stages; stage 1, stage 2 and the stage 3, work in serial fashion. That means first the signals are fed to the stage 1 which provides the output signals to stage 2 which then generates the kill_in signals to the stage three which will finally generate the encoder output signals. Thus while the signal is being processed in stage 1, stage 2 and stage 3 are sitting idle. And while the signal is being processed in stage 3, stage 1 is sitting idle. Also if the structure of the basic units in stage 1 and stage 3 are compared they show very similar structure. Figure 3.4 shows the structure for the basic unit of stage 3 which only requires one extra AND gate to tap the signal b_inr from it. The a_inr signal can be tapped directly from the leftmost column of the basic unit of stage 3, as shown in figure 3.4. Since stage 1 and stage 3 are never used at the same time and since the structure for each of them are very similar we can combine stage 1 and stage 3 together into a single stage. This combination affectively cuts the circuit area by half without any additional disadvantage as compared to the MPE. The area reduction is mainly due to the fact that the main contributors to the area are stage 1 and stage 3, each of which have M units, whereas stage 2 has only one unit with comparable size to the unit element of stage 1 or stage 3. Figure 3.5 shows the block diagram for the design.

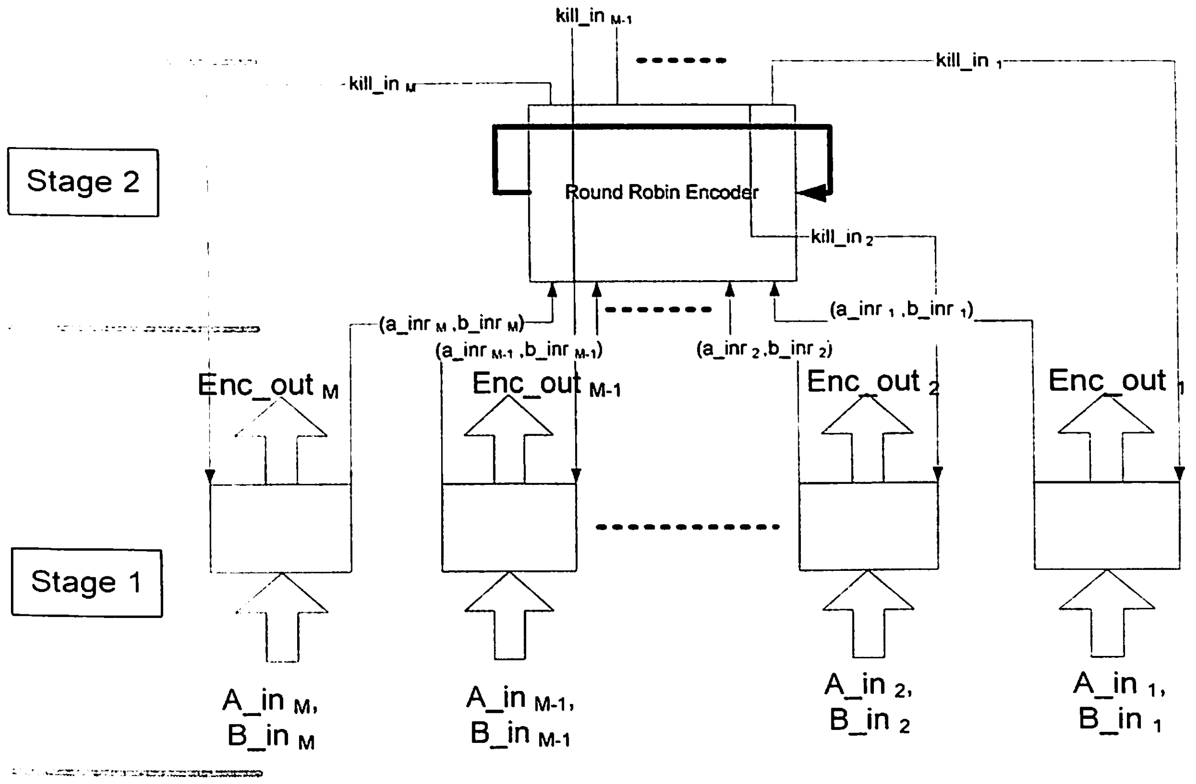


Figure 3.5: Block diagram for the improved modular priority encoder (IMPE)

3.2.1 Operation of IMPE

It is seen from figure 3.5 that the IMPE has only two physical stages. The total delay is still the same three-stage delay as that of the MPE shown in figure 3.1 The reason for this is that the IMPE approach reuses stage 1 to perform the function of stage 3 of the MPE.

The structure for the basic unit for stage 1, shown in figure 3.6, is similar to the basic unit of the stage 3 for the MPE with the additional and gate and the tapping required for stage 1. Thus the circuit operation is also very similar to that of the MPE.

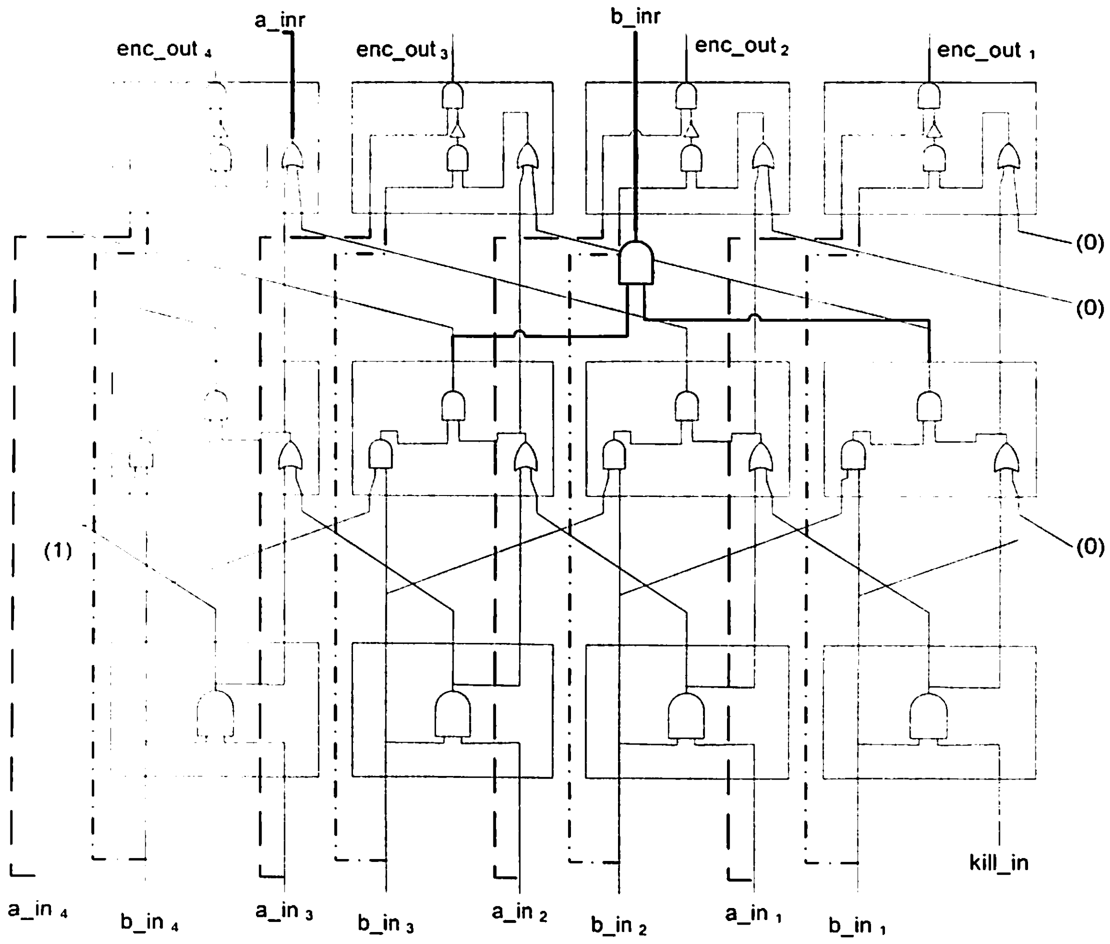


Figure 3.6: Basic Unit for stage 1 with four inputs used in IMPE

The input signals a_in and b_in are bundled into M groups, each of which can be of equal or unequal sizes. The choice is governed by the complexity that is bearable in the unit elements. Stage 1 decides which of the M group of elements have the highest priority input as well as which of the input groups have made the request to the encoder. This information is fed to stage 2 through the inputs a_inr and b_inr. At this time the output enc_out of the first stage are not the valid outputs. The structure for stage 2 is exactly the same as that for stage 2 of the MPE as shown in figure 3.3. Thus the stage 2 for the IMPE operates the same as stage 2 of the MPE and provides the kill_in signals. Now these kill_in signals are fed back to the stage 1. The effects of the kill_in signals in the IMPE are exactly the same as the effect of the kill_in signals in the third stage of the MPE. When stage 1 provides the a_inr and the b_inr signals to stage 2, more than one of the enc_out signals in the M input groups can be at active 1, because at this time each of the units of the stage 1 acts as a PLPE with kill_in as zero. When the kill in signals are provided by stage 2, only the enc_out with the highest priority will remain at logic high. All of the other enc_out will be forced to logic low. Since the signals travel through stage 1 twice and once through the stage 2, the delay through the network is exactly the same as the delay through the MPE. The great advantage of this approach is the reduction in the chip area to half that of the MPE.

Chapter 4

Implementation of the Designs

4.0 Introduction

This chapter gives the exact design implementation for each of the different encoder circuits that have been used in this thesis. The encoder circuits that have been implemented are the BSPE, LATPE, MPE and the IMPE. All of the designs were implemented using the Verilog hardware description language (HDL). Since the main objective of the design is to achieve the minimum delay that is possible, strict control of the structure of the circuits is necessary. Thus all of the circuits have been implemented using structural Verilog code. All of the encoders that have been implemented have 128 a_in inputs and 128 b_in inputs. Because of the large number of inputs, the amount of HDL that has to be written is very time consuming if the whole circuit is implemented using structural Verilog code. Since all of the circuits that have been implemented have a very regular form, software programs that are written in high level language can be used to generate the structural Verilog code. Thus for each of the implementations, software programs in Visual Basic (VB) were written to generate the structural Verilog code. Apart from the Verilog implementation, the LATPE and the IMPE were also implemented using the Cadence Virtuoso layout editor. Finally the IMPE was incorporated into a chip with a 40 pin dip pad driver for the fabrication of the circuit.

4.1 Implementation of BSPE

Implementation of the BSPE was done using the structural Verilog code only. The size of the encoder was 128 bits. In order to facilitate fast coding of the circuit, a program was written in Visual Basic to generate the Verilog code. Each of the nodes in the barrel shifter circuit was implemented using the built in multiplexer MX21 of the AMI3HS 0.35μ CMOS standard cell library and the OR gates for the KSG were implemented using the standard cell OR21 from the same cell library. The cell description and the delay for the components used from the cell library are presented in Appendix A.

4.2 Implementation of LATPE

Two different implementation of the 128-bit LATPE circuit were done. The first one was using the Verilog HDL and the second one using the Cadence Virtuoso Layout editor. The detail description for each of the implementation is presented below.

4.2.1 Verilog Implementation

The entire circuit as described in section 2.4 of this thesis was implemented in structural Verilog code. A software program written in Visual Basic was used to generate the structural Verilog code. The AMI3HS 0.35μ CMOS standard cell library was used to implement the individual components. The AND plane was implemented using the AA21 cell from the cell library and the OR plane was implemented with the OR21 cell of the cell library. The cell description and the delay for each of these are presented in Appendix A.

4.2.2 Layout of LATPE

The circuit layout was created using the Cadence Virtuoso Layout editor. No cell library was used for this purpose. The process used was the AMI 0.6 μ C5N from the American Microsystems Inc. This process has three metal layers and two poly layers. The second poly layer was never used in the circuit, since this poly layer is used for the fabrication of poly to poly capacitors. The basic implementation of the circuit is exactly as shown in figure 2.8. All the circuits are implemented as static CMOS circuits. The circuits are easy to implement if they are in NAND–NOR form rather than in AND-OR form. Thus the type1, type2 and the type3 networks as shown in the figure 2.10, 2.11 and 2.12 respectively are changed to the NAND-NOR form and are shown in figure 4.1,4.2 and 4.3 respectively.

