Microprocessor Design

Prepared By

Abdelrahman Khaled EL-sayed

Supervised by

Dr. Eslam Tawfik

Table of Contents

Table of figures

1. Introduction

1.1 specification

Simple as Possible (SAP) computers in general were designed to introduce beginners to some of the crucial ideas behind computer operations. SAP computers are classified into stages, each stage more evolved and considering more advanced concepts in computer architecture than the previous.

The Simple-As-Possible (SAP)-1 computer is a very basic model of a microprocessor explained by Albert Paul Malvino. The SAP-1 computer is the first stage in this evolution. Its primary purpose is to develop a basic understanding of how a computer works, interacts with memory and other parts of the system like input and output. The instruction set is very limited and is simple.

1.2 The features in sap-1 computer are

- W bus 8 bit.
- 16 bytes memory.
- Registers - are accumulator and B-register each of 8 bits.
- Program counter.
- Memory Address Register.
- Adder/Subtracter.
- Control Unit
- Simple Output

In this project, we implement a SAP-1 computer using Verilog , write a testbench for each block and check functionality of the design ,also analysis the timing reports , Fitter Resource Report and get the netlist viewer .

2. Architecture

Figure 1-1 shows the architecture of SAP-1, a bus-organized computer. All register outputs to the 8 bit W-bus are three states; this allows orderly transfer of data. The layout of Fig. 1-1 emphasizes the registers used in SAP- 1.

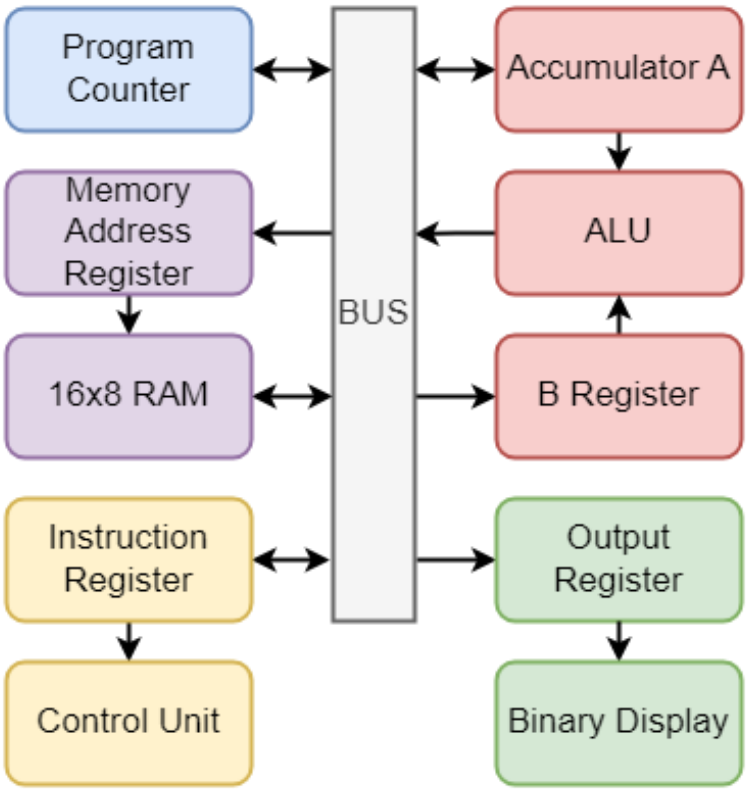


Figure1 -0-1shows the architecture of SAP-1

2.1 program counter

The program is stored at the beginning of the memory with the first instruction at binary address 0000, the second instruction at address 0001, the third at address 0010, and so on. sometimes called a pointer; it points to an address in memory where something important is being stored.

The program counter, which is part of the control unit, counts from 0000 to 1111. The program counter is reset to 0000 before each computer run. The program counter is then incremented to get 0001. After the first instruction is fetched and executed, the program counter sends address 0001 to the memory. Again the program counter is incremented. After the second instruction is fetched and executed, the program counter sends address 0010 to the memory.

