Design & verification of MS-CMOS logic based 64 bit ALU with optimal 64 bit adder

1Aashima Mishra, 2Utsav Malviya
1Mtech, 2Asst. Prof. GGITS, Jabalpur

ABSTRACT: The arithmetic logic unit (ALU) is core of a CPU in a computer. In ALU, adders play a major role not only in addition however also in performing many other basic arithmetic operations such as subtraction, multiplication, etc. most executed operation in data path is addition, which necessary a binary adder that adds two given numbers. Adders also play a vital role in lot complex computations such as multiplication, division & decimal operations. Hence, an efficient implementation of binary adder is crucial to an efficient data path. Using Monotonic-Static CMOS (MS-CMOS) logic, A 64-bit ALU has been designed with a feature size of 25nm. MS-CMOS is a low power, high speed logic family which may be seen as an intermediate logic design style in-between standard Static CMOS & Dynamic CMOS. In present work we have shown successful implementation of Critical units of ALU using MS-CMOS logic while rest of modules are implemented using static CMOS. designed ALU operates at a frequency of 1.25 GHz with a dual supply of 1.2-0.6 Volt. A modified carry-lookahead adder is used in ALU design which is considerably faster than ripple carry adder. Since Adder is main performance bottleneck in ALU design, improved adder design reduces critical path delay & improves overall performance. Also, concept of dual power supply is applied to reduce power consumption of whole circuit. Higher power supply (1.2V) is used to drive arithmetic unit components which lie in critical path whereas lower power supply (0.6V) drives logical unit. This saves nearly 75% of power consumption in logical unit & accounts for lot than 18% of total power consumed in ALU. designed ALU, performs basic arithmetic operations such as addition, subtraction, increment, decrement, transfer & logical operations such as AND, OR, XOR, NOT with logical, arithmetic & circular shifts in both direction.

I-INTRODUCTION

With ever increasing demand of high performance & low power digital processors, it is becoming lot & lot tough for traditional design approaches to meet system level specifications. Migration to newer technology method after a regular interval has only made this gap wider. This has led designers to think about novel design approaches & apply them in its design flow. This thesis explores possibility of implementing MS-CMOS logic in design of an arithmetic logic unit (ALU).

Author	Journal	Work	Outcomes
Aishwarya Verma et al [1]	IEEE 2016	They use Combination of carry-lookahead technique & MS-CMOS logic for development of 64 bit VLIW Microprocessor on 20nm & 28nm	Static powr obtain is 9 mW & Dynamic power obtain is 60 mW
Mansi Jhamb et al [2]	Elsevier, 2016	They use 64-bit MS-CMOS Carry-Lookahead Adder & Double Pass Transistor (DPL) full adder for developing a 64 multiplier	Static Power obtain for multiplier only is 3.175
Rajesh Pidugu et al [3]	IJARE EIE, 2013	Use Static Carry-Lookahead Adder & PTL Logic to develop modified 64 BIT LOW POWER ALU FOR DSP APPLICATIONS	They monitor total static power consumption of for 64 ALU 12mW

Table 1 summary of Literature work

II-PROPOSED 64-BIT MS-CMOS ALU DESIGN

The arithmetic logic unit (ALU) is a digital circuit that performs an arithmetic operation (addition, subtraction, etc.) & logic operations (Exclusive-OR, AND, etc.) in-between two numbers. ALU is a fundamental building block of central processing unit of a computer. In this chapter design methodology of MS-CMOS ALU is presented, which may operate on a 64-bit (8 byte) quad-word. Also, a detailed overview of dual supply implementation is given which is used to reduce power consumption of whole unit by exploiting *timing slack* present in various subsections.

ALU – A System's Perspective: Arithmetic Logic Unit (ALU) is part of a computer processor (CPU) that carries out arithmetic & logic operations on operands in computer instruction words. In most of processors, ALU is divided into two units, an arithmetic unit (AU) & a logic unit (LU). few processors contain lot than one AU - for example, one for *fixed-point* operations & another for *floating-point* operations. In personal computers floating point operations are sometimes done by a floating point unit on a separate chip called a numeric coprocessor.

