International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 1 Issue 7, September - 2012

LUT Optimization for Memory-Based Computation

1. M.Purna kishore
Pursuing M.Tech, NCET, Vijayawada

2. P.Srinivas
Assoc. Prof, NCET, Vijayawada

***Abstract*—Recently, we have proposed the antisymmetric product coding (APC) and odd-multiple-storage (OMS) techniques for lookup-table (LUT) design for memory-based multipliers to be used in digital signal processing applications. Each of these techniques results in the reduction of the LUT size by a factor of two. In this brief, we present a different form of APC and a modified OMS scheme, in order to combine them for efficient memory-based multiplication. The proposed combined approach provides a reduction in LUT size to one-fourth of the conventional LUT. We have also suggested a simple technique for selective sign reversal to be used in the proposed design. It is shown that the proposed LUT design for small input sizes can be used for efficient implementation of high-precision multiplication by input operand decomposition. It is found that the proposed LUT-based multiplier involves comparable area and time complexity for a word size of 8 bits, but for higher word sizes, it involves significantly less area and less multiplication time than the canonical-signed-digit (CSD)-based multipliers. For 16- and 32-bit word sizes, respectively, it offers more than 30% and 50% of saving in area–delay product over the corresponding CSD multipliers.**

***Index Terms*—*Digital signal processing (DSP) chip, lookuptable (LUT)-based computing, memory-based computing.*

I. INTRODUCTION

Digital signal processing algorithms typically require a large number of mathematical operations to be performed quickly and repetitively on a set of data. Signals are constantly converted from analog to digital, manipulated digitally, and then converted again to analog form, as diagrammed below. Many DSP applications have constraints on latency; that is, for the system to work, the DSP operation must be completed within some fixed time, and deferred processing is not viable. Digital signal processing:

In-order to reach a certain criteria memory based computation plays a vital role in dsp (digital signal processing) application.

1. FILTER DESIGNING :

Finite impulse response (FIR) digital filter is widely used as a basic tool in various signal processing and image processing applications. The order of an FIR filter primarily determines the width of the transition-band, such that the higher the filter order, the sharper is the transition between a pass-band and adjacent stop-band. Many applications in digital Communication (channel equalization, frequency channelization), speech processing (adaptive noise cancelation), seismic signal processing (noise elimination), and several other areas of signal processing require large order FIR filters. Since the number of multiply-accumulate (MAC) operations required per filter output increases linearly with the filter order, real-time implementation of these filters of large orders is a challenging task. Several attempts have, therefore, been made and continued to develop low-complexity dedicated VLSI systems for these filters.

As the scaling in silicon devices has progressed over the last four decades, semiconductor memory has become cheaper, faster and more power-efficient. According to the projections of the international technology roadmap for semiconductors (ITRS) , embedded memories will continue to have dominating presence in the system-on-chip (SoC), which may exceed 90% of total SoC content. It has also been found thatthe transistor packing density of SRAM is not only high, but also increasing much faster than the transistor density of logic devices .

1.1 BINARY MULTIPLICATION:

Multiplication in binary is similar to its decimal counterpart. Two numbers A and B can be multiplied by partial products: for each digit in B, the product of that digit in A is calculated and written on a new line, shifted leftward so that its rightmost digit lines up with the digit in B that was used. The sum of all these partial products gives the final result.

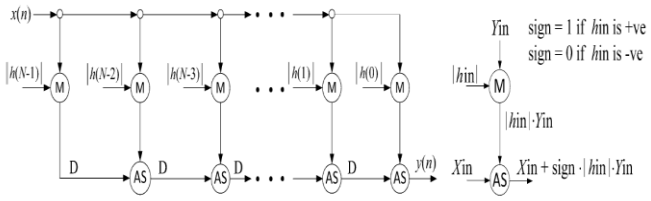
1.2 MEMORY BASED MULTIPLICATION :