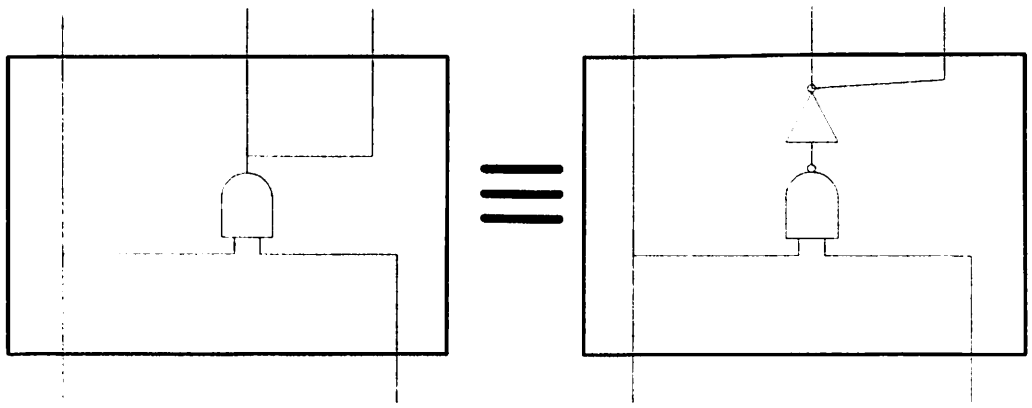


Figure 4.1: Modified Type 1 Node for LATPE

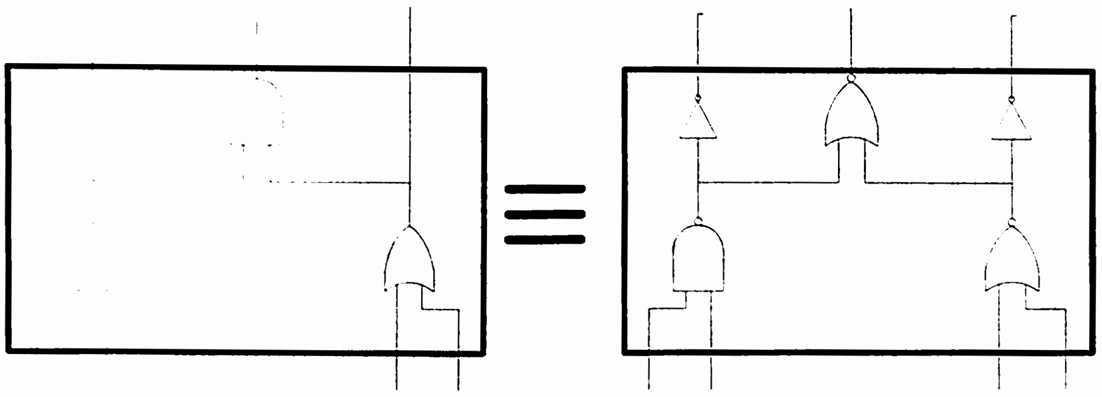


Figure 4.2: Modified type 2 node for LATPE

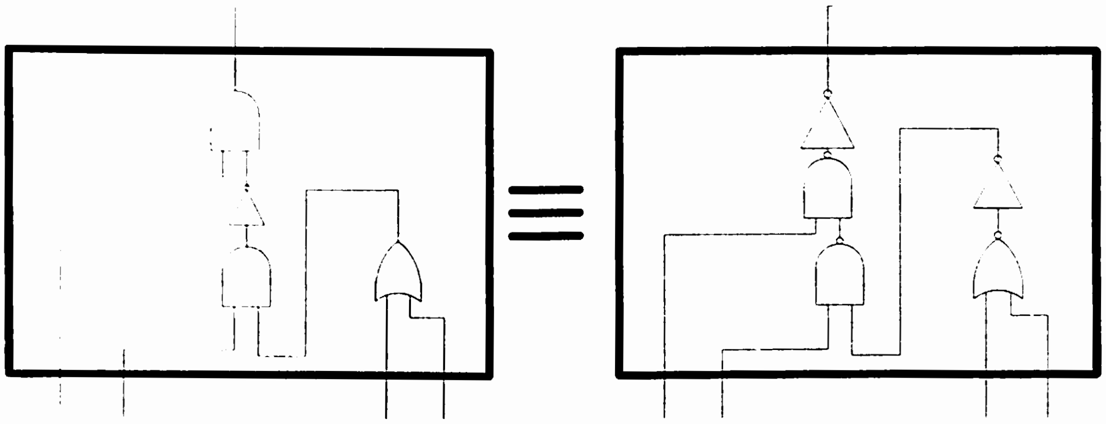


Figure 4.3: Modified type 3 node for LATPE

The total components that are used in these modified three types of nodes are the 2-input NAND, 2-input NOR and the inverter circuit. Each of these elements is implemented using static complementary MOS design as shown in figure 4.4 through 4.6. Both P and N type of MOSFET are used. All of the transistors that have been used are of minimum sized geometry.

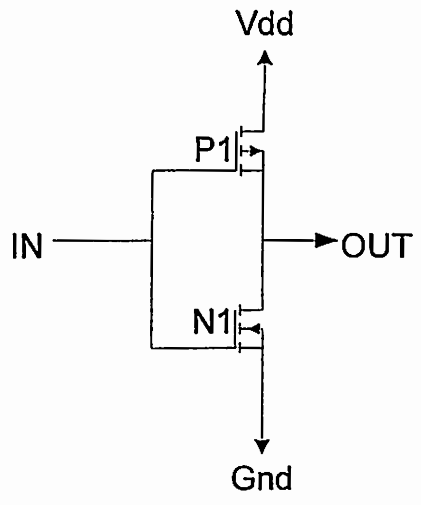


Figure 4.4: Static CMOS implementation of the inverter

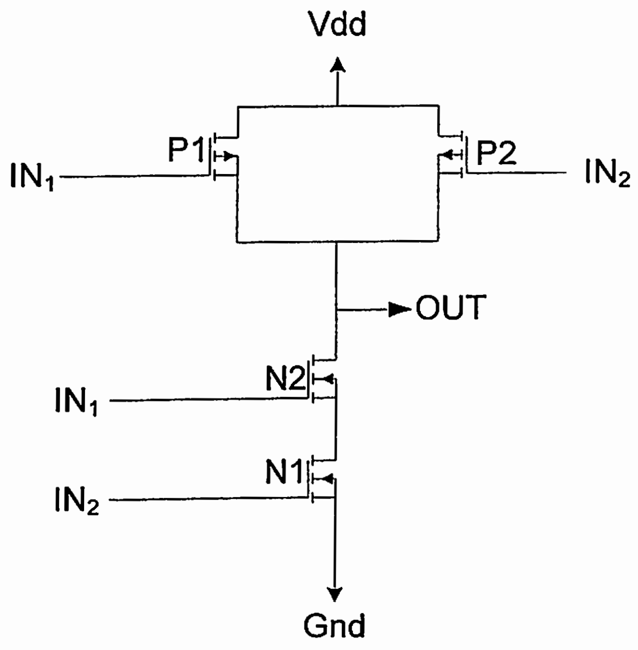


Figure 4.5: Static CMOS implementation of the 2-input NAND gate

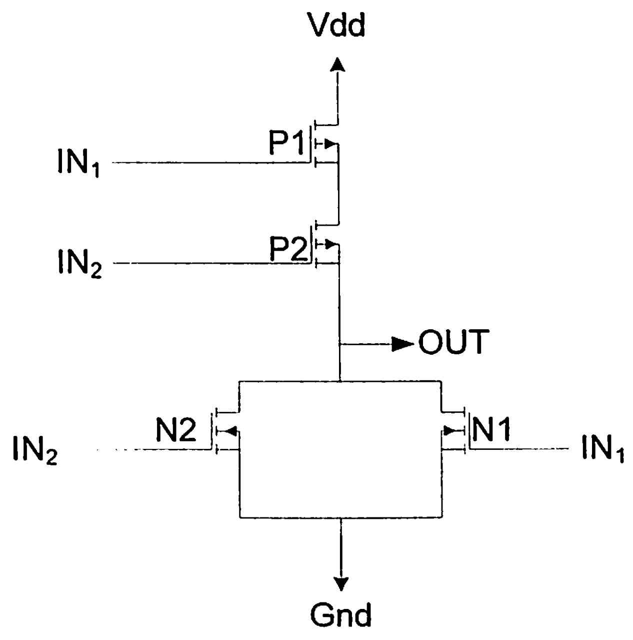


Figure 4.6: Static CMOS implementation of the 2-input NOR gate

The entire wrap around network is implemented in Metal 3, so that the delay offered by them is minimum. The cross over network is implemented in metal 1 and metal 2. All the transistors used in the circuit are of minimum geometry, with length 0.6μ and width of 12μ, no sizing up of the transistors was done. The layout of each element was done in such a way that the number of contacts on the output node was minimized, which helps to make the circuit faster. Also doglegged gates were used wherever possible to save the circuit area. The N-type of transistor was laid out directly on the P-type substrate, whereas the P-type transistor was laid out on the N well. Substrate contacts were placed at a number of places to prevent the latch-up problem.

4.3 Implementation of MPE

It is clear from the block diagram for the MPE that the hardware used in stage 1 is very similar to the hardware used in the stage 3. Thus there was no need to implement the circuit in the layout form, thus only Verilog version of the circuit was coded. This version allowed one to compare the relative speed with its IMPE counterpart. The entire circuit was coded in structural Verilog using software developed in Visual Basic. The 128-bit encoder was implemented in purely structural Verilog as shown in the block diagram of figure 3.1. Each of the elements of stage 1 was implemented as shown in figure 3.2. The implementation of stage 2 and the unit element for stage 3 was done as shown in figure 3.3 and figure 3.4 respectively.

4.4 Implementation of IMPE

The IMPE circuit was implemented using both Verilog HDL and the Cadence layout editor. The size of the encoder implemented was 128-bits. The following section provides the description for each of the implementations.

4.4.1 Verilog Implementation

The circuit was implemented as shown in figure 3.5. The unit element of stage 1 was implemented as shown in figure 3.6. Since stage 2 is exactly the same as that for the MPE, figure 3.3 was used for its implementation. The entire circuit was defined in purely structural Verilog code. Software was developed in Visual Basic to generate the structural Verilog code.

4.4.2 Layout of IMPE

The layout of the circuit also follows the block diagram as shown in figure 3.5. For the first stage, the input signals are grouped into 8 groups with each group having 16 inputs to it. Thus the second stage should have 8 inputs to it. This decision for taking 16 bit units in the first stage and the 8 bit second stage is based on the observation of the complexity in the crossover or the wrap around network of the LATPE circuit. From the layout of the LATPE it was found that with the wrap around network, the circuit didn't require extra buffer stages to drive the interconnect networks if the input was 8 bits. For the basic unit of stage 1 in figure 3.5, the circuit is very similar to stage 2 but without the wrap around network. Thus for this stage, the maximum number of inputs that can be placed in a group without requiring the circuits to have the extra buffer stages for driving the interconnect load was around 16. Since the wrap around network also provides significant load, the unit element of stage 1 can have more inputs than for stage 2. As is the case for the layout of the LATPE circuit, each of the gates are implemented in the NAND-NOR form rather than in the AND-OR form. All the gates are implemented in fully complementary static CMOS form and are shown in figure 4.4, 4.5 and 4.6 for the NOT, NAND and the NOR gate respectively. The floor plan for the layout is given in figure 4.7, which is very similar to the block diagram shown in figure 3.5.