Typically, an ALU has direct input & output access to processor controller, main memory (random access memory or RAM in a personal computer), & input/output devices. Inputs & outputs flow along an electronic path that is called a bus. input consists of an instruction word also called as machine instruction word) that contains an operation code (op-code), one or lot operands, & sometimes a format code. operation code tells ALU what operation to perform on operands. For example, two operands might be added together or compared logically. format may be combined with op code to tell about type of instruction. For example, an instruction may be a fixed-point or a floating-point instruction, among others. output consists of a result that is placed in a storage register & few flags which indicate whether operation was performed successfully or not. If it is not successful, few sort of status will be stored in a permanent place that is sometimes called machine status word.

In general, ALU includes storage places for input operands, operands on which calculations has to be performed, accumulated result which is stored in an accumulator, & shifted results. size of accumulator indicates word size of processor. flow of bits & operations performed on them in subunits of ALU is controlled by gated circuits. gates in these circuits are controlled by a sequence logic unit that uses a particular algorithm or sequence for each operation code. In arithmetic unit, multiplication & division are done by a series of adding or subtracting & shifting operations. An ALU must method numbers using same format as rest of digital circuit. There are several ways to represent negative numbers. Modern processors normally use two's complement binary number representation for negative numbers. Early computers used a wide variety of number systems, including one's complement, sign-magnitude format, & even true decimal systems. ALU's for each one of these number systems had various designs, & this is reason two's complement system is preferred over others as it necessary less complex hardware to implement addition & subtraction. design of ALU is obviously a critical part of processor & new approaches to speeding up instruction handling are continually being developed.

Design Methodology: In this section, architecture of designed ALU is discussed along with implementation of its arithmetic unit (AU) & logic unit (LU). Transistor-level schematic of its various components is also shown.

ALU Architecture: Basic job of an ALU is to perform a set of arithmetic & logical operations on of its inputs. This operation may be on both operands (inputs) or on single operand depending upon type of operation to be performed. Among many functions that it may perform, desired one is chosen by select signals. Therefore, number of select signals depends on number of various functions that a particular ALU may perform. Implementation of all functions & interconnection in-between its various components is decided by architecture of ALU. Performance of any ALU largely depends upon its architecture. Even a small flaw at architectural level, cannot be overcome by any level of circuit design expertise. Therefore, role of architecture of an ALU in its performance & efficiency is very important. architecture of designed ALU is represented in Figure 4.1

The design of ALU may be divided into two main parts- Arithmetic Unit & Logical Unit. In arithmetic unit (AU) inputs A & B are fetched from two separate registers, with A being directly fed to input of adder unit. B is connected to *second operand selection block* which computes second input to adder depending upon two lower order select signals, enabling various arithmetic operations to be performed. Logical Unit (LU) performs logical operations in-between two 64-bit input values along with left & right shift operation. There are four select signals provided which are used to select a particular operation of ALU. Complete list of ALU functions & corresponding select signals is given in Table 4.1. upper two select signals *S3 & S2* are used to select type of operation whether arithmetic, logical, shift right or shift left. Lower two select signals *S1 & S0* selects specific operation among them. For arithmetic operations input *Cin* also works as a select signal & generates two various output for *Cin* = 0 & *Cin* = 1 as shown in Table 4.1. For logical & shift operations *Cin* has no effect on output & is denoted by don't care symbol (x).