The input-output relationship of an N-tap FIR filter in time-domain is given by

$$y(n) = h(0) \cdot x(n) + h(1) \cdot x(n-1) + h(2) \cdot x(n-2) + \cdots h(N-1) \cdot x(n-N+1)$$

where h(n), for n = 0,1,2,--------N-1, represent the filter coefficients x(n-i), while for i== 0,1,2,--------N-1, for x(n) , represent recent input samples y(n), and represents the current output sample. Memory-based multipliers can be implemented for signed as well as unsigned operands

1.3 FIR FILTER ARCHITECTURE



The objectives of this work are:

- Multiplying two binary numbers one number is fixed X[4:0] and another variable ‘A’
- Using APC–OMS combined LUT design for the

multiplication of W-bit fixed coefficient A with 5-bit input X.

- Number of calculations reduced and memory required is less to perform multiplication.

For 16- and 32-bit word sizes, respectively, it offers more than 30% and 50% of saving in area–delay product over the corresponding CSD multipliers.

1.4 ANTI -SYMMETRIC PRODUCT CODING:

Anti symmetric product coding is the technique used to process the multiplication based on LUT multiplication which reduces the size of conventional lut by 50 % .

The anti symmetric product coding is based on the antisymmetric coding i.e the 2's complement phenomenon which is used to reduce the LUT size by half.

For simplicity of presentation, we assume both X and A to be positive integers.2 The product words for different values of X for L = 5 are shown in Table I. It may be observed in this table that the input word X on the first column of each row is the two's complement of that on the third column of the same row. In addition, the sum of product values corresponding to these two input values on the same row is 32A. Let the product values on the second and fourth columns of a row be u and v, respectively.

Since one can write

u = [(u + v)/2 − (v − u)/2] and

v = [(u + v)/2 + (v − u)/2], for (u + v) = 32A, we can have

TABLE I
APC WORDS FOR DIFFERENT INPUT VALUES FOR $L = 5$

Input, X	product values	Input, X	product values	address $x_3'x_2'x_1'x_0'$	APC words
0 0 0 0 1	A	1 1 1 1 1	$31A$	1 1 1 1	$15A$
0 0 0 1 0	$2A$	1 1 1 1 0	$30A$	1 1 1 0	$14A$
0 0 0 1 1	$3A$	1 1 1 0 1	$29A$	1 1 0 1	$13A$
0 0 1 0 0	$4A$	1 1 1 0 0	$28A$	1 1 0 0	$12A$
0 0 1 0 1	$5A$	1 1 0 1 1	$27A$	1 0 1 1	$11A$
0 0 1 1 0	$6A$	1 1 0 1 0	$26A$	1 0 1 0	$10A$
0 0 1 1 1	$7A$	1 1 0 0 1	$25A$	1 0 0 1	$9A$
0 1 0 0 0	$8A$	1 1 0 0 0	$24A$	1 0 0 0	$8A$
0 1 0 0 1	$9A$	1 0 1 1 1	$23A$	0 1 1 1	$7A$
0 1 0 1 0	$10A$	1 0 1 1 0	$22A$	0 1 1 0	$6A$
0 1 0 1 1	$11A$	1 0 1 0 1	$21A$	0 1 0 1	$5A$
0 1 1 0 0	$12A$	1 0 1 0 0	$20A$	0 1 0 0	$4A$
0 1 1 0 1	$13A$	1 0 0 1 1	$19A$	0 0 1 1	$3A$
0 1 1 1 0	$14A$	1 0 0 1 0	$18A$	0 0 1 0	$2A$
0 1 1 1 1	$15A$	1 0 0 0 1	$17A$	0 0 0 1	A
1 0 0 0 0	$16A$	1 0 0 0 0	$16A$	0 0 0 0	0

For $X = (0\ 0\ 0\ 0\ 0)$, the encoded word to be stored is $16A$.

