International Journal of Engineering Science Invention (IJESI)
ISSN (Online): 2319 – 6734, ISSN (Print): 2319 – 6726
www.ijesi.org ||Volume 7 Issue 9 Ver I || Sep 2018 || PP 88-94

AISC Implementation of Arithmetic Mean Circuit For High Speed SRAM Applications

Atchutuni Ashalatha[1], Thota Rasmitha[2] , Subha Sri Lakshmi. Thiruveedhi[3]

[1](M.Tech – VLSI SD, ECE Department, CVR College Of Engineering, Hyderabad, India)
[2](M.Tech – VLSI SD, ECE Department, CVR College Of Engineering, Hyderabad, India)
[3](Assistant Professor, ECE Department, CVR College Of Engineering, Hyderabad, India)
Corresponding Author: Atchutuni Ashalatha

Abstract : *In the early part of digital electronic science, the synthesis of circuits are carried out by arithmetic operations has been pivotal. Conventionally, the inspection of arithmetic circuits has been oriented towards general purpose computers, which provide most important applications of digital circuits. Application specific standard product (ASSP) is a more general purpose device that is created by using ASIC tools and technologies. Concurrently, a System on Chip (SOC) is an ASIC or ASSP that acts as entire subsystem. In this paper Arithmetic Mean is designed for High speed SRAM applications. Arithmetic Mean Circuits is designed by using Enhanced Divider Data Path circuit and finally all the blocks are coded using Verilog HDL, simulated (tested) using nchdl simulator and it is synthesized and implemented on Cadence – SOC Encounter using GPDK 45nm Technology Libraries.*

Keywords : *Application Specific Standard Product (ASSP), Arithmetic Mean, Enhanced Divider, Static RAM, SOC Encounter.*

Date of Submission: 31-08-2018 Date of acceptance: 15-09-2018

I. Introduction

To make our lives accessible there are two technologies: Processors and Storage. Substantially storage technologies are restricted in two keys: Random Access Memory (RAM), Read Only Memory (ROM), as a result of rapid storage, they enable the pursuance suspect in modern computers. RAM is an artifact of hardware progression. Access time to any piece of data stored in RAM is requisite the same. RAM is typically used in computer systems for main memory or primary storage. The gap between the performance of logic and memory has widened significantly. Memory has become the critical bottleneck in most of the systems ranging from computation to communications. Conventional memory technologies have not been able to meet the need for a fast and dense memory solution in modern systems and SOC. Recently, with the rapid growth of mobile multimedia applications, the specifications of SRAM performance have become challenging, requiring high speed, low power and low voltage operation with high density. However, with conventional low power 6T-SRAM cell, high density SRAM for mobile application has been limited by relatively larger chip size. RAMs ensure an unlimited number of read and write cycles.

RAMs classified in two types: SRAMs, and DRAMs that necessitate refresh operations to retain the data stored at cell capacitors. ROMs ensure Non Volatility of stored data, exemplified by flash memories and ferroelectric RAMs (FeRAMs). In the past, as a result of device scaling, large-capacity and high-speed e-memories have become available, and have grown to the extent of dominating a chip's area and speed. Consequently, e-memories have become indispensable as main memory, caches, and program-code storage for microprocessors (MPUs), microcontrollers (MCUs), and SOCs. They even increase system speed and bandwidth and improve system reliability; at the same time, they reduce the power consumption, cost, physical size, and development time of systems [1]. SRAMs with six-transistor (6T) flip-flop cells have been widely used since the late 1980s followed by one-transistor, one-capacitor (1TlC) DRAM and flash memories in the late 1990s. Currently, 6T SRAMs and ITIC DRAMs using a vertical capacitor that increases stored charges are now competitive at 256-Mbit-Ievel capacities, and both are used as large cache memory in MPUs. As for e-ROMs, 32-Mbit flash memories have been used in MCUs, and 4-Mbit FeRAMs have been used in IC cards and RFID tags. There have been extensive efforts to study the technical and physical limits of conventional memories and to find the ways to overcome the predicted technical barriers. On the other hand, many research groups and companies tried different ways, developing new types of memories aiming longer lifetime, less technical barriers and ideal memory characteristics such as non-volatility, high density, high speed and low power

consumption, which none of the conventional memories can satisfy at the same time. SRAMs are an important part of most of the digital chips and consume a large percent of area and power of each chip [3], so decreasing the power and area of SRAMs can lead to a decrease in the overall power and area of chips. Due to quadratic relation between power and supply voltage of transistors [4], one effective and common method to reduce the power consumption is to decrease the supply voltage.