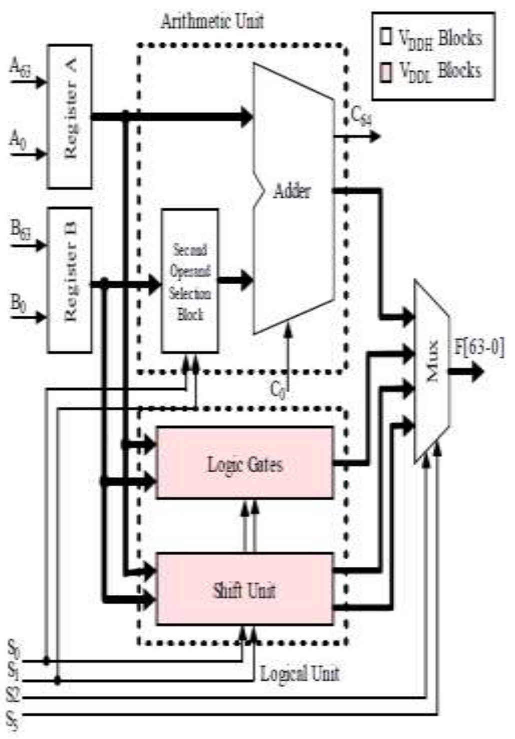


Fig 1: Architecture of designed 64-bit ALU

A 4x1 multiplexer is placed at output to choose among arithmetic, logical, shift right & shift left output. output values are selected at bit-level using two higher order select signals (*S3 & S2*). designed ALU provides a choice of eight arithmetic, four logical & six shift operation. first eight are arithmetic operations & are selected with *S3S2* = 00. next four are logical operations, selected by *S3S2* = 01. Shift right operations are selected by combination *S3S2* = 10 while *S3S2* = 11 selects shift left operation. job of multiplication & division may be performed by repeated additions or subtractions respectively. Alternatively, a separate block may be optimally designed to carry out multiplication & division since they are both, most time consuming & power consuming operations.

Arithmetic Unit

The basic elements of an arithmetic unit is parallel adder & second operand selection block. By controlling data inputs to adder, it is possible to obtain various types of arithmetic operations. In previous chapter discussion about various issues of adder design & implementation was presented. In this section we shall talk about interconnection of adder block with other blocks to perform arithmetic operations. Figure 4.2 illustrates design components of arithmetic unit [22].

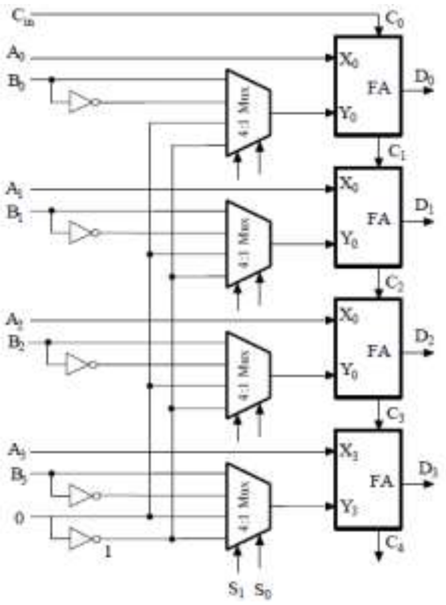


Fig 2: Design of 4-bit arithmetic unit

The above diagram is for 4-bit data & contains four full adder circuits that constitute 4-bit adder & four multiplexers in second operand selection block for choosing various operations. There are two 4-bit inputs *A & B &* a 4-bit output *D*. four inputs from *A* go directly to *X* inputs of binary adder. Each of four inputs from *B* is connected to data inputs of multiplexers. multiplexers data input also receive complement of *B*. other two data inputs are connected to logic-0 & logic-1. Logic-0 input may be connected to ground & logic-1 input to supply voltage. four multiplexers are controlled by two selection inputs, *S1 & S0*. Transistor level schematic of a two-level 4:1 multiplexer is shown in Figure 4.3.

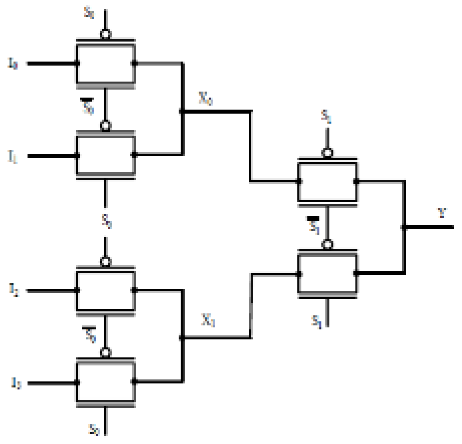


Fig 3: Transistor-level schematic of a 4:1 MUX

The input carry C_{in} is connected to carry input of full adder in least significant position. output of binary adder is calculated from following arithmetic sum:

$D = A + Y + C_{in}$

where A is 4-bit binary number at X inputs & Y is 4-bit binary number at Y inputs of binary adder. C_{in} is input carry, which may be equal to 0 or 1. Note that symbol + in equation above denotes an arithmetic plus. By controlling value of Y with two selection inputs S_1 & S_0 & making C_{in} equal to 0 ot 1, it is possible to generate eight arithmetic operations listed in Table 4.1.
When $S1S0$ = 00, value of B is applied to Y inputs of adder. If C_{in} = 0, output $D = A + B$. If C_{in} = 1, output $D = A + \bar{B} + 1$. Both cases perform add operation with or without adding input carry. For S_1S_0 = 01, complement of B is applied to Y inputs of adder. If C_{in} = 1 then $D = A + B$ +1. This produces A plus 2's complement of B, which is equivalent to $A – B$. When C_{in} = 0, then $D = A + \bar{B}$. This is equivalent to a subtract with borrow, that is, $A – B – 1$. When S_1S_0 = 10, inputs from B are neglected, & instead, all 0's are inserted into Y inputs. output becomes $D = A + 0 + C_{in}$. This gives $D = A$ when $Cin = 0$ & $D = A + 1$ when C_{in} =1. In first case we have a direct transfer from input A to output D. In second case, value of A is incremented by 1. When S_1S_0 = 11, all 1's are inserted into Y inputs of adder to produce decrement operation $D = A – 1$ when C_{in} = 0.
This is because a number with all 1's is equal to 2's complement of 1 (the 2's complement of binary 0001 is 1111). Adding a number A to 2's complement of 1 produces $D = A$ + 2's complement of $1 = A – 1$. When C_{in} = 1, then $D = A – 1 + 1 = A$, which causes a direct transfer from input A to output D. Note that operation $D = A$ is generated twice, so there are only seven distinct operations in arithmetic unit.

4.2.3 Logical Unit: In architecture of designed ALU, logical unit may be broadly divided into two parts. First block is of *logic gates* & is used for performing logical operations in-between two input operands. Second block is called *shift unit* & contains circuitry for performing various shift operations. Structure of one-bit logic gate is shown in Figure 4.4.

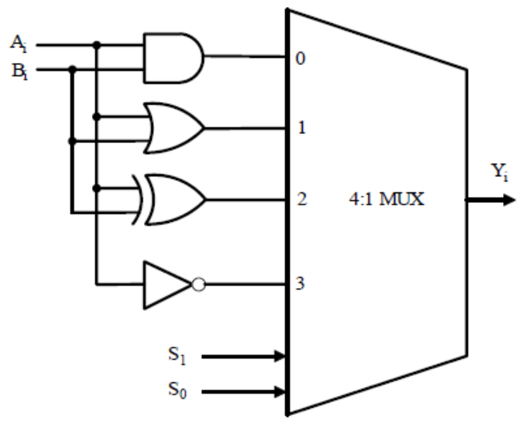


Fig 4: one-bit logic gate structure

Although there are many logic operations, most computers use only four – AND, OR, XOR, & complement – from which all others may be derived. Designed logical unit performs these four basic operations. It consists of four gates & a multiplexer as shown in Figure 4.4. Each of four logic operations is generated through a gate that performs required logic. outputs of gates are applied to data inputs of multiplexer. two selection inputs S_1 & S_0 choose one of data inputs of multiplexer & direct its value to output. diagram shows one typical stage with subscript i. For logic circuit with 64 bits, this stage is repeated 64 times for i = 0, 1, 2,…, 62, 63. selection inputs are applied to all stages. Figure 4.5 shows transistor level implementation of NOT, AND, & XOR. Note that OR operation may be derived from NAND gate with inverted inputs.