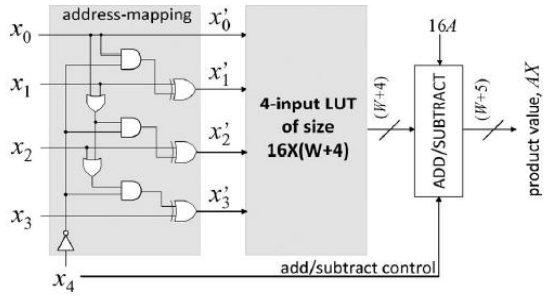
The product values on the second and fourth columns of Table I therefore have a negative mirror symmetry. This behavior of the product words can be used to reduce the LUT size,where, instead of storing u and v, only [(v − u)/2] is stored for a pair of input on a given row. The 4-bit LUT addresses and corresponding coded words are listed on the fifth and sixth columns of the table, respectively. Since the representation of the product is derived from the anti-symmetric behavior of the products, we can name it as anti-symmetric product code.

The 4-bit address X'= x3'x2'x1'x0' of the APC word is given by

X' = XL, if x4 = 1

=X'L , if x4 = 0

where XL = (x3x2x1x0) is the four less significant bits of X, and XL' is the two's complement of XL.



1.5 LUT –BASED MULTIPLICATION USING APC – OMS MODIFIED OPTIMIZATION TECHNIQUE

The APC approach, although providing a reduction in LUT size by a factor of two, incorporates substantial overhead of area and time to perform the two's complement operation of LUT output for sign modification and that of the input operand for input mapping. However, we find that when the APC approach is combined with the OMS technique, the two's complement operations could be very much simplified since the input address and LUT output could always be transformed into odd integers.

1.6 LUT COMBINED APC-OMS BASED MULTIPLICAT-ION TECHNIQUE

TABLE II
OMS-BASED DESIGN OF THE LUT OF APC WORDS FOR $L = 5$

input X' $x_3'x_2'x_1'x_0'$	product value	# of shifts	shifted input, X''	stored APC word	address $d_3d_2d_1d_0$
0 0 0 1	A	0	0 0 0 1	$P0 = A$	0 0 0 0
0 0 1 0	$2 \times A$	1			
0 1 0 0	$4 \times A$	2			
1 0 0 0	$8 \times A$	3			
0 0 1 1	$3A$	0	0 0 1 1	$P1 = 3A$	0 0 0 1
0 1 1 0	$2 \times 3A$	1			
1 1 0 0	$4 \times 3A$	2			
0 1 0 1	$5A$	0	0 1 0 1	$P2 = 5A$	0 0 1 0
1 0 1 0	$2 \times 5A$	1			
0 1 1 1	$7A$	0	0 1 1 1	$P3 = 7A$	0 0 1 1
1 1 1 0	$2 \times 7A$	1			
1 0 0 1	$9A$	0	1 0 0 1	$P4 = 9A$	0 1 0 0
1 0 1 1	$11A$	0	1 0 1 1	$P5 = 11A$	0 1 0 1
1 1 0 1	$13A$	0	1 1 0 1	$P6 = 13A$	0 1 1 0
1 1 1 1	$15A$	0	1 1 1 1	$P7 = 15A$	0 1 1 1

The proposed APC–OMS combined design of the LUT for L = 5 and for any coefficient width W is shown in Fig. 2.4. It consists of an LUT of nine words of (W + 4)-bit width, a four-to-nine-line address decoder, a barrel shifter, an address generation circuit, and a control circuit for generating the RESET signal and control word (s1s0) for the barrel shifter. The recomputed values of A × (2i + 1) are stored as Pi, for

i = 0, 1, 2, . . . , 7, at the eight consecutive locations of the memory array, as specified in Table II, while 2A is stored for input X = (00000) at LUT address "1000," as specified in Table III. The decoder takes the 4-bit address from the address generator and generates nine word-select signals, i.e., {wi, for $0 \le i \le 8$}, to select the referenced word from the

LUT. The 4-to-9-line decoder is a simple modification of 3-to-8-line decoder.