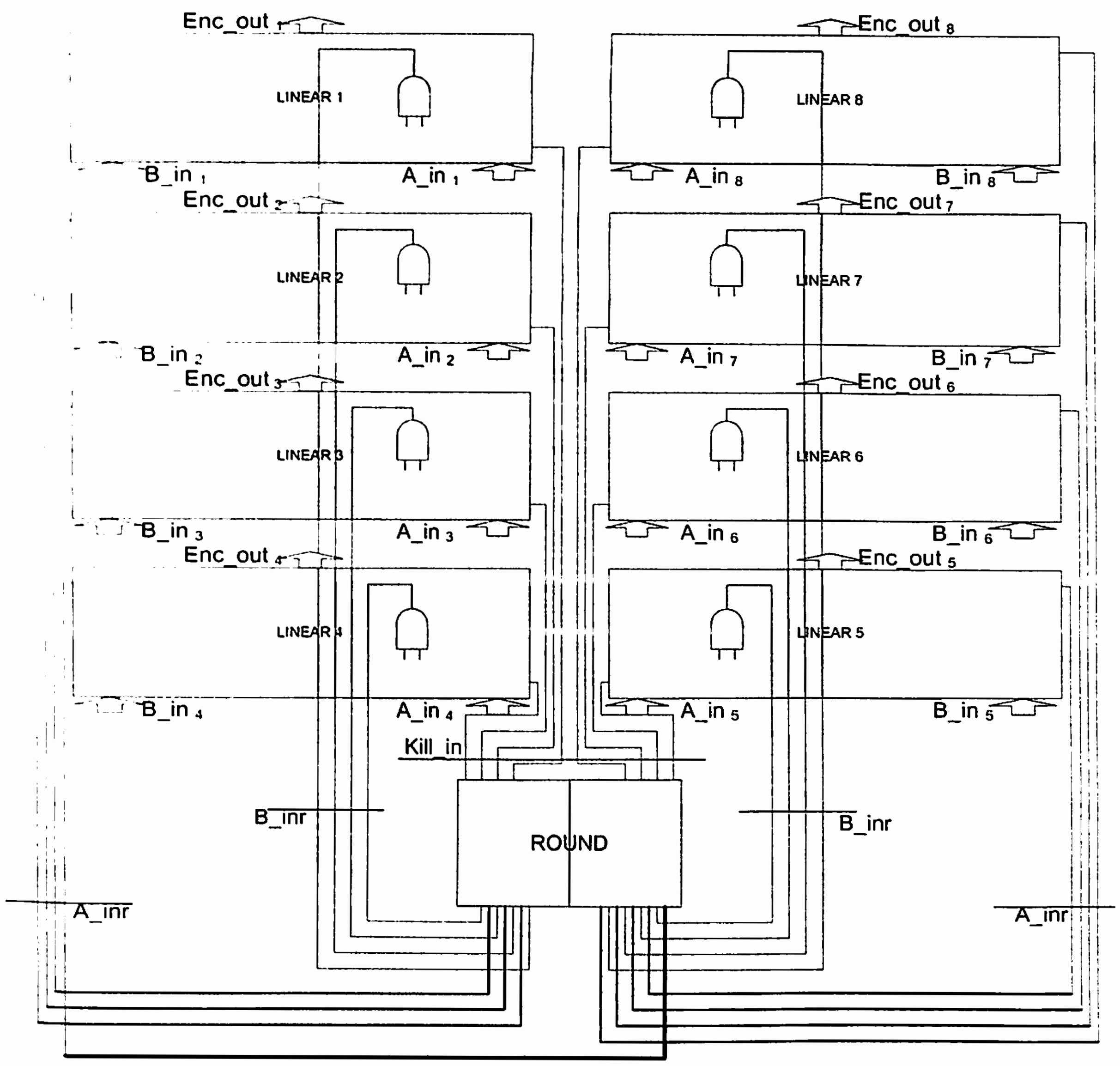


Figure 4.7: Floorplanning for the 128-bit IMPE

The block labeled ROUND contains the circuitry for stage 2 as shown in figure 3.3. All the blocks that are labeled LINEAR contain the circuitry for stage 1 of the IMPE as shown in figure 3.6.The layout for only two blocks needs to be done, one for the ROUND block and the other for the LINEAR block. Then the eight LINEAR blocks and the single ROUND block can be interconnected as shown in the figure above. The layout is made such that the encoder circuit as a whole is more or less square in shape. This helps to keep the circuit compact. Also since the circuit consists of several smaller blocks that are connected by a very few lines, the shape of the encoder can be adjusted to the available chip area if the encoder is used in conjunction with other circuits. Appendix B shows the layout for the 128 bit IMPE. The above circuit has been planned to be fabricated from MOSIS, thus additional circuitry are needed for connecting the inputs and outputs to the external world. The next chapter defines the floor plan and the additional layout issues that are associated with the complete chip.

Chapter 5

Chip Layout of 128-bit IMPE

5.0 Introduction

The IMPE circuit with complete input and output pads has been laid out to be fabricated using the MOSIS fabrication process. In addition to the basic units for the IMPE circuit that have been described above, other circuits are also needed to make the chip easily testable. The chip has to be tested both for functionality and speed. The pad circuits need to be used to interface with the external world. Since the encoder is 128 bits in size, there are 128 a_in inputs, 128 b_in inputs and 128 enc_out outputs. Thus the total numbers of input and output pads that are needed are around 384. To fabricate a chip with this many inputs requires a very expensive package. One common method that has been used to get around this problem is to use scan registers. This methodology has been implemented to reduce the number of inputs and outputs so that the entire circuitry can fit into a 40-pin package. The pad frame has been supplied by MOSIS group. With the use of scan registers, the additional issue of supplying the clock signals to the registers arises. Also there has to be some mechanism to determine the speed of the encoder circuit alone without the additional delay of the scan registers and the input output pad drivers.

5.1 Input/Output Pads

The signal generated by the circuit has to be transferred to the external world. The pad drivers are used for this purpose. The pads are then connected to the pin of the chip package using bond wires. For the purpose of testing the IMPE in this thesis, a pad frame laid out using 0.6μ technology. The pad frame is shown in figure 5.1. The pad frame was provided by MOSIS and was not designed as part of this thesis.

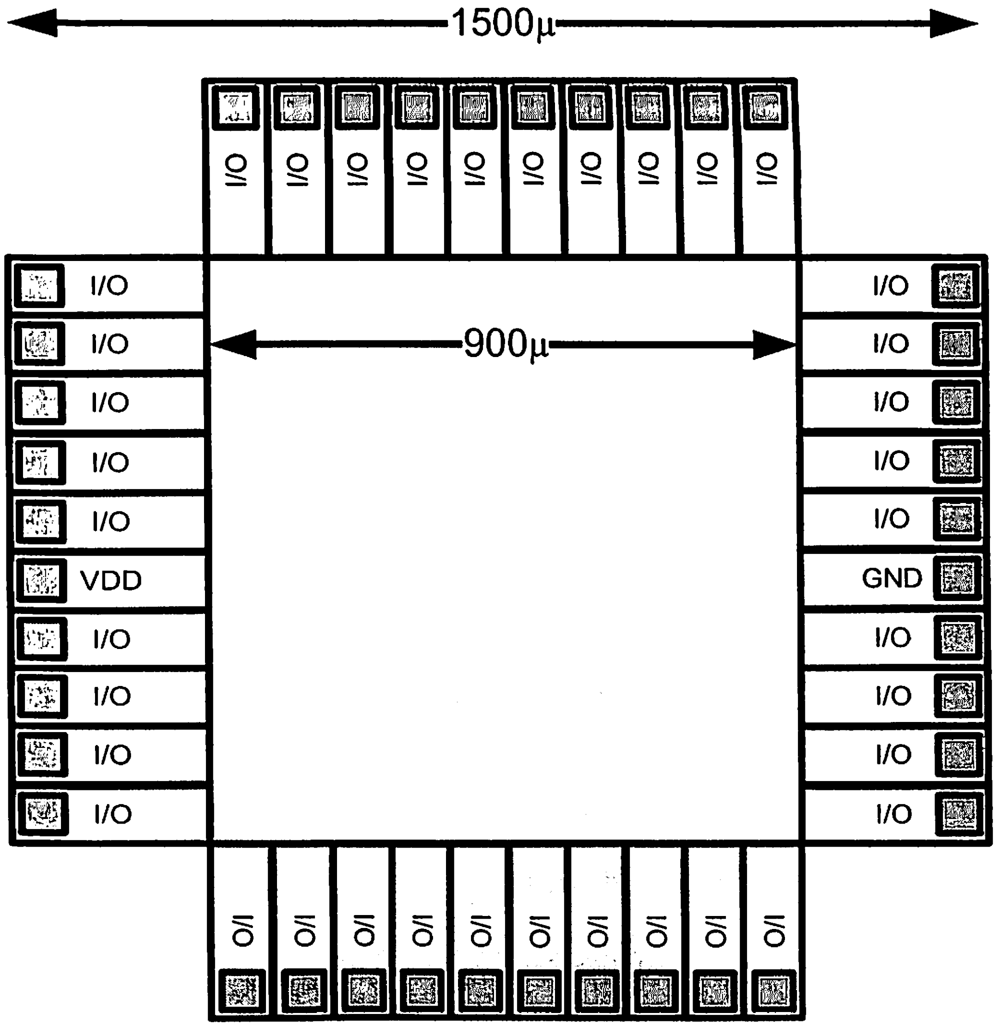


Figure 5.1: 40-pin pad frame

There are 40 pads, with 38 of them being the bidirectional input/output pads and the remaining two being used for the ground and the VDD connections. The structure for the bidirectional input/output pads is shown in figure 5.2.

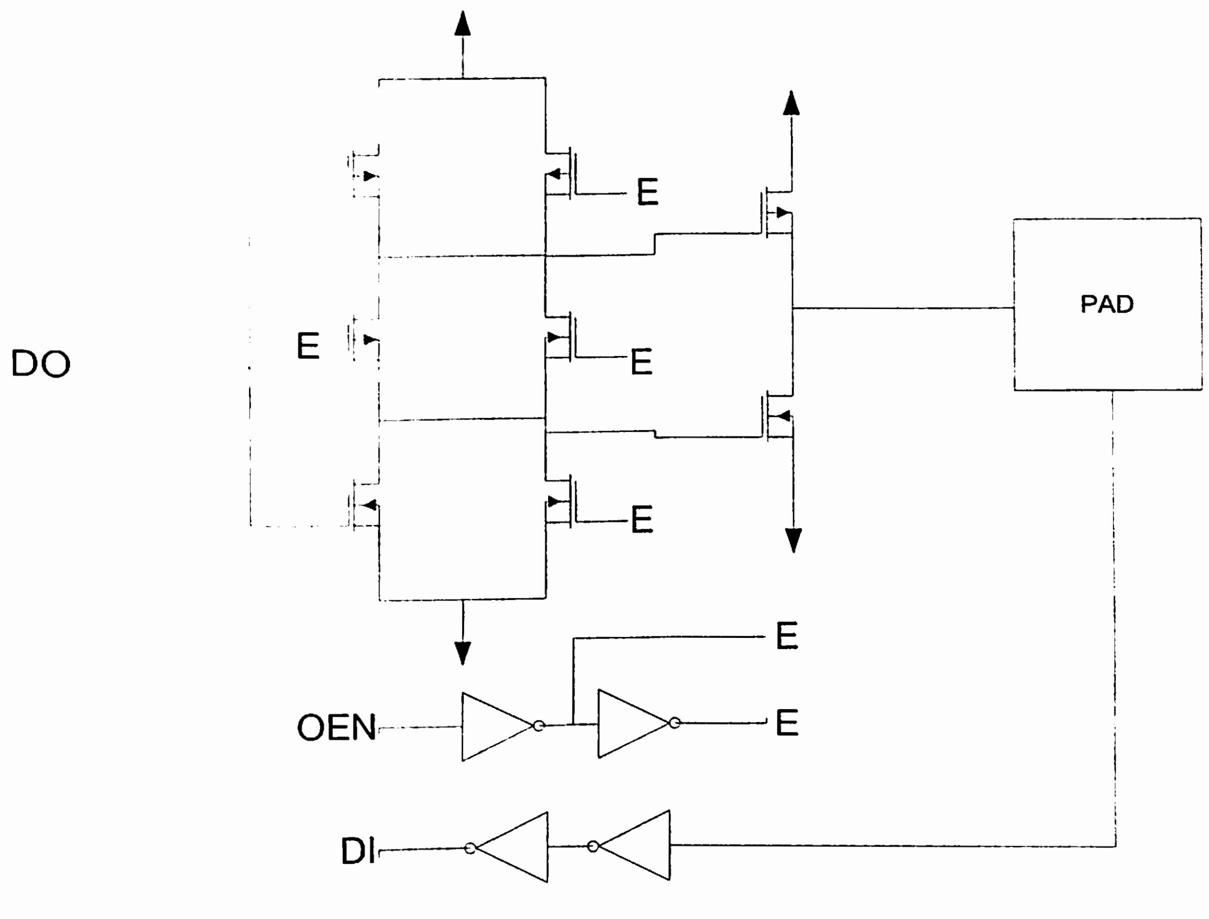


Figure 5.2: Circuit diagram for the bidirectional input/output pad

All the incoming and the outgoing signals are defined with respect to the chip. Thus an output signifies that the signal is fed from the circuitry to the external world. The input signal signifies the signal flowing from the external world to the chip. Since each of the input/output pads is bidirectional, they can be used either as an input pad or an output pad. The OEN input is used to define whether the pad works as an input or the output. If OEN is set at logic high, then the pad works as an output pad and if it is set at logic low it works as an input pad. The OEN input should be connected to the logic low level for all the pads that are not used or used as input only pads. The transistors that are used for the output buffer are very large with W/L of the order of 100. Also the channel length is made larger than the minimum length allowable to provide a high breakdown voltage. To prevent the gate resistance to grow to a large value for these wide transistors, folded layouts are used. To prevent the latch up problem for these wide transistors guard rings are used around the transistors.