Division is the act or process of dividing. The arithmetic process is opposite of multiplication. Division is used to find out how many times a number will go into another number. For example, two goes into nine, four and a half times. This can also be written down as 9 ÷ 2 = 4.5, or 9 / 2 = 4.5 or spoken verbally as "nine over two is four and a half." The numbers in the operation have special names: Dividend ÷ divisor = quotient. Division is neither commutative nor associative. As it is helpful to look at subtraction as addition, it is helpful to look at division as multiplication of the dividend times the reciprocal of the divisor, that is $a \div b = a \times {}^{1}\!/_{b}$. When written as a product, it will obey all the properties of multiplication. Some PLDs contain SRAM blocks that can be used as part of circuits implemented in the chips. One popular chip has a number of SRAM blocks, each of which contains 4096 SRAM cells. The SRAM blocks can be configured to provide different aspect ratios, depending on the needs of the design being implemented. Aspect ratios from 512 × 8 to 4096 × 1 can be realized using a single SRAM block, and multiple blocks can be combined to form large memory arrays. To include SRAM blocks in a circuit, designers use prebuilt modules that are provided in a library as part of the CAD tools, or they write VHDL code from which synthesis tools can infer memory blocks [2].

II. Datapath & Control Path Circuits

2.1 Divider

Given two unsigned n-bit numbers X and Y, to design a circuit that produces two n-bit outputs S and A, where S is the quotient X/Y and A is the remainder. After each shift operation, we compare A with Y. If A ≥ Y, x 1 is placed in the appropriate bit position in the quotient and Y is subtracted from A. Otherwise, x 0 bit is placed in the quotient. This algorithm is described using pseudo-code below [2].

```
R = 0 ;
for i = 0 to n – 1 do
    Left-shift R‖A ;
    if R ≥ B then
        q_I = 1 ;
        R = R – B ;
    else
        q_I = 0 ;
    end if ;
end for ;
```

Fig 1 Pseudo-Code for Divider

a. Datapath Circuit

n-bit shift registers that shift right to left for A, X, and Q. An n-bit register is needed for B, and a subtractor is needed to produce X − B. We can use an adder module in which the carry-in is set to 1 and B is complemented. The carry-out, cout, of this module has the value 1 if the condition X ≥ B is true. Hence the carry-out can be connected to the serial input of the shift register that holds Q, so that it is shifted into Q in state S3. Since R is loaded with 0 in state S1 and from the outputs of the adder in state S3, a multiplexer is needed for the parallel data inputs on A. The Datapath circuit is depicted in Fig1. Note that the down-counter needed to implement C and the NOR gate that outputs a 1 when C= 0 are not shown in the fig.

b. Control Circuit

An ASM chart that shows only the control signals needed for the divider is given in Figure 1(b). In state S3 the value of cout determines whether or not the sum output of the adder is loaded into A. The shift enable on S is asserted in state S3. It's not necessary to specify whether 1 or 0 is loaded into S, because cout is connected to S's serial input.