2.1.1 RTL viewer

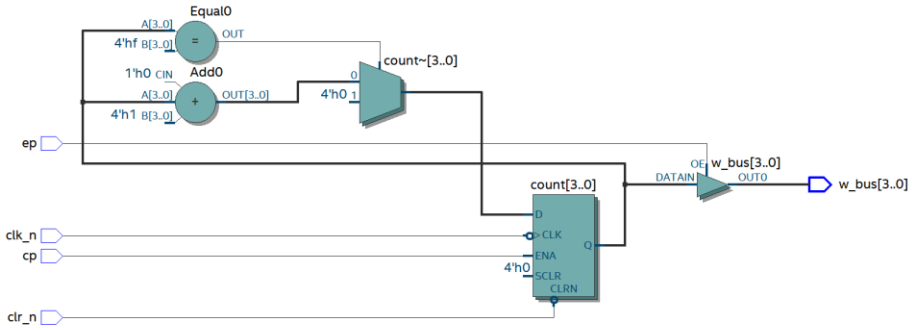


Figure 1-0-2 program counter RTL

2.1.2 program counter simulation

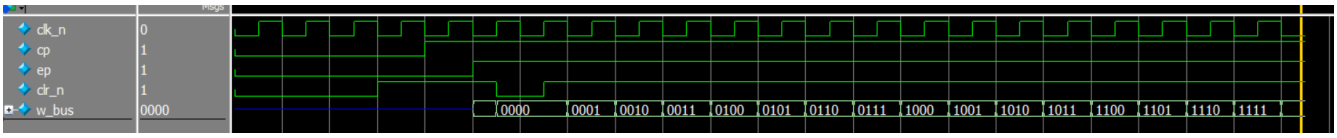


Figure 1-0-3 program counter simulation

2.2 memory address register

The MAR stores the 4-bit address of data or instruction which are placed in memory. When the SAP-1 is running, the 4-bit address is gotten from the Program Counter through the W-bus and then stored. This stored address is sent to the RAM where data or instructions are read from.

2.2.1 RTL viewer

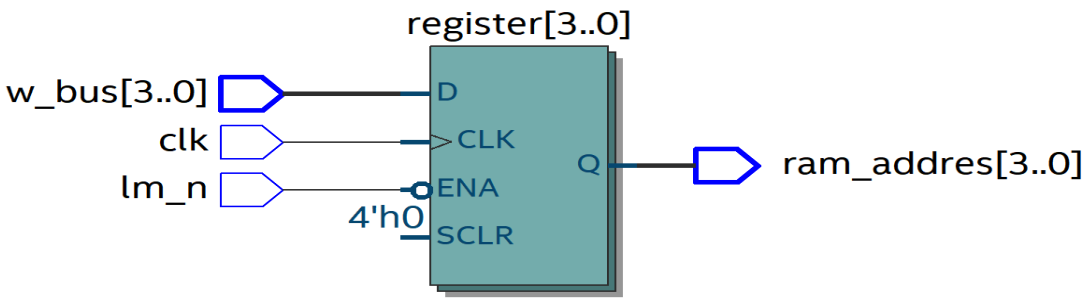


Figure 1-0-4 memory address RTL

2.2.2 simulation

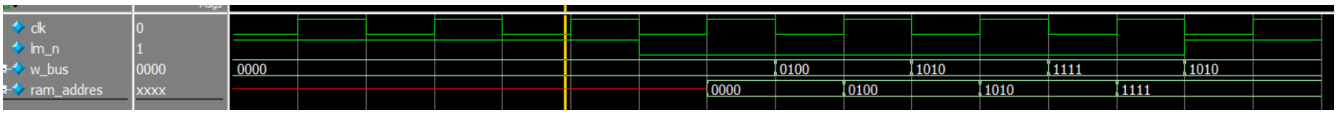


Figure 1-0-5 memory address register simulation

2.3 RAM

The SAP-1 makes use of a 16 x 8 RAM (16 memory locations each storing 8 bits of data). The RAM can be programmed by means of the address and data switches allowing you to write to the memory before a computer run. During a computer run, the RAM receives its 4-bit address from the MAR and read operation is performed. In this way the instruction or data word stored in the RAM is placed on the W bus for use in some other part of the computer. The ram appear in the figure 1-16.