5.2 Scan-Based Test Technique

The IMPE requires 128 a_in inputs and 128 b_in inputs as well as 128 outputs from the encoder circuit. Thus a large number of input/output ports are required. In order to reduce the number of input/output ports to a more practical limit, several approaches available for testing can be used. In the current thesis, serial and parallel scan based techniques have been incorporated. A general block diagram for the serial scan technique is shown in figure 5.3. It shows that the combinational logic that has to be tested is sandwiched between two shift registers. The first shift register takes the input from the external world through the single input terminal termed as Scan_in in serial fashion. After all the input signals are scanned into the input register, the entire input signal is applied to the circuit under test in parallel fashion. The circuit under test in this case is the combinational logic shown in figure 5.3. The circuit under test operates on the input and generates the output signals, which is latched into the output scan register in parallel fashion. The content of the output scan register is then clocked out serially through the

Scan_out terminal to the external world. Thus using this approach the number of input/output pads can be significantly reduced.

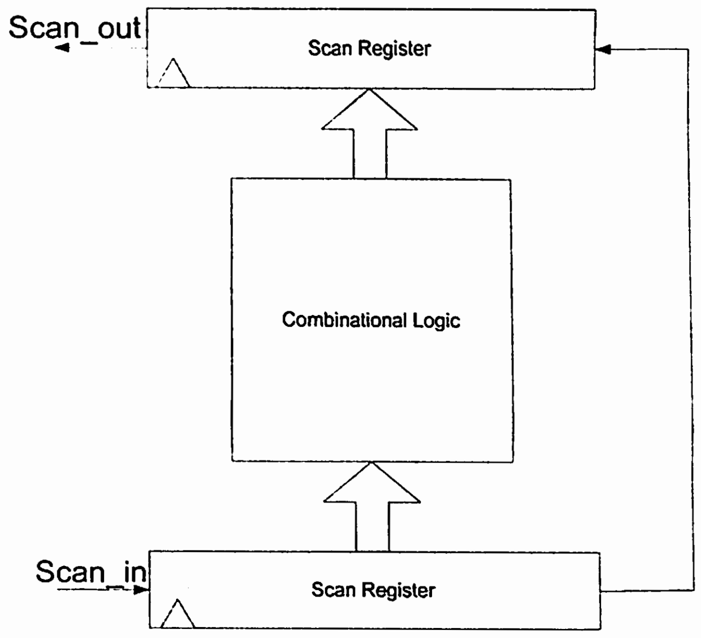


Figure 5.3: General diagram for scan based testing technique

If a single scan register is used for the entire inputs and outputs, the scan chain will be quite long which can dominate the testing time. The scan chain can be broken into several small chains. Thus in this approach, there can be more than one scan register at the inputs and the outputs. More about the scan based techniques can be found in reference 3.

Each of the scan registers is implemented with of edge sensitive registers. Several different designs of the edge sensitive registers exist. The following subsections provide the design that has been used in this thesis and other issues related with the design.

5.2.1 Scan Registers

The scan registers are implemented by cascading several edge sensitive latches in series. Different versions of the edge sensitive register design exist. For this thesis, static true two phase positive edge sensitive registers have been used. Figure 5.4 shows the basic diagram for the positive edge sensitive register used.

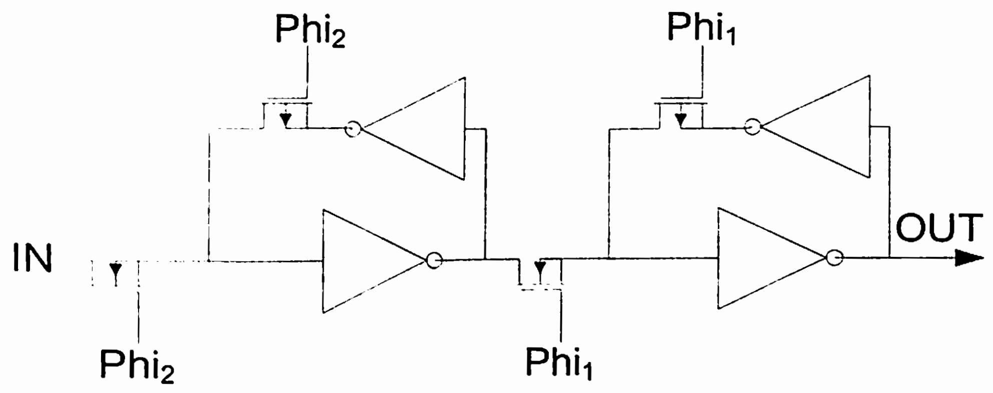


Figure 5.4: Schematic for a true two-phased clock positive edge clock latch

The two-phased approach is used to make the register immune to clock skew. Clock skew and clock distribution problem are discussed in the next subsection. Phi_1 and Phi_2 are the non-overlapping clock signals. The system can be made to work by increasing the non-overlapping period between Phi_1 and Phi_2. Since non-full swing logic has been used, this approach consumes more power. But since it requires only two clock lines, clock distribution is easier. In this thesis, power is not an issue because there are only a few registers. Reducing area is more important since the entire IMPE circuit along with the scan registers has to fit into the pad frame with a central area of $900\mu \times 900\mu$. The true two phase approach which requires only two clock lines as opposed to the lower power consuming pseudo two phase clocked approach, which requires four clock lines, is more advantageous.

5.2.2 Clock Generation and Distribution Issues

If a single-phase clock is used, the different edge sensitive latches of the scan register may receive the clock signal at different times. This is known as clock skew. Clock skew can be a significant problem in the circuit implemented as shown in figure 5.4. One of the common methods is to use non-overlapping two-phase clock signals. The non-overlapping clock signal can be generated from the single clock line as shown in the figure 5.5.

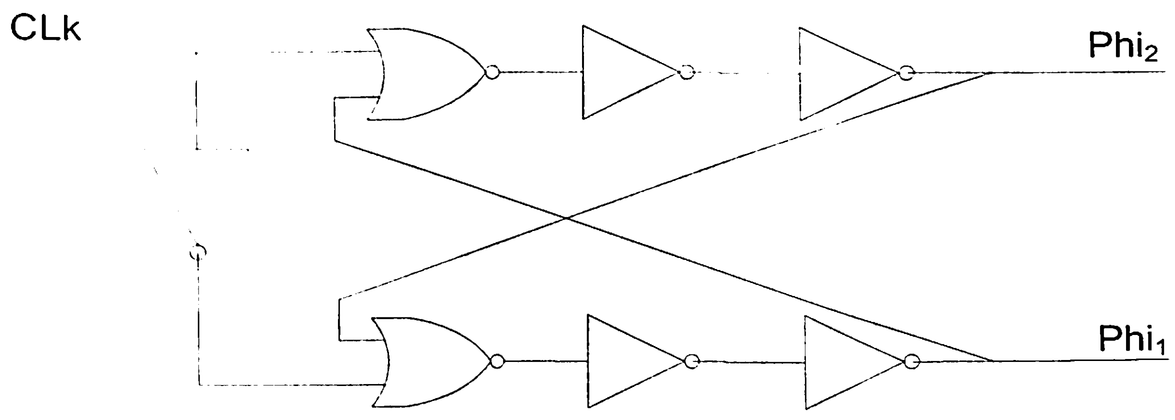


Figure 5.5: Two-phase clock generator

Adding or removing the inverters can vary the non-overlap period. Figure 5.6 shows the signal Phi_1 and Phi_2.

Another issue that has to be addressed is that of clock distribution. After the clock generator generates the two-phase clock signals, both of these signals have to be routed to each and every edge triggered latch in the scan registers. In order for the clock signal to reach to all the registers at almost the same time, a balanced clock tree or the H clock tree can be used [4]. The H tree is shown in figure 5.7.

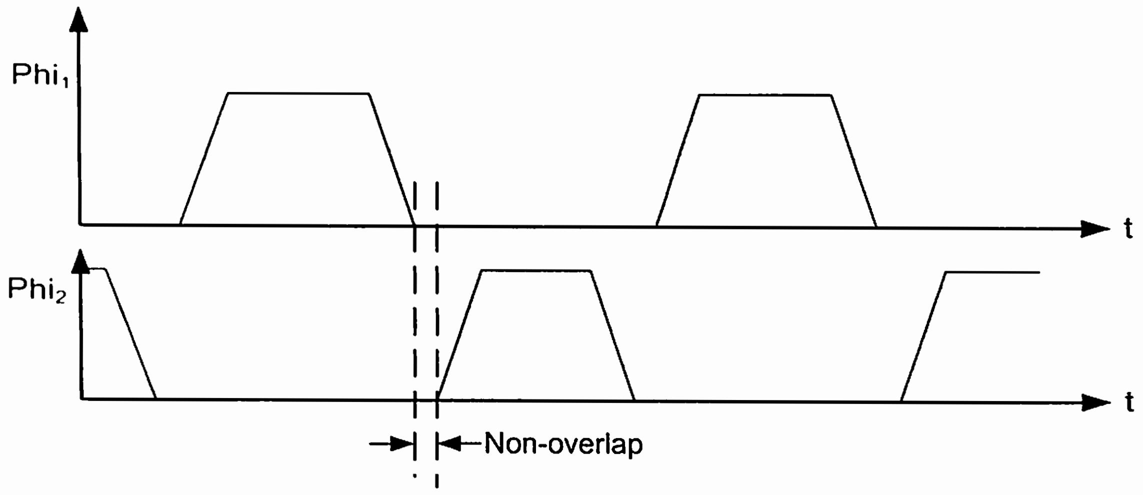


Figure 5.6: Non Overlapping - clock

The H tree is a very regular structure with recursive construction of H's. For each level of H structure, four smaller H structures can be added at the four end points of the H bars [4]. The size of the wires can be varied to account for the variations in the load capacitance to equalize the skew throughout the H tree [4].The H tree defines the floor plan of the circuit which may not be suitable for all cases. A balanced tree is based on the concept of the H tree, where the tree is irregular in shape but the branches are arranged so as to balance the branches to minimize the skew [4].

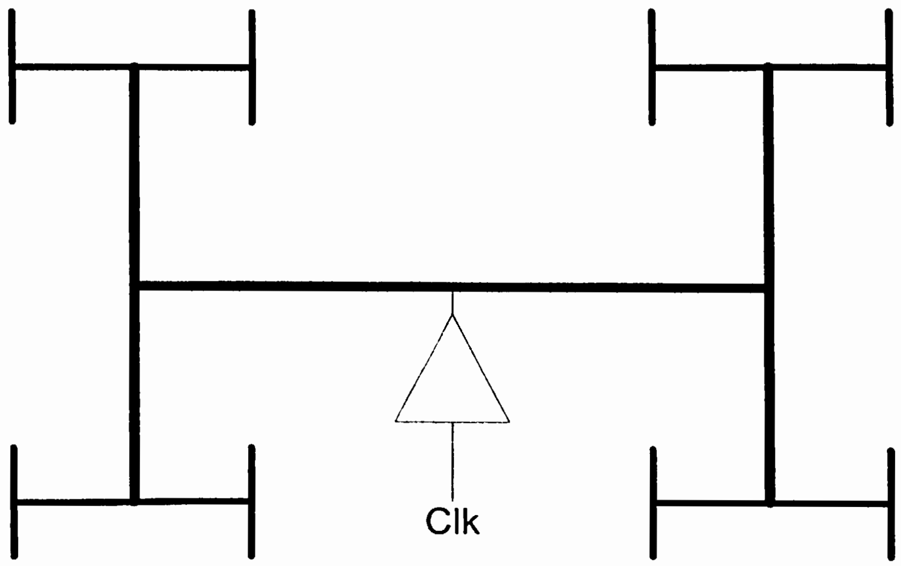


Figure 5.7: H-tree for the distribution of clock

5.3 Floorplan of IMPE in Pad Frame

The IMPE circuit has been laid out inside the 40-pin pad frame with the internal area of $900\mu \times 900\mu$. The floorplan is shown in figure 5.8 and the layout in Appendix B. In order to prevent very large scan chains, the scan registers have been broken down into several small units. There are 12 scan registers. Each of these scan registers are 32 bits in size. The entire stage 1 of the IMPE has been divided into four groups. Since there are eight unit elements in stage 1 for the 128-bit encoder, each of the four groups consists of two unit blocks. Thus each group has 32 inputs. Tn figure 5.8, signals are named after the group to which they serve. Thus each 128-bit a_in input to the encoder has to be broken down into four groups of 32 bit words and fed serially through the input terminal ain_1, ain_2, ain_3 and ain_4. Similarly the 128 bit b_in inputs are fed in serially through the input pins named bin_1, bin_2, bin_3 and bin_4. These 8 input registers are implemented using the edge triggered latched as shown in figure 5.4. A single true two-phase clock network controls all of these eight scan registers. The input terminal named, Clock_in, is used to give the clock signal to these eight registers. Since a single clock line is used to control all eight registers, the serial inputs to all the eight registers need to be supplied at the same time. The clock signal supplied by the external circuit is converted to the two-phase clock signal using an internal circuit as shown in figure 5.5. The distributions of these two phases are done utilizing a balanced tree clock distribution network as shown in figure 5.8. Since these two phase clock distribution networks run all over the chip surface, they provide considerable capacitive load to the two-phase clock generator circuit driving it. Thus a buffer stage is introduced for each phase to improve the driving