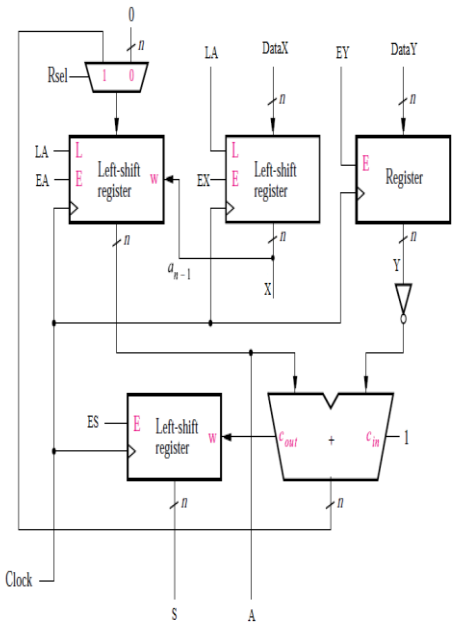
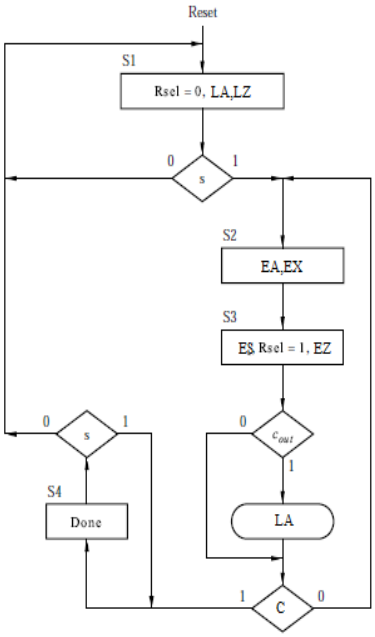


Fig 1(a) Datapath of Divider



1(b) Controlpath of Divider

2.1.1 Enhanced Divider

Using the ASM chart in Fig1(b) causes the circuit to loop through states *S*2 and*S*3 for 2*n* clock cycles. If these states can be merged into a single state, then the number of clock cycles needed can be reduced to *n*. In state *S*3, if *cout* = 1, load the sum output (result of the subtraction) from the adder into *A*, and (assuming *z* = 0) change to state *S*2.In state *S*2 we then shift *A* (and *X*) to the left. To combine *S*2 and *S*3 into a new state, called*S*2, to able to place the sum into the left-most bits of *A* while at the same time shifting the MSB of *X* into the LSB of A[6]. This step can be accomplished by using a separate flip-flop for the LSB of *A*. Let the output of this flip-flop be called *rr*0. It is initialized to 0 when *s* = 0 in state *S*1. Otherwise, the flip-flop is loaded from the MSB of *A*. In state *S*2,if *cout* = 0, *R* is shifted left and *rr*0 is shifted into *A*. But if *cout* = 1, *A* is loaded in parallel from the sum outputs of the adder.

An ASM chart that shows the values of the required control signals for the enhanced divider is depicted in Fig1 (b). The signal EA0 is used in conjunction with the flip-flop that has the output *rr*0. When EA0 = 0, the value 0 is loaded into the flip-flop. When EA0 is set to 1, the MSB of shift register *X* is loaded into the flip-flop. In state *S*1, if *s* = 0, then *LA* is asserted to initialize *A* to 0. Registers *X* and *Y* can be loaded with data from external inputs. When *s* changes to 1, the machine makes a transition to state *S*2 and at the same time shifts *A*||*A*0||*X* to the left. In state *S*2, if *cout* = 1, then *A* is loaded in parallel from the sum outputs of the adder. At the same time, *A0*||*X* is shifted left (*rr*0 is not shifted into *A* in this case). If *cout* = 0, then *R*||*R*0||*A* is shifted left. The ASM chart shows how the parallel-load and enable inputs on the registers have to be controlled to achieve the desired operation. The Datapath circuit for the enhanced divider is illustrated in Fig2 (a). As discussed for Fig2(b), the digits of the quotient S are shifted into register *X*. Note that one of the *n*-bit data inputs on the adder module is composed of the $n-1$ least-significant bits in register *A* concatenated with bit *rr*0 on the right.