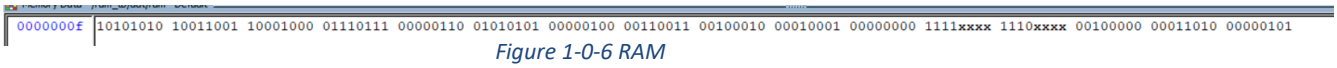


Figure 1-0-6 RAM

2.3.1 RTL viewer

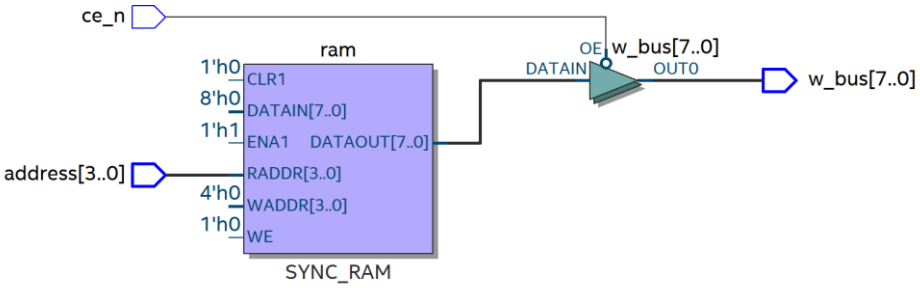


Figure 1-0-7 RAM RTL

2.3.2 simulation

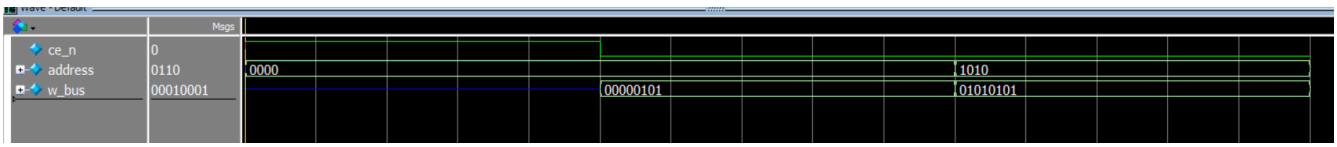


Figure 1-0-8 RAM simulation

2.4 instruction register

The instruction register is part of the control unit. The contents of the instruction register are split into two nibbles. The upper nibble is a two-state output that goes directly to the block labeled "Controller sequencer". The lower nibble goes directly to the bus . the two nibbles indicate the op code and data interface.