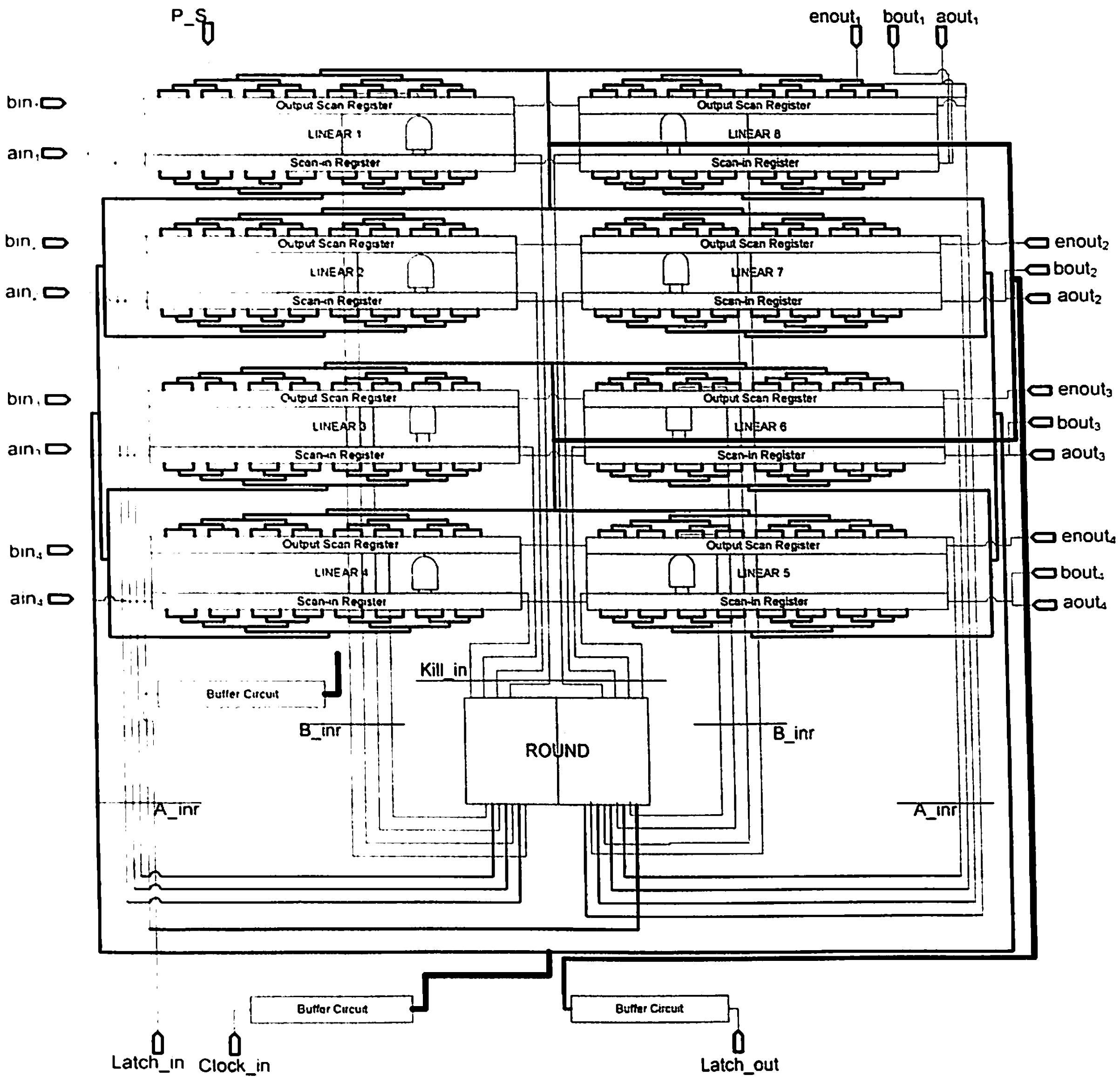


Figure 5.8: Floorplan with scan registers for a 128-bit IMPE

capability. The circuit diagram for the buffer is as shown in figure 2.9.

Each output from the scan register needs to be fed to the input of the encoder. Latches at the output of each of the scan-in registers are used to gate the input signals to the encoder circuit. While the scan-in registers are being fed serially, this latch is turned off thus disconnecting the scan-in registers from the encoder circuit. During this time, some input signals must be provided to the encoder circuit. Thus a second latch is connected to the inputs of the encoder circuit. The input to these latches is fixed by hardwiring them either to the ground or the supply terminal during layout. This latch should turn on when the latch connected to the output of the scan-in register is turned off. And it should turn off when the latch connected to the output of the scan-in registers are turned on. Thus these two latches connected to each of the input terminals of the encoder circuit should have complimentary control inputs. These two latches together act like a 2-1 multiplexer. The schematic for the latch is shown in figure 5.9. Since all of these latches need to operate at precisely the same time so that they provide signals to all the inputs of the encoder at the same time, the control signal to all of the latches must arrive at the same time. Thus a balanced tree is used to supply the control signals to these latches. This control signal is connected to the input pad named, Latch_in. In order to enhance the driving capability this balanced tree is also driven by a buffer circuit as shown in figure 2.9.

Finally there are four output registers, each of them are 32-bits wide. These registers operate slightly differently than the scan-in registers. While the scan in registers operate in serial input, parallel output and serial output mode, the output registers operate in parallel in and serial out mode only. The parallel input to the output registers are provided by the output from the encoder circuit. These outputs from the encoder circuit are latched into these registers by means of control signal named, Latch_out. With a two-phase clock generator circuit, this Latch_out signal is transformed in to true two non-overlapping clock signals, which are distributed to all the latches in the output registers by a balanced clock distribution network. After the output from the encoder is latched to the output registers, the content of these registers are serially scanned out for the purpose of verification. Thus the four output pads $enout_1$, $enout_2$, $enout_3$ and $enout_4$ are used to

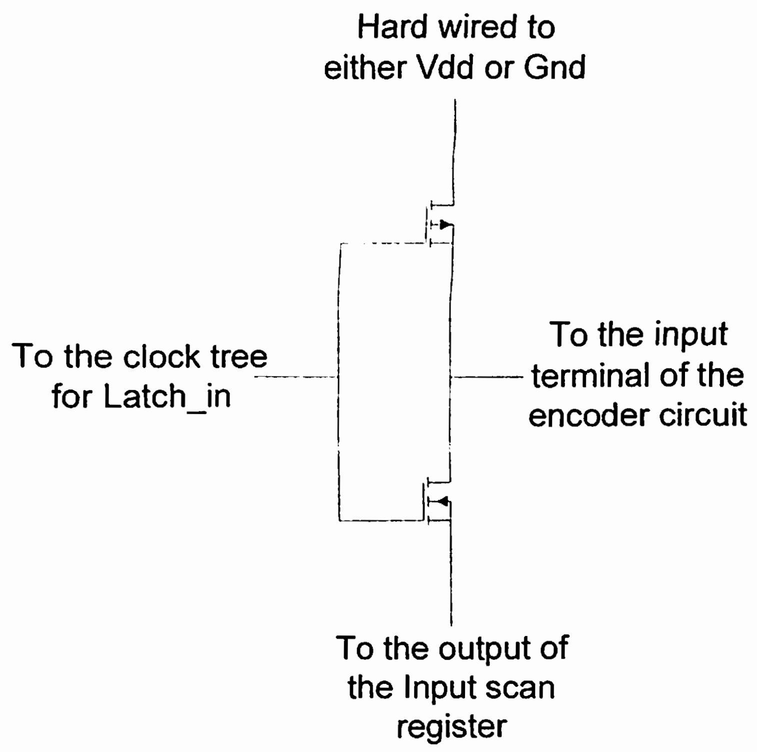

Hard wired to either Vdd or Gnd
To the clock tree for Latch_in
To the input terminal of the encoder circuit
To the output of the Input scan register

Figure 5.9: 2 – 1 multiplexer used in the input side of the encoder circuit

read the content of these output registers serially. Since the output registers are used in both parallel and serial mode, extra latches and control signals to separate these two conditions are needed. Thus an input pad named P_S is used for this purpose. This input signal is used to select the operation of the output registers in parallel or the serial mode. Figure 5.10 shows the latches that are needed for the output registers to function in either parallel or the serial mode as selected by the P_S input.

The outputs from all the eight scan-in registers are also read serially for the purpose of verification. These signals are connected to the output pads named $aout_1$,

$aout_2$, $aout_3$ and $aout_4$ for the a_in inputs and the output pads $bout_1$, $bout_2$, $bout_3$ and $bout_4$ for the b_in inputs.

The VDD and the Ground lines are supplied to all the circuit blocks shown in figure 5.8 by using metal layers. Also all of the clock distribution lines are implemented in metal 1, metal 2 or metal 3 wherever possible.

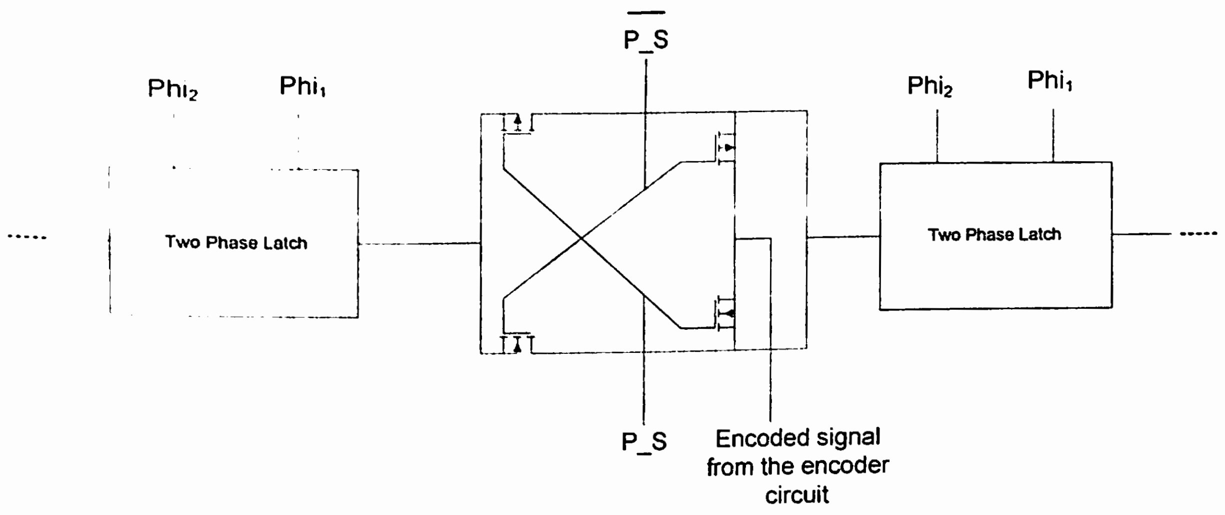


Figure 5.10: Latch circuit for the output registers

Chapter 6

Testing and Verification Results

6.0 Introduction

This chapter presents the testing methodologies used to verify the functionality of the encoder circuit and the results from the simulation. All of the different approaches for the encoder designs were fully tested and verified for their functionality and correctness.