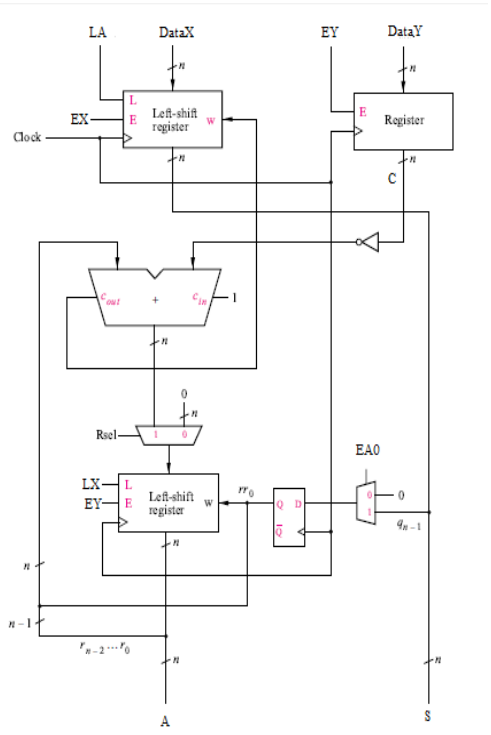
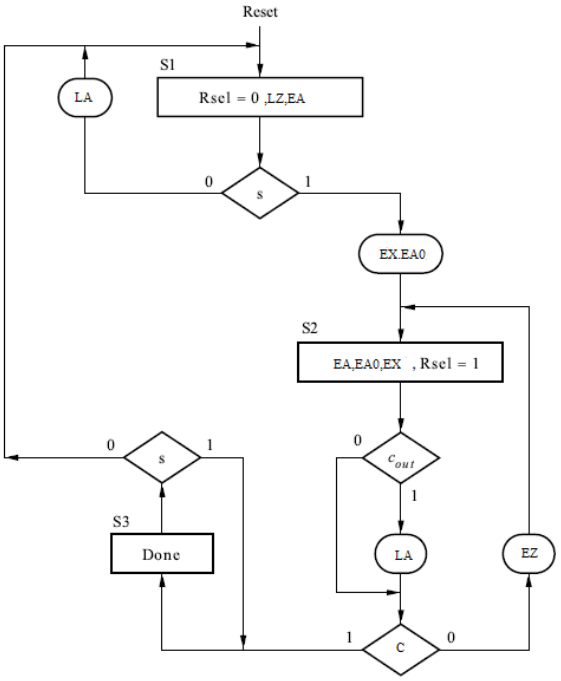


Fig 2(a) Datapath of Enhanced Divider



2(b) Control Path of Enhanced Divider

2.2 Arithmetic Mean

Assume that n-bit numbers are stored in a set of registers A0... Ak−1. To design a circuit that computes the mean M of the numbers in the registers. Each iteration of the loop adds the contents of one of the registers, denoted Ai, to a Sum variable. After the sum is computed, A is obtained as Sum/k. The Datapath circuit for this task is more complex than in our previous examples. It is depicted in Fig3 (a). A register with an enable input to hold *Sum*. For simplicity, assume that the sum can be represented in *n* bits without overflowing. A multiplexer is required on the data inputs on the *Sum* register, to select 0 in state *S*1 and the sum outputs of an adder in state *S*2. The *Sum* register provides one of the data inputs to the adder. The other input has to be selected from the data outputs of one of the *k* registers [4],[5]. One way to select among the registers is to connect them to the data inputs of a *k*-to-1 multiplexer that is connected to the adder. The select lines on the multiplexer can be controlled by the counter *C*. The operation is based on $k = 4$, but the same circuit structure can be used for larger values of *k*. Note that the enable inputs on the registers *R*0 through *R*3 are connected to the outputs of a 2-to-4 decoder that has the two-bit input *RAdd*, which stands for "register address." The decoder enable input is driven by the *ER* signal. All registers are loaded from the same input lines, *Data*. Since $k = 4$, we could perform the division operation simply by shifting *Sum* two bits to the right, which can be done in one clock cycle with a shift register that shifts by two digits. To obtain a more general circuit that works for any value of *k*, the divider circuit designed in section 2.