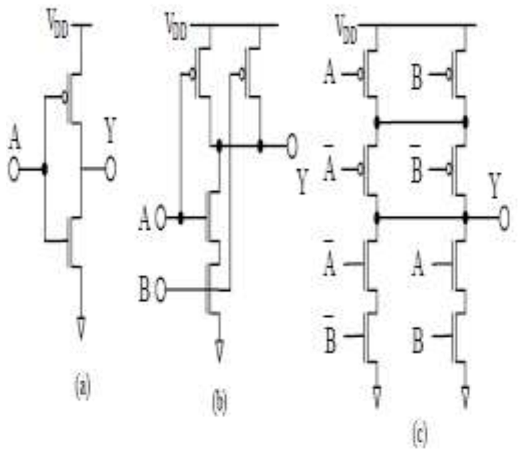
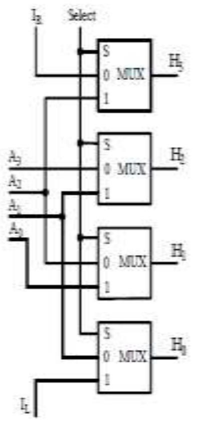


Fig 5: Transistor level static CMOS implementation of (a) NOT Gate (b) NAND Gate (c) XOR Gate
The second block of logical unit is shift unit. This unit performs all shifting operations. Shift operations are used for serial transfer of data. They are also used in conjunction with arithmetic, logic, & other data-processing operations. A possible choice for hardware implementation of a shift unit may be a bidirectional shift register with parallel load. Data may be transferred to register in parallel & then shifted to right or left. In this type of configuration, a clock pulse is needed for loading data into register, & another pulse is needed to initiate shift. In a processor unit with many registers it is lot efficient to implement shift operation with a combinational circuit. In this way content of a register that has to be shifted is first placed onto a common bus whose output is connected to combinational shifter, & shifted number is then loaded back into register. This necessary only one clock pulse for loading shifted value into register [22].
A combinational shifter may be constructed with multiplexers as shown in Figure 4.6. given diagram works

on 4-bit data inputs & is extended to 64-bit in similar fashion. This shifter unit has four data inputs, *A0* through *A3*, & four data outputs G_0 to G_3. There are two serial inputs, one for shift left (*IL*) & other for shift right (*IR*). When selection input *S* is 0, input data is shifted right (down in diagram). When *S* = 1, input data is shifted left (up in diagram). function table beside diagram shows which input goes to each output after shift. In a similar manner, a shifter with 64-bit data inputs needs 64 multiplexers.



Function Table

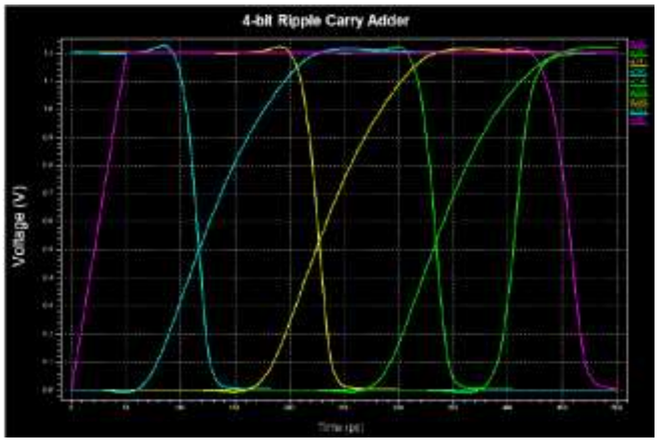
Select	Output			
S	H_3	H_2	H_1	H_0
0	I_R	A_3	A_2	A_1
1	A_2	A_1	A_0	I_L

Fig 6: A 4-bit combinational shifter with function table.

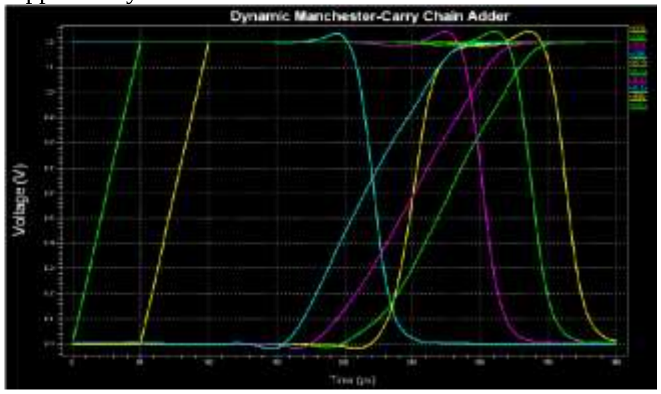
In designed 64-bit ALU, *S3S2* = 10 selects right shift operation while *S3S2* = 11 selects left shift operation. Lower