All the circuits that were simulated were tested using a set of test vectors. Since all the circuits that were tested had 256 inputs, exhaustive testing was not possible. For a 128-bit encoder there are 128 a_in inputs and 128 b_in inputs. Since each of the 128 a_in inputs can be either at logic high or low, irrespective of what logic levels the other bits are at, there can be $2^{128} = 3.402 \times 10^{38}$ total input combinations. Since only one of the b_in inputs can be at logic low at any time and all the remaining should be at logic high, the total number of input combinations for the b_in inputs is $128=2^7$. Combining the 128 a_in inputs and 128 b_in inputs the total input bits combination is $2^{128+7} = 2^{135} = 4.355 \times 10^{40}$. With a typical testing device that can provide a test vector and sample the output at a clock frequency of 500 MHz, then it will take 2.76×10^{24} years. Thus, the exhaustive testing of the circuit is not possible. A practical approach is to test the circuit using a set of randomly generated test vectors from the full set of possible test vectors. A computer program was written in Visual basic to generate a set of randomly generated test vectors. All of the simulated circuits were tested using approximately 1000 test vectors. The following sections describe the procedures used for testing the circuits.

6.1 Testing Procedures

All of the simulated circuits except the chip implementation of the IMPE follow the same general procedure. The circuits that are simulated using the Cadence Virtuoso Layout Editor are first extracted to determine the parasitic capacitances. Then a netlist of the circuit is generated. Then the test file is generated using the 'test vector generator software'. The test file contains the test vectors for verification as well as other information like the rate at which the inputs are to be supplied to the circuit under test (CUT), as well as the expected output from the CUT. For the circuit simulated using the Verilog code the extraction procedure is not necessary. As mentioned earlier, there are no inclusion of the capacitive load due to the interconnect networks. Apart from the chip implementation of the IMPE circuit, all the simulated circuits have parallel input, and parallel output. Thus the testing procedure merely involved applying the test vectors one at a time, waiting for a preset interval of time and then capturing the output of the CUT. This output is then compared to the expected output. If the expected output for all of the test vectors matches that of the simulated output, the circuit is working correctly in the preset interval. The procedure is repeated until a least time interval between the applying of the test vector and capturing the output of the CUT is found for which the circuit output matches the expected output. This time interval is the minimum time interval at which the circuit works as expected. The inverse of this time will give the maximum frequency at which the circuit can operate.

The testing for the chip implementation of the IMPE circuit is different from the other. The testing procedure follows the flowchart as shown in figure 6.1. First the output register is configured into parallel mode by setting the P_S signal high. Then the inputs ain_1, ain_2, ain_3, ain_4, bin_1, bin_2, bin_3 and bin_4 are set to the desired value of 1 or 0 and the Clock_in signal is applied. Since each of the scan in registers are 32 bits wide, the input pattern is changed 32 times and then the Clock_in signal is applied. To allow the signals to be stable a certain delay is needed after this operation. After this operation, the input signals need to be fed into the encoder circuit. Thus, the Latch_in signal is set high. This operation connects the input of the circuits to the output of the scan-in registers, which have the valid input signals. After applying the input signals a certain delay is allowed until the output of the encoder circuit becomes valid. This set amount of time is determined iteratively, initially it is set to some value. If the circuit performs successfully in that time the duration is reduced, until the output of the circuit doesn't match the expected output. After waiting this preset interval of time, the output of the circuit is latched into the output registers. This is done by setting the Latch_out signals high. After the output signal is latched to the output registers, the output registers are configured to work in serial mode, by setting the P_S signal to 0. Then the content of the output registers as well as the input registers are read out serially by clocking 32 times into the

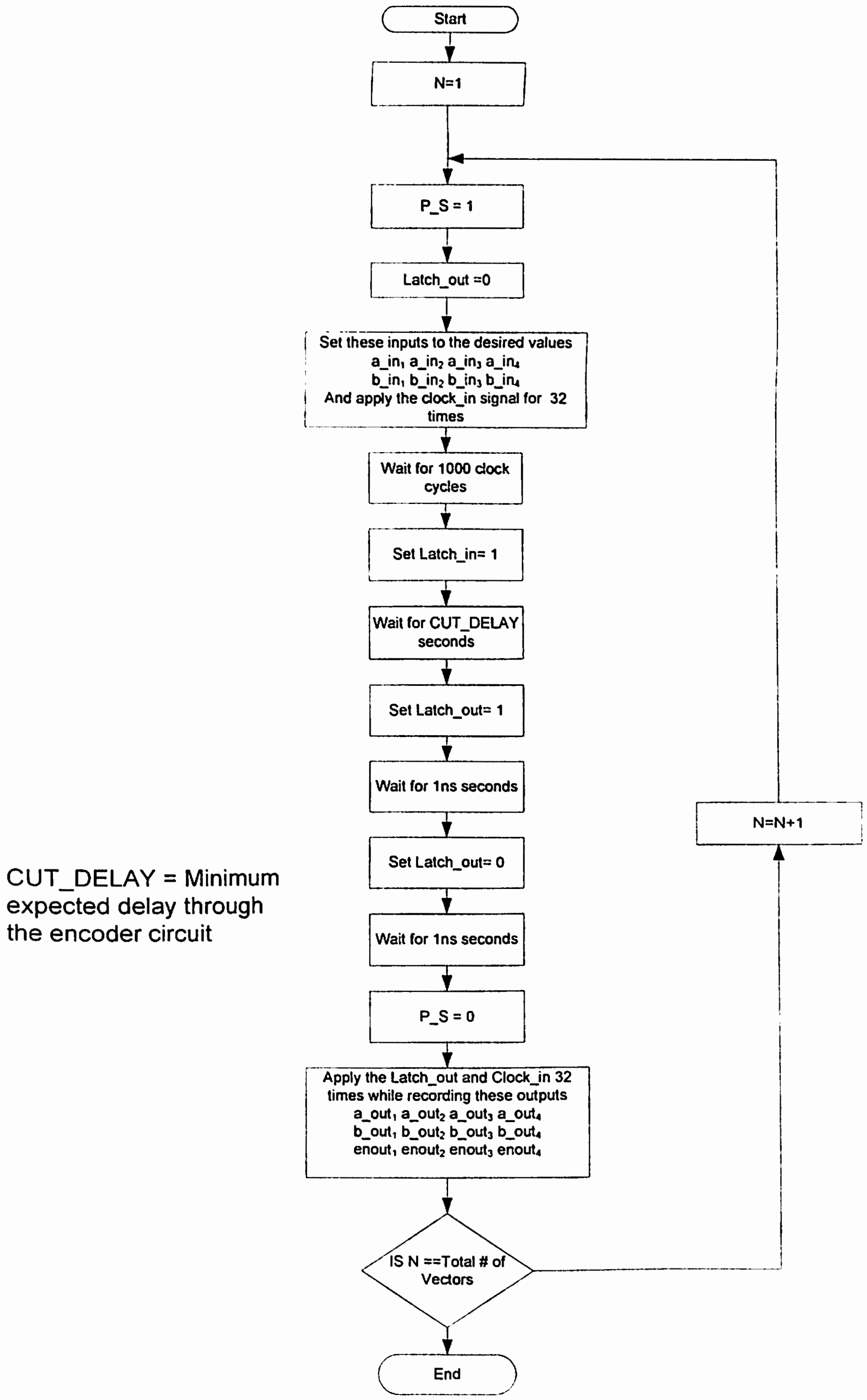

Start
N=1
P_S = 1
Latch_out =0
Set these inputs to the desired values
a_in1 a_in2 a_in3 a_in4
b_in1 b_in2 b_in3 b_in4
And apply the clock_in signal for 32 times
Wait for 1000 clock cycles
Set Latch_in= 1
Wait for CUT_DELAY seconds
Set Latch_out= 1
Wait for 1ns seconds
Set Latch_out= 0
Wait for 1ns seconds
P_S = 0
Apply the Latch_out and Clock_in 32 times while recording these outputs
a_out1 a_out2 a_out3 a_out4
b_out1 b_out2 b_out3 b_out4
enout1 enout2 enout3 enout4
IS N ==Total # of Vectors
End
N=N+1
CUT_DELAY = Minimum expected delay through the encoder circuit

Figure 6.1: Flowchart showing testing procedure for the chip implementation of IMPE

Clock_in and the Latch_out signals. These serially read signals are stored and then combined to form a complete 128-bit string which are then compared with their expected values. The time interval denoted as CUT_DELAY in figure 6.1 denotes the time interval between the event when the Latch_in signal is turned high and the event when the Latch_out is turned high.

For a particular value of CUT_DELAY, if the output of the circuit is as expected, then the circuit performs as expected within that time interval. The value of the CUT_DELAY can be progressively decreased until one or more of the output just doesn't match the expected output, which gives the minimum time interval at which the circuit performs properly. In the flowchart shown in figure 6.1, the CUT_DELAY also includes the delay caused by the latch circuits as well as the setup and hold time due to the output registers. To determine these extra delays in the hardware circuit is complicated. Thus, one can use a circuit simulator to determine the extra delays and then subtract them from the CUT_DELAY to find the actual delay through the encoder circuit alone.

6.2 TESTING RESULTS

Table 6.1 presents the delay through each of the circuits that were designed and simulated for this thesis. It should be noted that the Verilog implementation uses the 0.35μ cell library, while the layout in the Cadence Virtuoso editor was done for 0.6μ process.

Circuit Name	Simulated Using	Delay (ns)
BSPE	Verilog	4.5
LATPE	Verilog	2.4
LATPE	Cadence Virtuoso	50
MPE	Verilog	3.7
IMPE	Verilog	3.4
IMPE	Cadence Virtuoso	3.1
Chip Implementation of IMPE	Cadence Virtuoso	5

Table 6.1: Delay through the various versions of 128-bit encoder circuits

Also the chip area comparison is necessary in order to assess the true value of a design apart from the delay determination. Only two of the designs need a direct comparison of the area. The relative area for the rest of the designs can be approximated from their structure itself. The two designs that needs direct comparisons are the LATPE and the IMPE layout circuits. Table 6.2 presents the sizes for these two designs for the input size of 128-bits.

Design Type	Area (μ^2)
LATPE	$4575 \times 567 = 2594025$
IMPE	$1225 \times 637 = 780325$

Table 6.2: Chip area taken up by different designs of 128-bit priority encoders

Chapter 7

Conclusion and Future Work

7.0 Conclusion

The main objective of this study was to design a round robin priority encoder with a large number of inputs with a delay of O(logN) where N denotes the number of inputs to the encoder circuits. Several different designs were conceived and then simulated as well as verified using either Verilog HDL or the Cadence Virtuoso Layout tools. Each of the designs presents their own pros and cons which have been discussed in detail in the preceding chapters. The final results of the simulation are presented in table 6.1. The BSPE was simple in design but the fact that it has inherent delay of O(3* log N) as well as the fact that the interconnect network gets too complicated in the higher levels of the barrel shifters make them less favorable. Also the size of the encoder more than linearly increases with the increase in the number of the inputs. Thus this is not an area efficient design. The LATPE circuit has a smaller delay than the BSPE, which is also evident from Table 6.1. The LATPE has a delay of the order of O(logN). But this design still has larger interconnect loads for the higher levels, thus complicating the design. Also the relative chip area taken up by this circuit also increases linearly with thenumber of inputs and is thus not an area efficient design. The MPE design is very simple, since it reuses the hardware designs. Only a few basic blocks need to be designed. These basic blocks can be instantiated many times to realize the final circuit. Thus, it is very easy to increase the number of inputs to the circuit if needed. The area taken up by the IMPE circuit is also very small compared to the area taken up by the BSPE and the LATPE circuits. The delay through this circuit is slightly larger than the O(logN) delay for LATPE. The exact delay through the MPE design depends upon the number of groupings that have been made in stage 1 and stage 2 of the design. If the N inputs have been grouped into M groups of L signals in the stage 1 then the total delay is approximately given by log(L) + log(M) + log(L) = 2 log(L) + log(M). Where the stage1and stage 3 presents delay of the order O(logL) delays and the stage 2 presents delay of the order of O(logM). For the 128 bit encoder in which the input signals are bundled as 8 group of 16 inputs the total delay is given by $\log_2 16+\log_2 8+\log_2 16 = 11$ as opposed to the $\log_2 128 =7$. Thus the increase in the delay is minimal, whereas the area reduction is very high, as well as the reduction in complexity of the design. The IMPE has a similar delay as that for the MPE circuit, but the added advantage is the reduction in area to almost half as compared to the MPE. This is due to the fact that the IMPE utilizes the similarities in structure of stage 1 and stage3 of the MPE circuit and combines them into a single stage. Since stage 1 and stage 3 of the MPE are the major contributors to the area of the MPE circuit, the combination of stage 1 and stage 3 in the IMPE reduces the area to almost half. Although the area for the IMPE is almost half of the area of the MPE, the delay is almost the same. The reason for this is that the signal travels twice through stage 1 of IMPE. Thus the delay remains unchanged as compared to the MPE circuit. For the IMPE circuit only two basic building blocks need to be designed, one for the stage 2 and another for the basic unit of stage1. The final circuit uses several instances of these basic building blocks, thus making the design procedure very simple. Also since each of the building blocks have very few inputs to them, the interconnect networks in these blocks do not offer large capacitive loads to the stage driving them, thus obviating the need for the buffer circuits. This is another reason why these circuits are easy to design. The comparison of the area of the LATPE circuit against that of the IMPE is presented in table 6.2, which shows a large reduction in area. The IMPE circuit was also included into a 40 pin pad frame for tape out of the circuit. The circuit has been designed to be fabricated in a 0.5μ AMI process offered by MOSIS. The final circuit has additional circuits apart from the IMPE to facilitate testing and verification. These additional circuits add extra delays, which is apparent from table 6.1. The delay caused by these extra circuits needs to be subtracted from the total delay to determine the true delay of the encoder circuit alone.