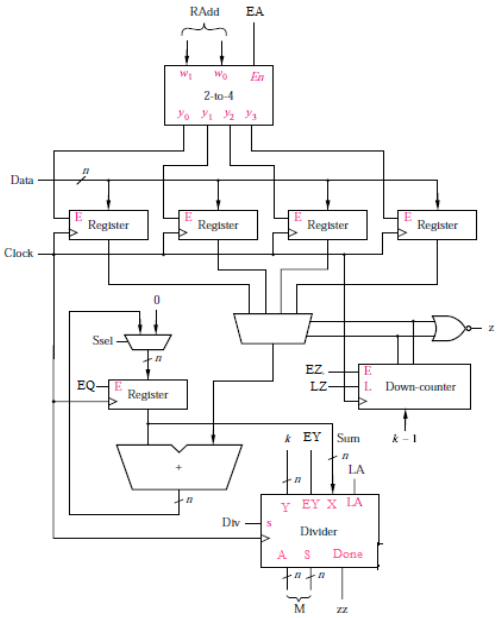
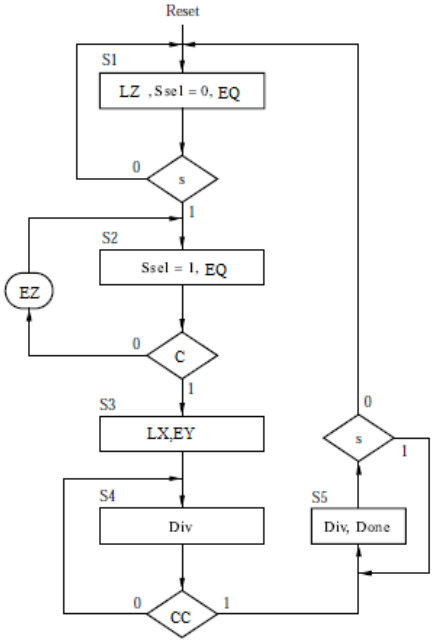


Fig 3(a) Datapath for Arithmetic Mean



3(b) ASM chart for Control path of Arithmetic Mean

Figure 3(b) gives an ASM chart for the FSM needed to control the circuit in Figure 3(a). While in state *S*1, data can be loaded into registers *A*0, . . . , *Ak*−1. But no control signals have to be asserted for this purpose, because the registers are loaded under control of the *EA* and *A Add* inputs, as discussed above. When *s* = 1, the FSM changes to state *S*2, where it asserts the enable *EQ* on the *Sum* register and allows *Z* to decrement. When the counter reaches0 (*c* = 1), the machine enters state *S*3, where it asserts the *LX* and *EY* signals to load the *Sum* and *k* into the *X* and *Y* inputs of the divider circuit, respectively. The FSM then enters state *S*4 and asserts the *Div* signal to start the division operation. When it is finished, the divider circuit sets *cc* = 1, and the FSM moves to state *S*5. The mean *M* appears on the S and *A* outputs of the divider circuit. The *Div* signal must still be asserted in state *S*5 to prevent the divider circuit from reinitializing its registers. Note that in the ASM chart in Figure 3(b), only one state is shown for computing *M* = *Sum/k*, but in Figure 3(a), states *S*3 and *S*4 are used for this purpose. It is possible to combine states *S*3 and *S*4.

The down-counter *C* can be used to enable each tri-state buffer at the proper time (when the FSM is in state *S*2), by connecting a 2-to-4 decoder to the outputs of the counter and using one output of the decoder to enable each tri-state buffer. For large values of *k*, it is preferable to use SRAM block with *k* rows and *n* columns, instead of using *k* registers. Predefined modules that represent SRAM blocks are usually provided by CAD tools. If the circuit being designed is to be implemented in a custom chip, then the CAD tools ensure that the desired SRAM block is included on the chip [7].

2.3 Static Random Access Memory (SRAM)

Some PLDs include SRAM blocks that can be configured to implement various numbers of rows and columns. Figure 4 gives a schematic diagram for the arithmetic mean circuit, using the parameters *k* = 16 and *n* = 8.. Four of the graphical symbols in the schematic represent subcircuits described using Verilog HDL code, namely *downcnt*, *re*gne, *divider*, and *mean cntl*. The schematic also includes a multiplexer connected to the *Sum* register, an adder, and a NOR gate that detects when the counter *C* reaches0. The outputs of the counter provide the address inputs to the SRAM block, called *MReg*. The SRAM block has16 rows and eight columns. In Fig3(a) a decoder controls the loading of data into each of the *k* registers. To read the data from the registers, the counter *C* is used. To keep the schematic in Fig4 simple, we have included the
counter to read data from the SRAM block, but the issue of writing data into the SRAM block is ignored. It is possible to modify the *meancntl* code to allow the counter *C* to address the SRAM block for loading the initial data, but we will not pursue this issue here [1],[3]. For simulation purposes we can use a feature of the CAD system that allows initial data to be stored in the SRAM block. To store 0 in *R*0 (row 0 of the SRAM block);1 in *R*1, . . . ; and 15 in *R*15. The results of a timing simulation for the circuit implemented in an FPGA chip are shown in Fig4(a). Only a part of the simulation, from the point where *C* = 5, is shown in the figure. At this point the *meancntl* FSM is in state *S*2, and the *Sum* is being accumulated. When *C* reaches0, *Sum* has the correct value, which is 0 + 1 + 2 + ⋅ ⋅ ⋅ + 15 = 120 = *(*78*)*16. The FSM changes to state *S*3 for one clock cycle and