III- RESULTS

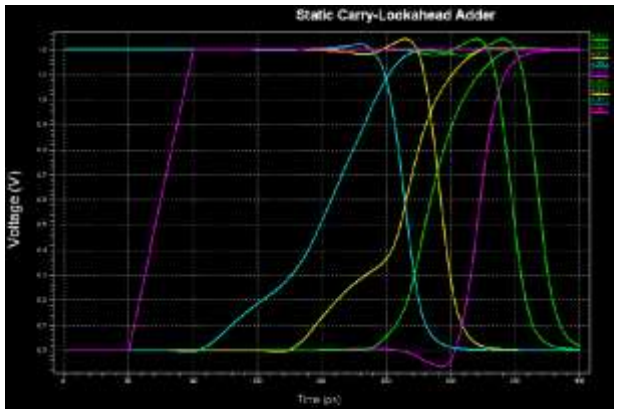
Finally output in Waveform 7 shows progressive transition of signals when performing arithmetic operation. Largest delay is found in signal *F*[63]. Note that signal *X*[63] which is logical output for 64th bit is computed much ahead of *F*[63] which denotes critical delay of design. Waveform 8 shows output for all possible combinations of operations performed by ALU. total design took around 9.5k transistors & dissipates 1.61 mW of power while operating at 1.2V-0.6V dual supply. extreme frequency of ALU is found to be 1.25 GHz.



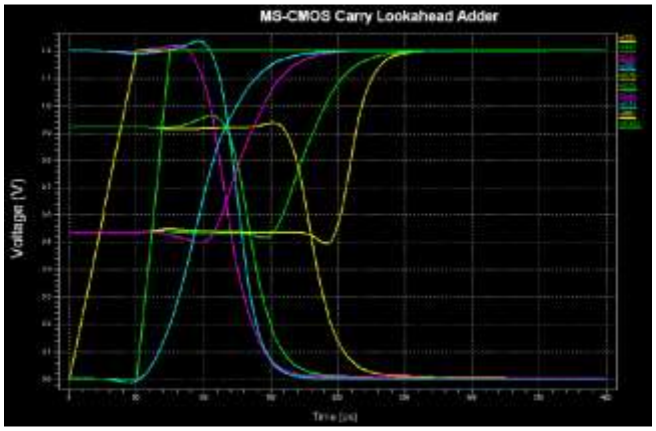
Waveform 1: Sum & carry signal transitions for 4-bit Ripple Carry Adder



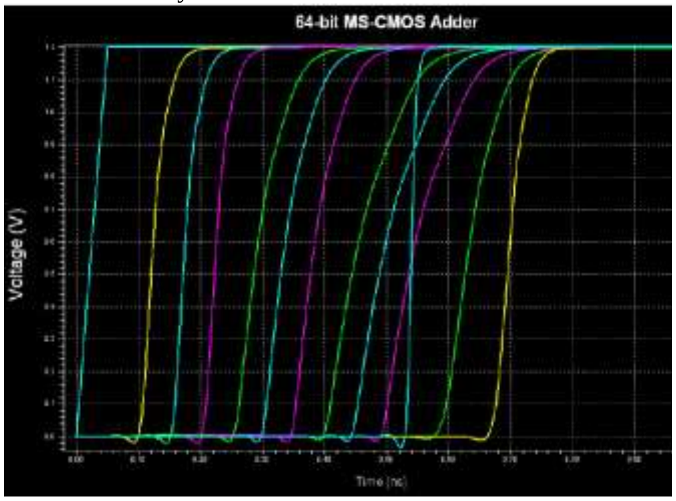
Waveform 2: Sum & carry signal transitions for 4-bit Dynamic Manchester Carry-Chian Adder
I