7.1 Future Studies

The IMPE circuit has a similar delay as that of the MPE circuit but the area for the IMPE is half the area for the MPE. If the speed is the concern and not the area, then the MPE circuit can be slightly modified and used to form a very fast circuit. From the structure of the MPE, it is quite obvious that the circuit consists of three distinct stages. The interconnections between these stages are small in number. Moreover, the circuit uses the three stages one at a time, in a serial fashion. When stage 1 is being used, stage 2 and stage 3 are not being used and similarly when stage 3 is being used stage 1 and stage

2 are not being used. The lower stages are just preserving their outputs until the higher stages reach a stable state. Thus pipeline registers can be included between these stages that preserve the state of the lower level stage. If this is done the lower level stages are free to start processing a new set of inputs. The circuit can be called the pipelined MPE (PMPE). Thus using the pipelined registers helps to greatly increase the throughput of the circuit. The pipelined registers need to be clocked at a certain frequency which is determined by the stage with the largest delay in it. For a 128 bit encoder, if the input signals are bundled into 8 groups of 16 signals, then the delay through the first and the last stage are the largest. These delays determine the pipeline clock frequency. The delay is given as $\log_2 16 = 4$, which is smaller than the $\log_2 128 = 7$. Apart from the delay of $\log_2 16$ there will be a small additional delay due to the setup and the hold time of the pipeline registers. The disadvantage of this design is that, although the circuit works at a very fast speed, there is some latency involved. The latency of the circuit is 3 pipeline clock cycles. Thus for the applications which cannot afford latency, this approach will not be useful. If the circuit operation is not affected by the latency then this approach provides a great promise for a very fast operation. If the application circuit permits more latency, then the MPE circuit can be modified to have a larger number of simpler stages, thus further increasing the circuit throughput.

References

[1] Gupta, Pankaj; Nick, McKeown; *Designing and Implementing a Fast Crossbar Scheduler*, Proc. The 45th IEEE International Midwest Symposium on Circuits and Systems 2002, Page(s): 636-641

[2] Sheng, S. Q.; Yang, Mei; Blanton, John; Galla, Prasad; Verchere, Dominique; *A Simple and Fast parallel Round-Robin Arbiter for High-speed Switch control and Scheduling*, Micro, IEEE, Volume:19, Issue:1, Jan.-Feb. 1999, Page(s): 20 –28

[3] Weste, Neil H. E.; Eshraghian, Kamran; *Principles of CMOS VLSI design. A systems perspective,* Second Edition

[4] Wolf, Wayne; *Modern VLSI Design, Systems on Silicon,* Second Edition

[5] Goldovsky, A.; Srinivas, H.R.; Kolagotla, R.; Hengst, R.; *A folded 32-bit prefix tree adder in 0.16-μm static CMOS* Circuits and Systems, 2000. Proceedings of the 43rd IEEE Midwest Symposium, Volume: 1, 8-11 Aug. 2000 Page(s): 368 -373 vol.1

[6] P. Brent, Richard;Kung,H.T.; *A regular Layout for parallel adders*, IEEE Transactions on Computers, 31: 260 -264 , Mar , 1982.

[7] Beaumont-Smith, A.; Lim, C.-C.; *Parallel prefix adder design,* Computer Arithmetic, 2001. Proceedings. 15th IEEE Symposium on, 11-13 June 2001, Page(s): 218 -225

[8] Jinn-Shyan Wang; Chung-Hsun Huang; *High-speed and low-power CMOS priority encoders,* Solid-State Circuits, IEEE Journal of, Volume: 35 Issue: 10, Oct. 2000, Page(s): 1511 -1514

[9] Delgado-Frias, J.G.; Nyathi, J.; *A VLSI high-performance encoder with priority lookahead,* VLSI, 1998. Proceedings of the 8th Great Lakes Symposium on , 19-21 Feb. 1998, Page(s): 59 -64

[10] Delgado-Frias, J.G.; Nyathi, J.; *A high-performance encoder with priority lookahead,* Circuits and Systems I: Fundamental Theory and Applications, IEEE Transactions on, Volume: 47 Issue: 9, Sept. 2000, Page(s): 1390 -1393

[11] Jinn-Shyan Wang; Chung-Hsun Huang; *High-speed and low-power CMOS priority encoders,* Solid-State Circuits, IEEE Journal of , Volume: 35 Issue: 10 , Oct. 2000, Page(s): 1511 -1514

[12] Katevenis, M.; *Fast Switching and Fair Control of Congested Flow in Broadband Networks,* Selected Areas in Communications, IEEE Journal on, Volume: 5 Issue: 8, Oct 1987, Page(s): 1315 -1326

[13] Sun Huajin; Gao Deyuan; Zhang Shengbing; Wang Danghui; *Design fast round robin scheduler in FPGA,* Communications, Circuits and Systems and West Sino Expositions, IEEE 2002, International Conference on, Volume: 2, 29 June-1 July 2002, Page(s): 1257 -1261 vol.2

[14] Yamagata, T.; Mihara, M.; Hamamoto, T.; Kobayashi, T.; Yamada, M.; *A 288-kbit fully parallel content addressable memory using stacked capacitor cell structure,* Custom Integrated Circuits Conference, 1991., Proceedings of the IEEE 1991 , 12-15, May 1991, Page(s): 10.3/1 -10.3/4

[15] Yamagata, T.; Mihara, M.; Hamamoto, T.; Murai, Y.; Kobayashi, T.; Yamada, M.; Ozaki, H.; *A 288-kb fully parallel content addressable memory using a stacked-capacitor cell structure,* Solid-State Circuits, IEEE Journal of , Volume: 27 Issue: 12 , Dec. 1992, Page(s): 1927 -1933

[16] Shin, E.S.; Mooney, V.J., III; Riley, G.F.; *Round-robin Arbiter Design and Generation,* System Synthesis, 2002. 15th International Symposium on, 2-4 Oct. 2002, Page(s): 243 -248

APPENDIXES

Appendix A

Description of 0.35 Micron CMOS Standard Cells Used

AA21

Description

AA21 is a two input gate which performs the logical AND function.

Verilog Syntax: AA21 inst_name(Q,A,B);

Where, Q is the output and, A and B are the two inputs to the AND gate. And each of the pins A and B offers 1 equivalent load to the circuit driving it. Each equivalent load is equal to 27.7fF.

Propagation Delays (ns)

No of Equivalent Loads		1	4	8	11	15
From Any input to Q	t_{PLH}	0.15	0.23	0.34	0.42	0.53
	t_{PHL}	0.14	0.23	0.34	0.41	0.5

t_{PLH} : Input to output propagation delay for a rising edge on the output

t_{PHL} : Input to output propagation delay for a falling edge on the output

Also each of the equivalent load in this table needs to be multiplied by 1.47, because an interconnect capacitance of 13fF is already added while calculating the delays.

OR21

Description

OR21 is a two input gate which performs the logical OR function.

Verilog Syntax: OR21 inst_name(Q,A,B);

Where, Q is the output and, A and B are the two inputs to the OR gate. And each of the pins A and B offers 1 equivalent load to the circuit driving it. Each equivalent load is equal to 27.7fF.

Propagation Delays (ns)

No of Equivalent Loads		1	4	8	11	15
From Any input to Q	t_{PLH}	0.13	0.22	0.33	0.41	0.51
	t_{PHL}	0.17	0.27	0.37	0.45	0.54

t_{PLH} : Input to output propagation delay for a rising edge on the output

t_{PHL} : Input to output propagation delay for a falling edge on the output

Also each of the equivalent load in this table needs to be multiplied by 1.47, because an interconnect capacitance of 13fF is already added while calculating the delays.

MX21

Description

AA21 is a two–to-one multiplexer.

Verilog .... MX21 inst_name(Q,I0,I1,S);

Where, Q is the output and, I0 and I1 are the two inputs,and S is the control signal. And pins I0 and I1 offers 1 equivalent load to the circuit driving it and pin S offers equivalent load of 1.5.Each equivalent load is equal to 27.7fF.

Truth table

S	I0	I1	Q
L	L	X	L
L	H	X	H
H	X	L	L
H	X	H	H

Propagation Delays (ns)

No of Equivalent Loads		1	4	8	11	15
From Any IX input to Q	t_{PLH}	0.19	0.29	0.40	0.48	0.57
	t_{PHL}	0.23	0.33	0.43	0.50	0.59
From S to Q	t_{PLH}	0.27	0.36	0.47	0.55	0.66
	t_{PHL}	0.32	0.42	0.53	0.61	0.71

t_{PLH} : Input to output propagation delay for a rising edge on the output

t_{PHL}: Input to output propagation delay for a falling edge on the output

Also each of the equivalent load in this table needs to be multiplied by 1.47, because an interconnect capacitance of 13fF is already added while calculating the delays.

INV1

Description

INV1 is an inverter that performs *logical NOT* function.

Verilog .... INV1 inst_name(Q,A);

Where, Q is the output and, A is the input to the inverter. Pin A offers 1 equivalent load to the circuit driving it. Each equivalent load is equal to 27.7fF.

Propagation Delays (ns)

No of Equivalent Loads		1	4	8	11	15
From A to Q	t_{PLH}	0.07	0.16	0.26	0.34	0.45
	t_{PHL}	0.08	0.17	0.27	0.35	0.44

t_{PLH} : Input to output propagation delay for a rising edge on the output

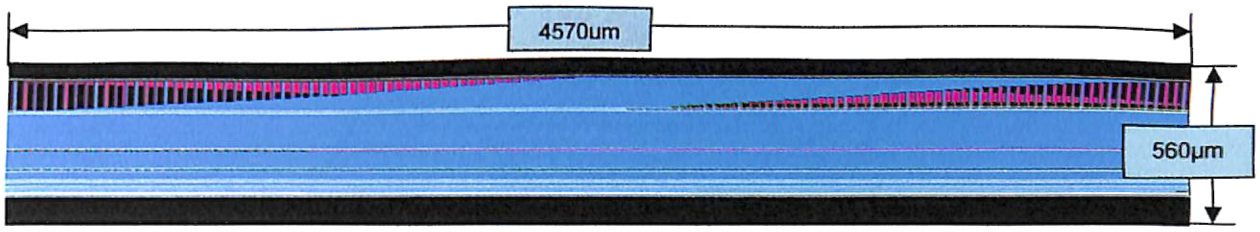
t_{PHL}: Input to output propagation delay for a falling edge on the output

Also each of the equivalent load in this table needs to be multiplied by 1.47, because an interconnect capacitance of 13fF is already added while calculating the delays.