then remain sin state *S*4 until the division operation is complete. The correct result, Q= 7 and *R* = 8, is obtained when the FSM changes to state *S*5.

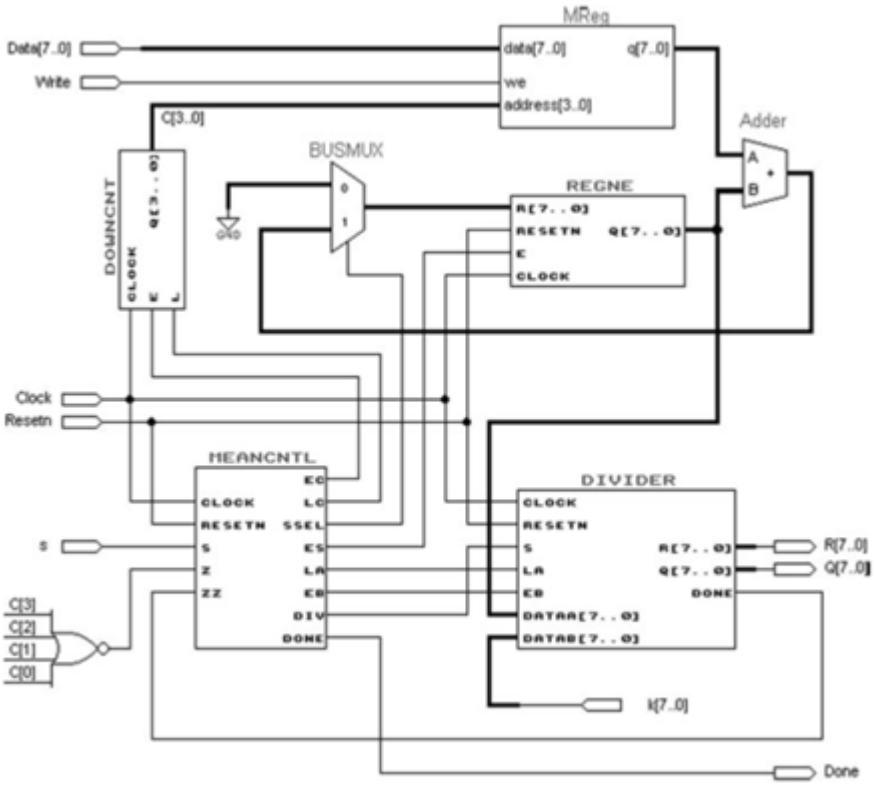


Fig 4 SRAM Mean Circuit

Tools Required

Cadence GPDK	- 45nm Technologies
Language used	- Verilog HDL
Simulation Tool	- Nchdl Simulator
Synthesis	- RTL Complier
Physical Design Tool	- SOC Encounter