2.4.1 RTL viewer

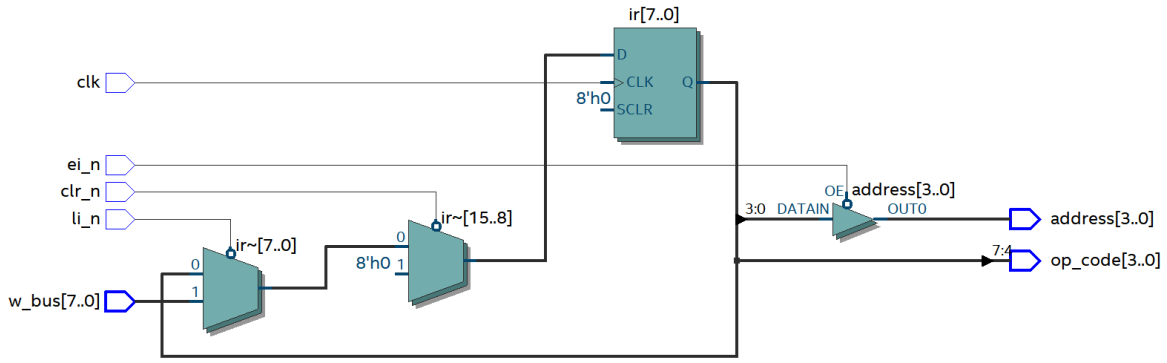


Figure 1-0-9 instruction register RTL

2.4.2 simulation

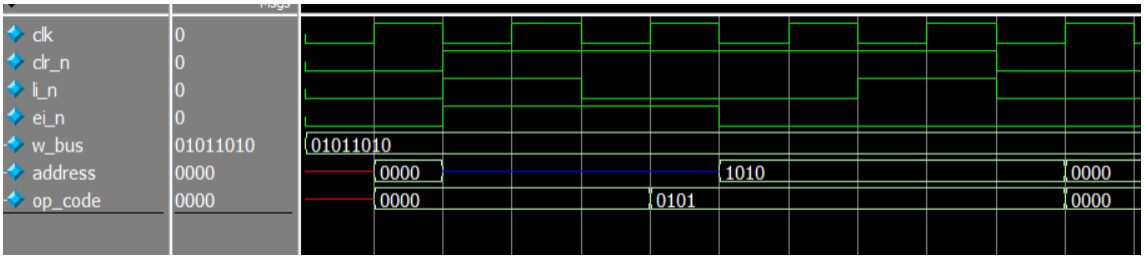


Figure 1-0-10 instruction register simulation

2.5 Accumulator

The accumulator is an 8-bit buffer register that stores intermediate answers during a computer run. The accumulator has two outputs. The two-state output goes directly to the adder-subtractor and the three-state output goes to the bus. This implies that the 8-bit accumulator word continuously drives the adder- subtractor but only appears on the W bus when ea is high.

2.5.1 RTL viewer

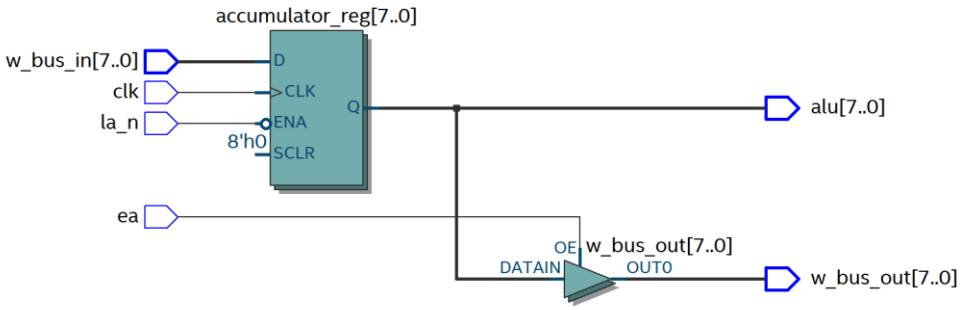


Figure 1-0-11 accumulator RTL

2.5.2 simulation

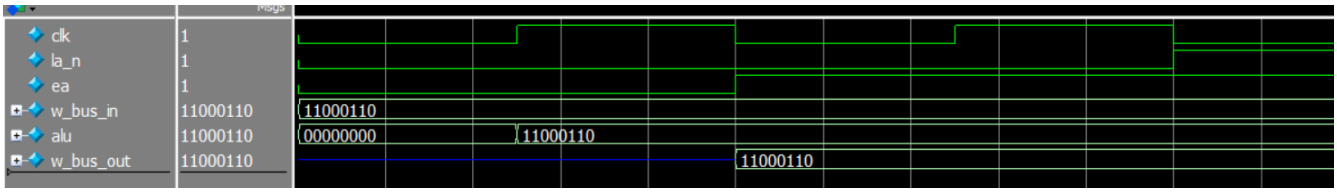


Figure 1-0-12 accumulator simulation

2.6 Adder / subtractor

The adder-subtractor asynchronously adds to or subtracts a value from the the accumulator depending on the value of Su. It makes use of 2's complement to achieve this When Su is low the output of the adder-subtractor is the sum of the values in the accumulator .