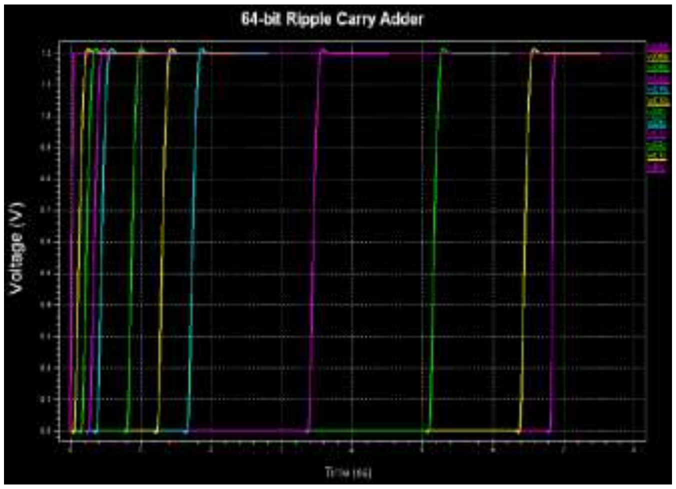
Waveform 3: Sum & carry signal transitions for 4-bit Static Carry-Lookahead Adder



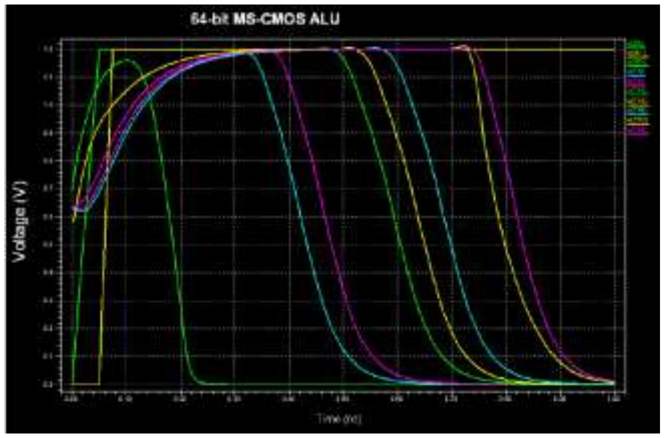
Waveform 4: Sum & carry signal transitions for 4-bit MS-CMOS Carry Lookahead Adder



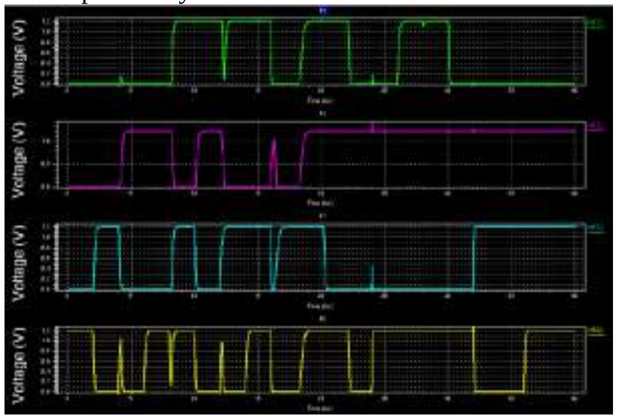
Waveform 5: Delay distribution of carry bits in 64-bit MS-CMOS Adder



Waveform 6: Delay distribution of carry bits in 64-bit Ripple Carry Adder



Waveform 7: 64-bit MS-CMOS ALU output showing critical path delay



Waveform 8: MS-CMOS ALU output for all operations

The simulation waveform show that presented design of 64 bit ALU is working correctly.

5.5 COMPARATIVE RESULTS

Author	**Outcomes**
Aishwarya Verma et al [1]	Static power obtain is 9 mW & Dynamic power obtain is 60 mW
Mansi Jhamb et al [2]	Static Power obtain for multiplier only is 3.175 mW
Rajesh Pidugu et al [3]	They monitor total static power consumption of for 64 ALU 12mW
Proposed	Proposed work is design based on modifies 64 CSA using MS-CMOS logic & 25nm transistor & Double Pass Transistor (DPL) based FA, static power monitor for 64 bit Multiplier is 2.982 mW which is better as compare with Mansi Jhamb et al [2], static power foe fu ALU design is 7.58 mW which is better as compare with Rajesh Pidugu et al [3] & Aishwarya Verma et al [1], &

	dynamic power monitor is 48.65 mW which is also better then Aishwarya Verma et al [1], The results are monitor at Tanner tool.