The control bits s0 and s1 to be used by the barrel shifter to produce the desired number of shifts of the LUT output are generated by the control circuit, according to the relations.

Here a simple design for sign modification of the LUT output.

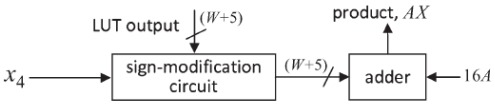
TABLE -3

input X $x_4x_3x_2x_1x_0$	product values	encoded word	stored values	# of shifts	address $d_3d_2d_1d_0$
1 0 0 0 0	$16A$	0	– – –	– –	– – –
0 0 0 0 0	0	$16A$	$2A$	3	1 0 0 0

The product values and encoded words for input words X = (00000) and (10000) are separately shown in Table III. For X = (00000), the desired encoded word 16A is derived

by 3-bit left shifts of 2A [stored at address (1000)]. For X = (10000), the APC word "0" is derived by resetting the LUT output, by an active-high RESET signal given by

ADDER/SUBSTRACTOR (ANTISYMMETRY GENERATION CIRCUIT)



The adder /sub circuit is also called as an ant symmetryl generation circuit

Based on the sign of x4,the circuit generates the anti symmetry based on the msb of x input.

$$\text{Product word} = 16A + (\text{sign value}) \times (\text{APC word})$$

2. LUT Optimation

2.1 Basic Components of LUT Optimization :

The modules contributed for combined APC-OMS based LUT optimization technique are

1 .Xin generation module (based on antisymmetric process)

2. Address generation module

3. line decoder

4. 9*(w+4) LUT

> line selector module

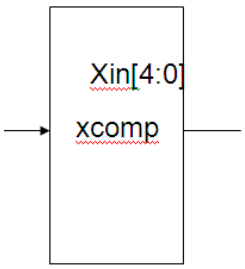
> multiplier result module

> resultant multiplier module

5. Barrel Shifter

6. Add/Substractor (Sign Determination) module

Xin generation module (based on antisymmetric process): A input of 5-bit length is given as input to this module. It used to generate antisymetric of last 4-bits (Xin(3 to 0)) when the msb of Xin i.e Xin(4) is '0' and and process the same input when the msb of Xin is '1' hence only 16 combinations will be achived for 5-bit of input as in table 1.



Block diagram

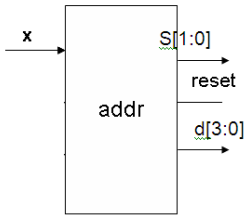
If (xin(4) = '0') then

Xcomps = Xin(4) & 2'scomplement of(Xin(3 to 0));

Else

Xcomps = Xin

2.2 Address Generation Unit :



The address generation unit generats the 4-bit address for the input given by Xin generation module the 4-bit address is named as d.

The reset output will be set when the input combination Xin = "10000";

Inorder to make the output of the barrel shifter to '0' .

$$\text{RESET} = (\overline{x_0 + x_1 + x_2 + x_3}) \cdot x_4.$$

The oupt s[1:0] are used to get the shift terminology in barrel shifter maximum of 3 shifts .

$$s_0 = \overline{x_0 + \overline{(x_1 + \overline{x_2})}}$$
$$s_1 = \overline{(x_0 + x_1)}.$$

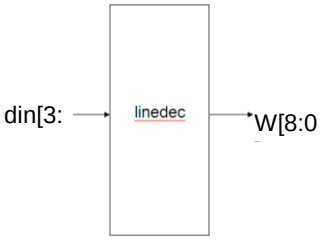
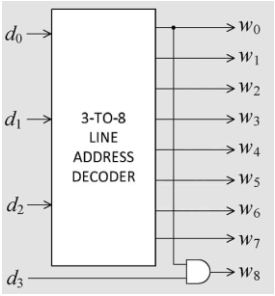


Figure: 4 to 9 Line decoder

The 4 input lines 'din' is converted into 9 output lines 'w' which is used to calculate the LUT output .