III. Simulation Results

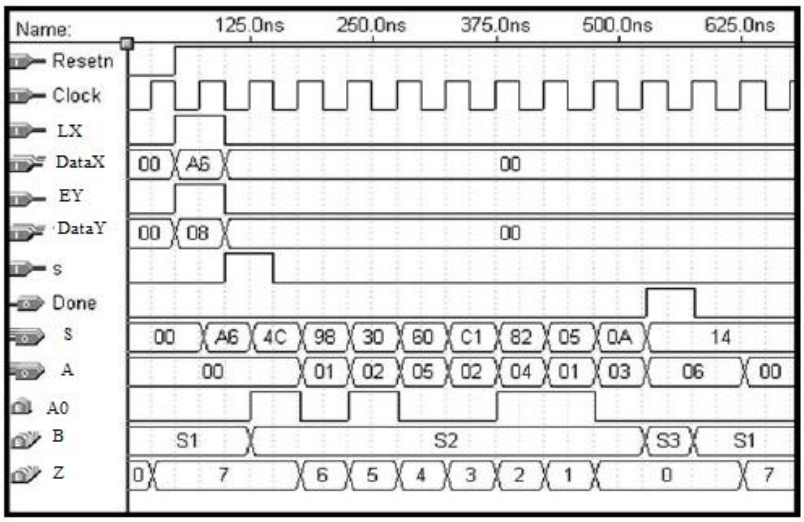


Fig 4(a) Simulation Waveform of Divider Circuit

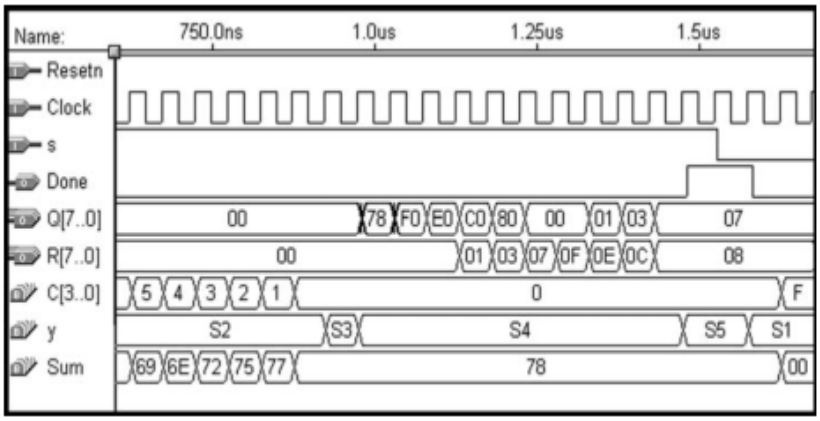


Fig 4(b) Simulation Waveform of SRAM Circuit

IV. Conclusion

Logic circuits are used to built computer hardware as well as many other types of products. All such products are broadly classified as digital hardware. The technology used to build digital hardware has evolved dramatically over the past four decades. Until the 1960's logic circuits were constructed with bulky components, such as transistors and resistors that came as individual parts. The advent of integrated circuits made it possible to place a number of transistors, and thus an entire circuit, on a single chip. Further it is named as System on Chip (SoC). In this paper, Arithmetic Mean circuit is designed for High Speed SRAM applications. Nearly every field in mathematics and science uses the arithmetic mean. Many of the most common metrics in economics, such as per capita income and per capita gross domestic product (GDP), are calculated using arithmetic mean. Apart from this Arithmetic mean is used to calculate Time, Speed and Distance in any real time application.

References

[1]. C. T. Chuang, et al., "High-performance S challenges and techniques," Proc. IEEE I Technology, Design, and Testing (MTDT), 2007, pp 4-11.Note that the journal title, volume number and issue number are set in italics.
[2]. Stephen Brown 3rd edition fundamental of digital logic with vhdl design
[3]. D. Burnett et al. "Implications of fundam high-density SRAM and logic circuits " Digest Tech. Papers Symp. VLSI Technology 1994 pp. 15-16.
[4]. R. Omondy Computer Arithmetic System: Algorithms Architectures and Implementations Prentice Hall 1994.
[5]. K. Koren Computer Arithmetic Algorithms Prentice Hall 1991.
[6]. N. Takagi S. Kadowaki K Takagi "A hardware algorithm for integer division" Computer Arithmetic ARITH-17 2005 17th IEEE Symposium on Computer Arithmetic pp. 140-146 27-29 June 2005.
[7]. S.F. Oberman M. Flynn "Division algorithms and implementations" in IEEE Transactions on Computer Arithmetic CA USA:Comput. Syst. Lab. Stanford Univ vol. 46 no. 8 pp. 833-854 august 2002.

Atchutuni Ashalatha"AISC Implementation of Arithmetic Mean Circuit For High Speed SRAM Applications." International Journal of Engineering Science Invention(IJESI), vol. 7, no. 9, 2018, pp. 88-94