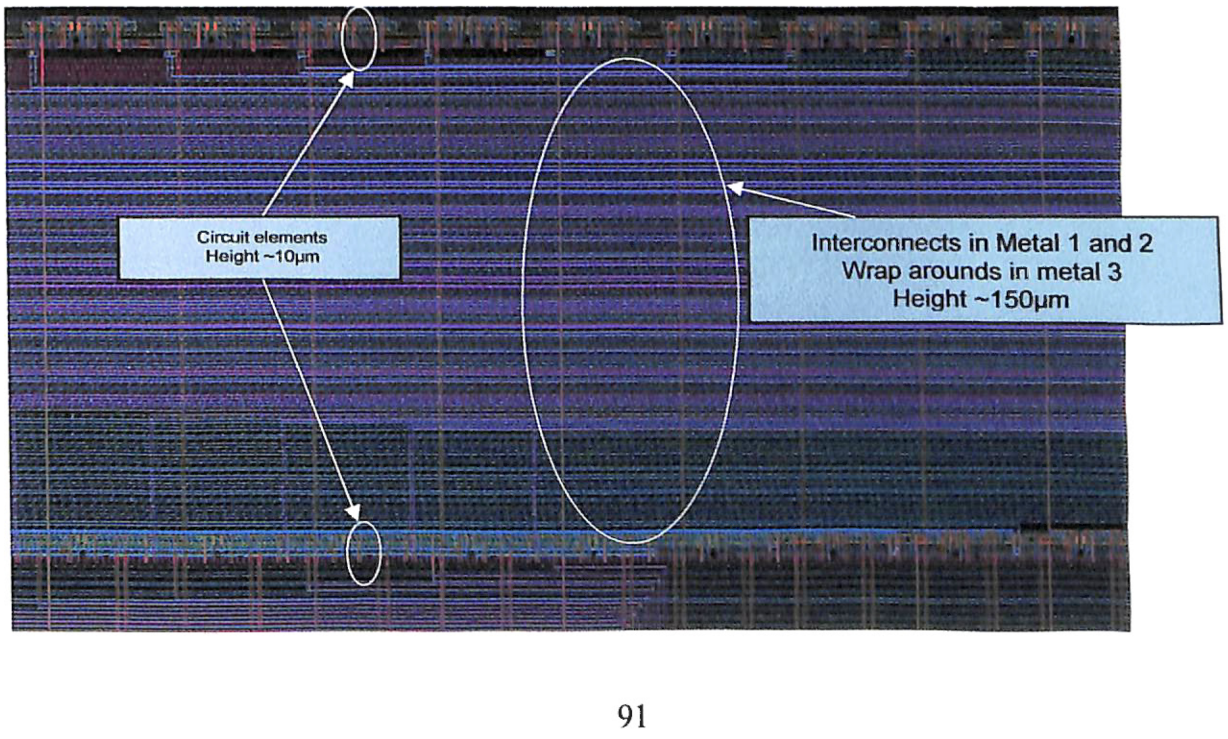
Appendix B

Layout in AMI 0.6 Micron Process

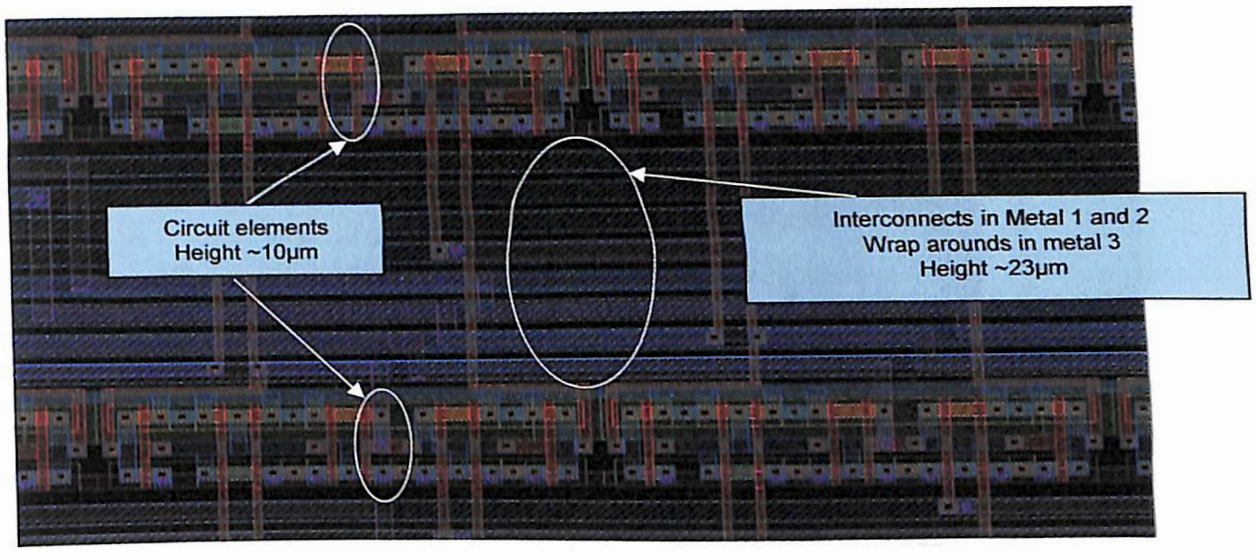
128 bit LATPE



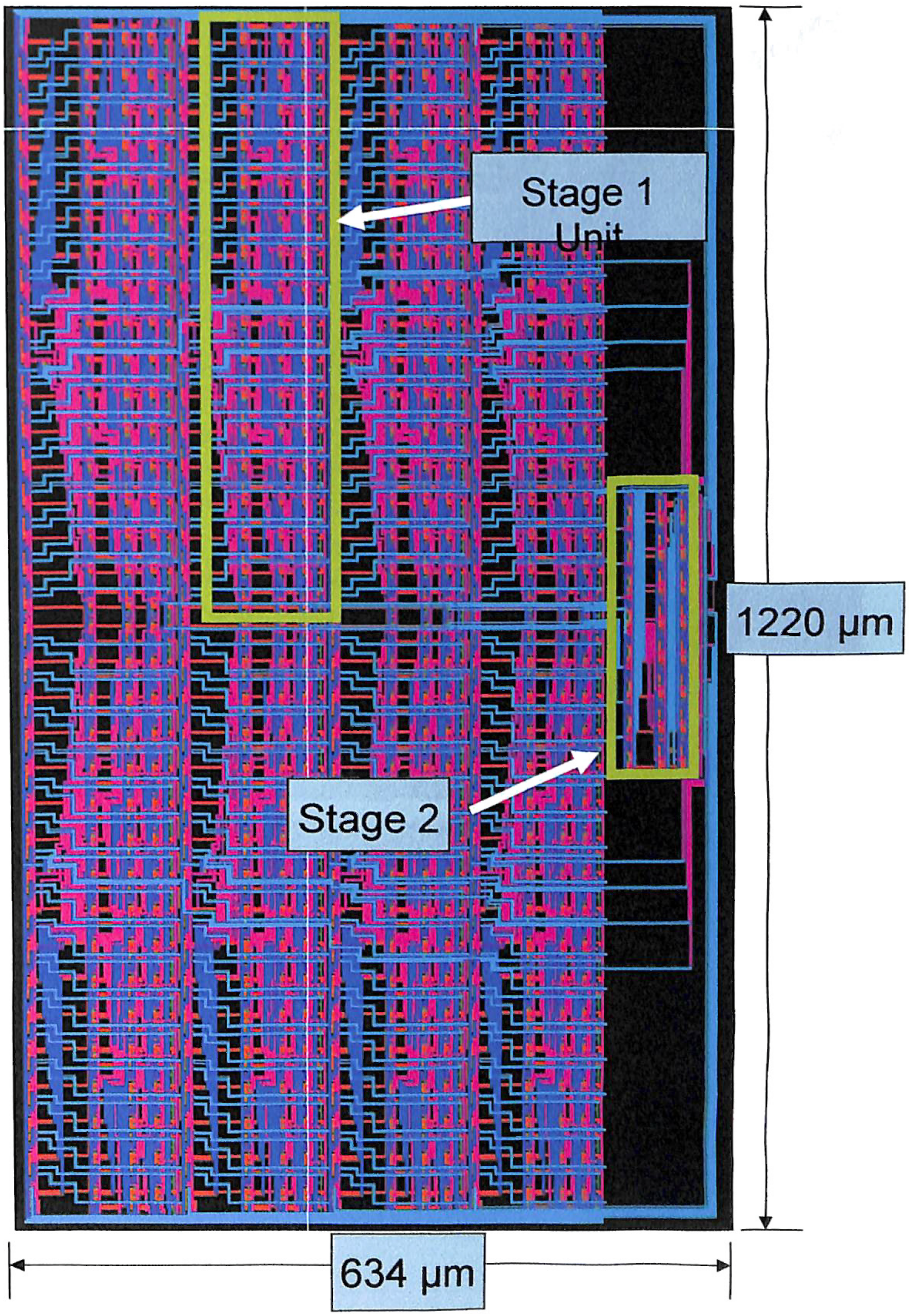
7th Stage of bit LATPE



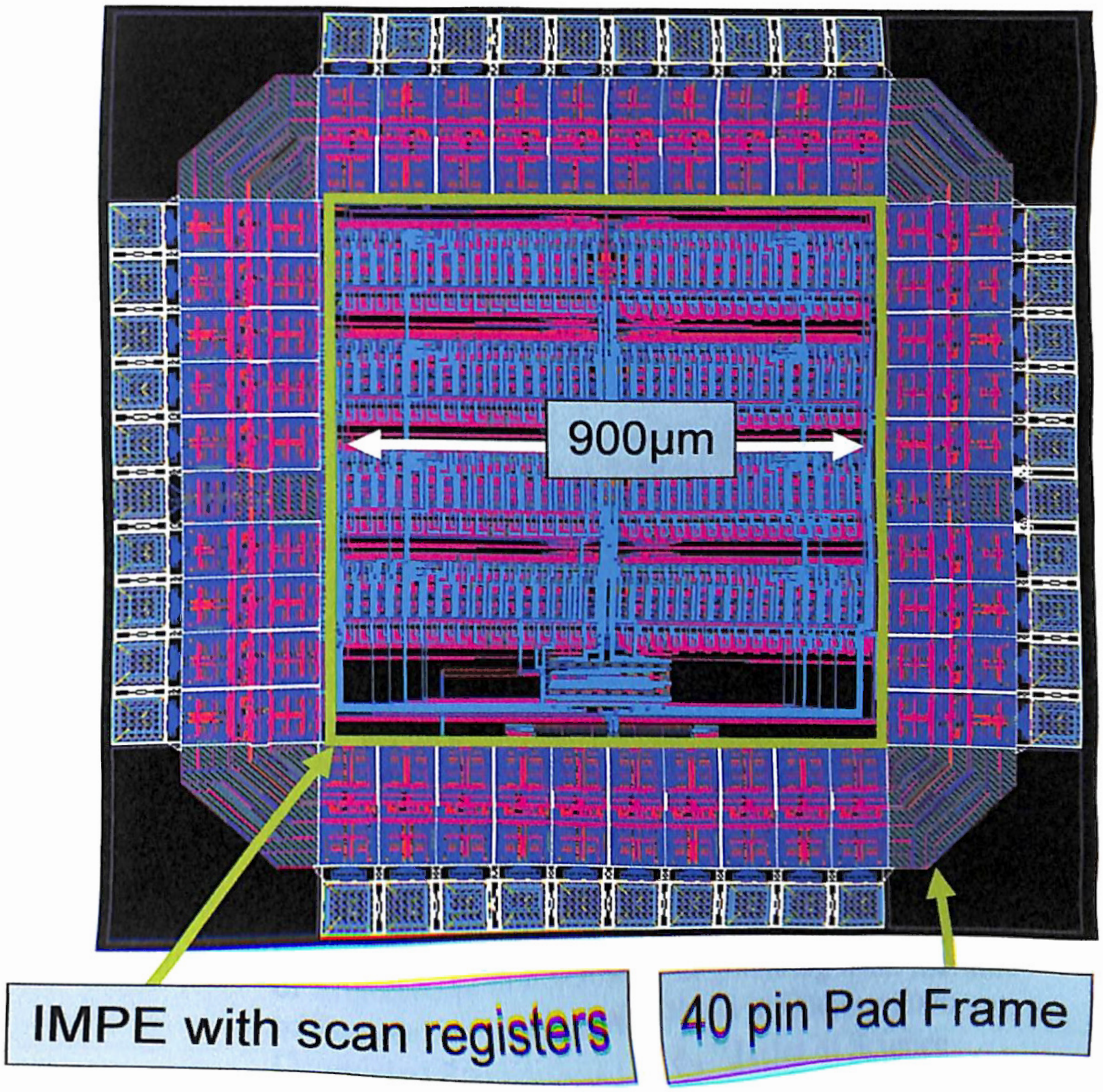
3rd Stage of 128 bit LATPE



128 bit IMPE



128 bit IMPE in 40 pin pad frame



1

VITA

KIRAN RAJ JOSHI

Candidate for the Degree of

Master of Science

Thesis: DESIGN OF 128 BIT ROUND ROBIN PRIORITY ENCODER

Major Field: Electrical Engineering

Biographical:

Personal Data: Born in Kathmandu, Nepal; son of Krishna Gopal Joshi and Bhagabati Joshi

Education: Graduated from Viswa Niketan Science Campus, Kathmandu, Nepal in 1996; received a Bachelor Degree of Engineering in Electronics Engineering from Institute of Engineering, Kathmandu, Nepal in October 2000. Completed Requirements for the Master of Science degree with a major in Electrical Engineering at Oklahoma State University in July 2004.

Experience: Research Assistant, Electrical and Computer Engineering, Oklahoma State University, 2002-2004.

Professional Associations: Phi Kappa Phi Honor Society