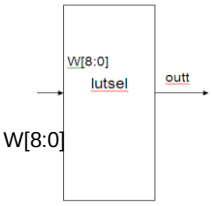
A decoder is a device which does the reverse of an encoder, undoing the encoding so that the original information can be retrieved. The same method used to encode is usually just reversed in order to decoder.



Decoding is necessary in applications such as data multiplexing, 7 segment display and memory address decoding.

A simple CPU with 8 registers may use 3-to-8 logic decoders inside the instruction decoder to select two source registers of the register file to feed into the ALU as well as the destination register to accept the output of the ALU. A typical CPU instruction decoder also includes several other things.

LUT selector



LUT selector is used to generate PVN (product value number) which is used to calculate the corresponding product value i.e (PVN X A)

The PVN is calculated depending on the W input corresponding bit set in order to generate the stored APC word i.e

The possible PVN values are

When w = 000000001 then PVN = 1
When w = 000000010 then PVN = 3
When w = 000000100 then PVN = 5
When w = 000001000 then PVN = 7

MULTIPLIER RESULT:

Multiplier result module is used to calculate multiplication of individual bits of operand and get the individual multiplication results .

Ex: 1 0 1 1 (A)
× 1 0 1 0 (B)
---
0 0 0 0 ← ress0 i.e B(0) X A
\+ 1 0 1 1 ← ress1 i.e B(1) X A
\+ 0 0 0 0 ← ress2 i.e B(2) X A
\+ 1 0 1 1 ← ress3 i.e B(3) X A

BARREL SHIFTER :

Barrel Shifter is an combinational logic circuit which is used to do any no. of shift's for one clock cycle. Depending upon the 's' the no of shift's is decided and output 'outp' is given .

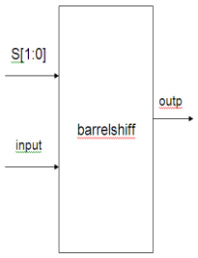


Fig:Block diagram:

For example, take a 4-bit barrel shifter, with inputs A, B, C and D. The shifter can cycle the order of the bits ABCD as DABC, CDAB, or BCDA; in this case, no bits are lost. That is, it can shift all of the outputs up to three positions to the right (and thus make any cyclic combination of A, B, C and D). The barrel shifter has a variety of applications, including being a useful component in microprocessors (alongside the ALU).

Implementation

A barrel shifter is often implemented as a cascade of parallel 2×1 multiplexers. For a 4-bit barrel shifter, an intermediate signal is used which shifts by two bits, or passes the same data, based on the value of S[1]. This signal is then shifted by another multiplexer, which is controlled by S[0]:

im = IN, if S[1] == 0
= IN << 2, if S[1] == 1
OUT = im, if S[0] == 0
= im << 1, if S[0] == 1

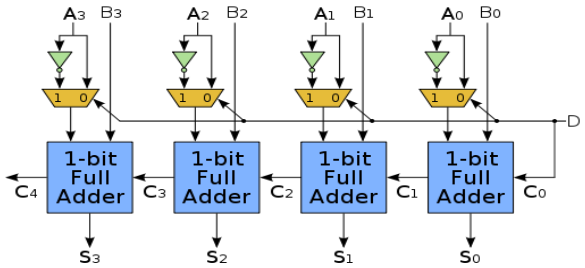
It is used to add the intermediate results to 16A to get the final output .It may make output 0 when 'clr' is high.

u = [(u + v)/2 − (v − u)/2] and
v = [(u + v)/2 + (v − u)/2], for (u + v) = 32A,

$$u = 16A - \left[\frac{v-u}{2}\right] \quad v = 16A + \left[\frac{v-u}{2}\right].$$

$$\text{Product word} = 16A + (\text{sign value}) \times (\text{APC word})$$

When xin(4) = '1' then sign value = 1
When xin(4) = '0' then sign value = 0.