Table 2 : comparative results

IV-CONCLUSION

In this thesis, potential of MS-CMOS logic for use of ALU design has been highlighted. designed ALU may operate at 1.25 GHz at 1.2-0.6 volt dual-supply & may be pushed to 1.9 GHz at 2.4-1.0 volt dual-supply with roughly five times increase in power consumption. Its performance may be further improved by increasing skew ratio which is optimally set at value 2. total design took around 9.5k transistors to implement. Overall power dissipation of design was found to be 1614μW. Out of this around 52% is consumed in adder circuit, 8% in logical gates & rest 40% in other circuits especially multiplexers. Power dissipation in clock signal is negligible as compared to other units. Implementation of dual supply has saved a power of 75% in logical unit which accounts for lot than 18% saving in total power consumption of ALU.

REFERENCES

[1] Aishwarya Verma, Karan Jain, Anu Mehra1, Nidhi Gaur, Design & Implementation of 64 bit VLIW Microprocessor on 20nm & 28nm Technologies, International Conference on Information Science (ICIS), , DOI: 10.1109/INFOSCI.2016.7845329

[2] Mansi Jhamb, Garima, Himanshu Lohani, Design, implementation & performance comparison of multiplier topologies in power-delay space, GGSIPU, Sector-,Engineering Science & Technology, an International Journal 19 (2016) 355–363, Production & hosting by Elsevier

[3] Rajesh Pidugu, P. Mahesh Kannan, DESIGN OF 64 BIT LOW POWER ALU FOR DSP APPLICATIONS, ISSN(Online): 2278 – 8875, International Journal of Advanced Research in Electrical, Electronics & Instrumentation Engineering, Vol. 2, problem 4, April 2013

[4] Pragati Nagdeote1, Prof. Manisha Waje, Design of 64-bit Arithmetic Logic Unit (ALU) Based on BSIM4 Model Using Tanner, IJSART - Volume 2 problem 10 –OCTOBER 2016 ISSN [ONLINE]: 2395-1052

[5] Priyanka Yadav, Gaurav Kumar, Sumita Gupta, Design & Implementation of 4-Bit Arithmetic & Logic Unit Chip with Constraint of Power Consumption, IOSR Journal of Electronics & Communication Engineering (IOSR-JECE) e-ISSN: 2278-2834,p- ISSN: 2278-8735.Volume 9, problem 3, Ver. V (May - Jun. 2014), PP 36-43

[6] Sung-Mo Kang, Yusuf Leblebici, "CMOS Digital Integrated Circuits – study & Design" 3rd edition, TMH Edition, 2005.

[7] Kaijian Shi, David Howard, "Challenges in Sleep Transistor Design & Implementation in Low-Power Designs," *DAC 2206*, July 24-28, 2006.

[8] Yasuhisa Shimazaki, Radu Zlatanovici, & Borivoje Nikolic, "A Shared-Well Dual-Supply-Voltage 64-bit ALU," *IEEE Journal of Solid-State Circuits*, VOL. 39, NO. 3, MARCH 2004.

[9] D. Harris, M. Horowitz, "Skew-tolerant domino circuits," *IEEE J. Solid-State Circuits*, vol. 32, pp. 1702–1711, Nov. 1997.

[10] T. Thorp, G. Yee, & C. Sechen, "Design & Synthesis of Dynamic Circuits," *IEEE Trans. VLSI Systems*, vol. 11, no. 1, Feb. 2003, pp. 141-149.

[11] Ali Bastani, Charles A. Zukowski, "Monotonic static CMOS tradeoffs in sub-100nm technologies," *Proceedings of 16th ACM Great Lakes symposium on VLSI*, April 30-May 2, 2006, pp. 278 – 283.

[12] Ali Bastani, Charles A. Zukowski., "A Low-Leakage High-Speed Monotonic Static CMOS 64b Adder in a Dual Gate Oxide 65-nm CMOS Technology," *Proceedings of 7th International Symposium on Quality Electronic Design (ISQED'06)*.