4-bit_ripple_carry_adder-subtracter.svg

In digital circuits, an adder-subtractor is a circuit that is capable of adding or subtracting numbers.

This works because when D = 1 the A input to the adder is really $\overline{A}$ and the carry in is 1. Adding Bto $\overline{A}$ and 1 yields the desired subtraction of B − A.

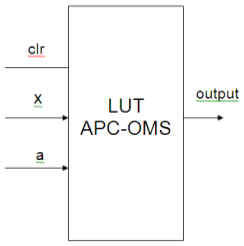
The adder-subtractor above could easily be extended to include more functions. For example, a 2-to-1 multiplexer could be introduced on each Bi that would switch between zero and Bi; this could be used (in conjunction with D = 1) to yield the two's complement of A since $-A = \overline{A} + 1$.

A further step would be to change the 2-to-1 mux on A to a 4-to-1 with the third input being zero, then replicating this on Bi thus yielding the following output functions:

0 (with the both Ai and Bi input set to zero and D = 0)
1 (with the both Ai and Bi input set to zero and D = 1)
A (with the Bi input set to zero)
B (with the Ai input set to zero)
A + 1 (with the Bi input set to zero and D = 1)
B + 1 (with the Ai input set to zero and D = 1)
A + B
A − B
B − A
$\overline{A}$ (with Ai set to invert; Bi set to zero; and D = 0)
− A (with Ai set to invert; Bi set to zero; and D = 1)
$\overline{B}$ (with Bi set to invert; Ai set to zero; and D = 0)
− B (with Bi set to invert; Ai set to zero; and D = 1)

By adding more logic in front of the adder, a single adder can be converted into much more than just an adder — an ALU.

LUT APC – OMS Optimization Top Model



The APC approach, although providing a reduction in LUT size by a factor of two, incorporates substantial overhead of area and time to perform the two's complement operation of LUT output for sign modification and that of the input operand for input mapping.

The proposed APC–OMS combined design of the LUT for L = 5 and for any coefficient width W is shown in Fig. 2.4. It consists of an LUT of nine words of (W + 4)-bit width, a four-to-nine-line address decoder, a barrel shifter, an address generation circuit, and a control circuit for generating the RESET signal and control word (s1s0) for the barrel shifter.

The recomputed values of A × (2i + 1) are stored as Pi, for i = 0, 1, 2, . . . , 7, at the eight consecutive locations of the memory array, as specified in Table II, while 2A is stored for input X = (00000) at LUT address "1000," as specified in Table III. The decoder takes the 4-bit address from the address generator and generates nine word-select signals, i.e., {wi, for 0 ≤ i ≤ 8}, to select the referenced word from the LUT.

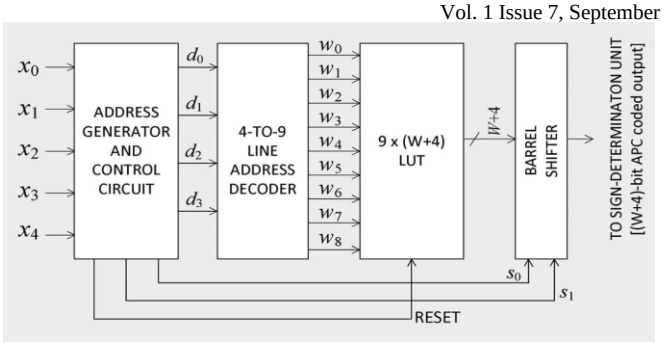
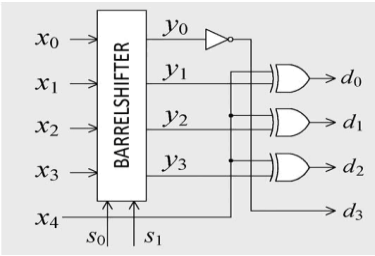


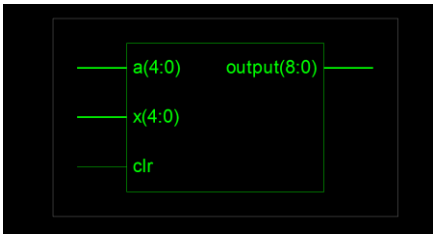
fig 2.4 lut combined apc-oms based multiplication technique



Here we observe that they will Antisymmetry in the address for the LSB 4 bits. We will get all the address from 0 to 15 for 0 to 31.Thus we reduce the memory locations required to store coefficients by half. Then we will store only odd coefficients in the look up table .

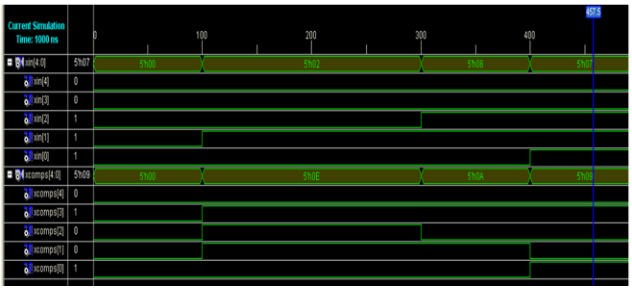
Thus we reduce the number of coefficients by half again. On total we have reduced the number coefficients by quarter.

RTL SCHEMATIC:

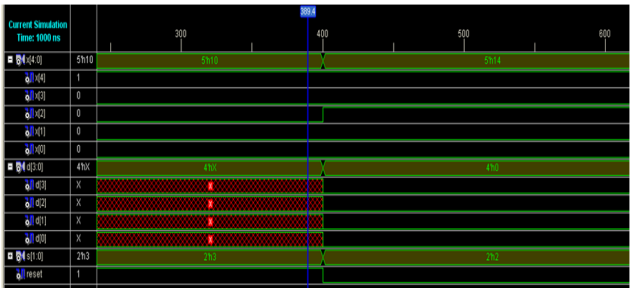


SIMULATION RESULTS:

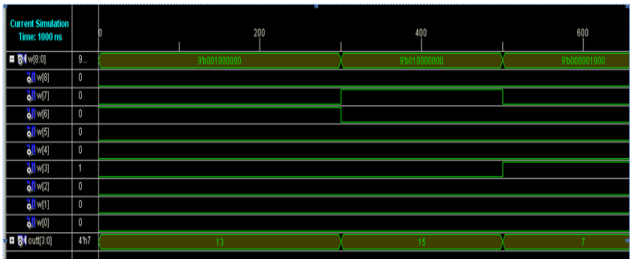
Xin Generation Module:



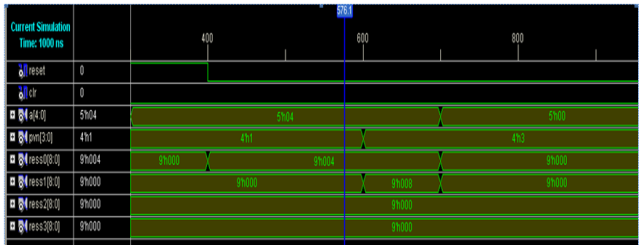
ADDRESS GENERATION MODULE:



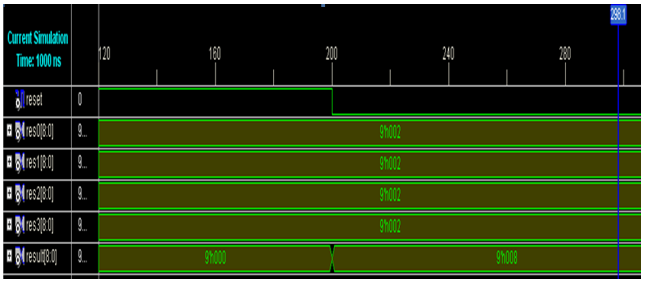
LINE SELECTOR:



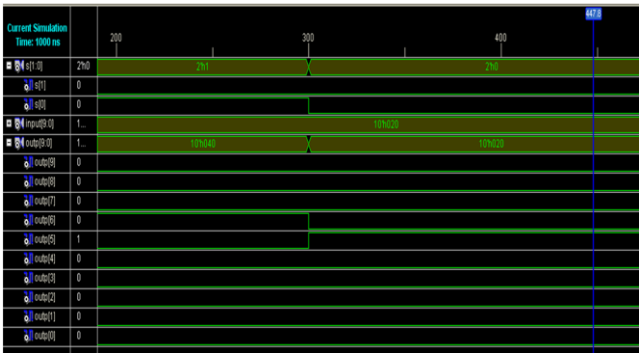
MULTIPLIER RESULT MODULE:



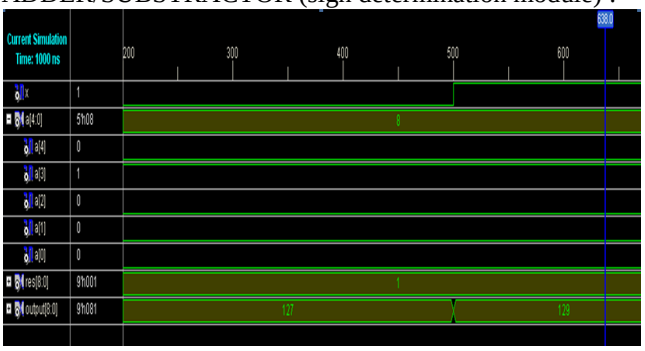
RESULTANT MULTIPLICATION MODULE:



BARREL SHIFTER :



ADDER/SUBSTRACTOR (sign determination module) :



References:

[1] LUT Optimization for Memory-Based Computation- Meher, P.K- IEEE Transactions onCircuits and Systems II: Express Briefs, April 2010**Vol** 57 , Issue: 4 pp 285 - 289

[2]MBARC: A Scalable Memory Based Reconfigurable omputing Framework for Nanoscale Devices, IEEE 2008 978-1-4244-1922-7/08 PP:77-82

[3]M. L. Bushnell and V. D. Agrawal, “Essentials of Electronic Testing for Digital, Memory, and Mixed-Signal VLSI Circuits”, Springer, 2000.

[4] L. Carloni et al., “Theory of latency-insensitive design”, IEEE TCAD, 2001.

[5] M. Tehranipoor, “Defect tolerance for molecular electronics-based nanofabrics using built-in self-test procedure”, DFT, 2005.

[6] A. Dehon et al., “Seven strategies for tolerating highly defective fabrication”, IEEE Design & Test of Computers, 2005, pp: 306-315.

[7] M. Mishra and S.C. Goldstein, “Defect Tolerance at the End of the Roadmap”, ITC, 2003, pp: 1201-1211.

[8] S.C. Goldstein et al., “NanoFabrics: Spatial Computing Using Molecular Electronics”, ISCA, 2001.

[9] R. F. Service, “Molecules get wired”, Science, vol. 294, 2001.

[10] Yong Chen et al., “Nanoscale molecular-switch crossbar circuits”, Nanotechnology 14, pp. 462-468, 2003.

[11] C. P. Collier et al., “Electronically configurable molecular-based logic gates”, Science, vol. 285, pp 391-394, 1999.

[12] A. Dehon et al., “Hybrid CMOS/nanoelectronic digital circuits: devices, architectures, and design automation”, ICCAD, 2005.

[13] M. M. Ziegler and M. R. Stan, “CMOS/Nano Co-Design for Crossbar-Based Molecular Electronic System”, IEEE Trans. on Nanotech. 2003.

[14] M. M. Ziegler and M. R. Stan, “Design and Analysis of crossbar circuits for molecular nanoelectronics”, IEEE Nano, pp. 323-327, 2002.

[15] P. Farm et al., “Nanoeda: architecture and design methodology for nano-scale elecctronic systems”, Swedish SoC Conf., 2003.