2.6.1 RTL viewer

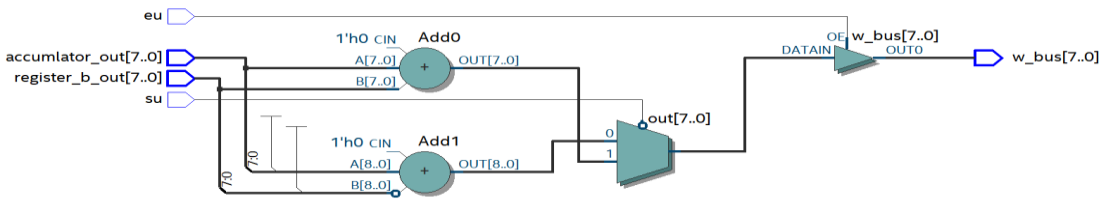


Figure 1-0-13 adder subtractor RTL

2.6.2 simulation

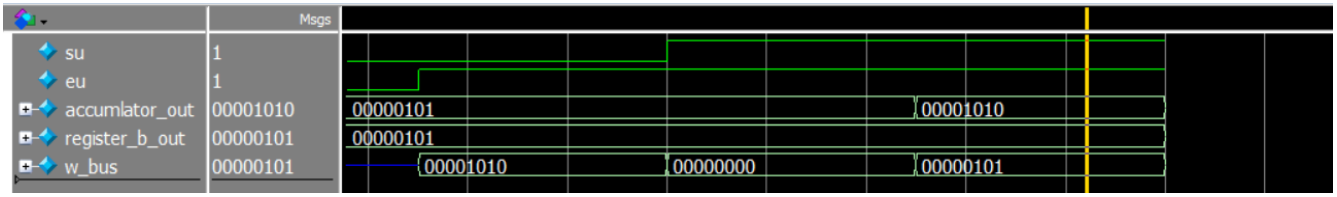


Figure 1-0-14 adder subtractor simulation

2.7 register b

The B-register is a buffer register used in performing arithmetic operations. It supplies the number to be added or subtracted from the contents of the accumulator to the adder/subtractor. When data is available at the bus and Lb is low, at the positive clock edge, B register gets and stores the data.

2.7.1 RTL viewer

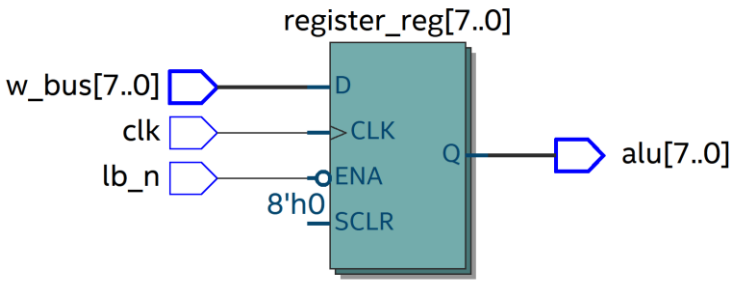


Figure 1-0-15 register B RTL

2.7.2 simulation

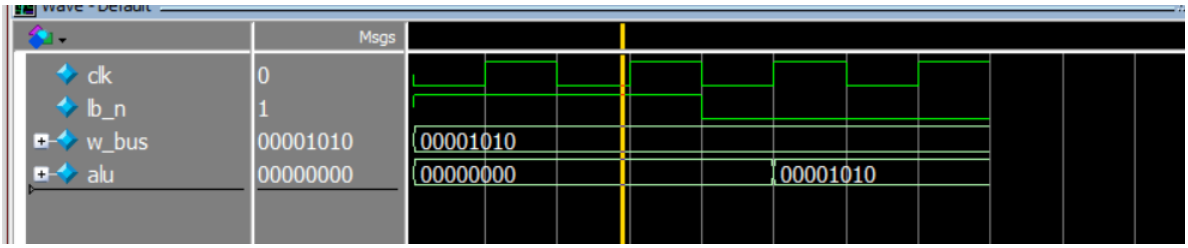


Figure 1-0-16 register B simulation

2.8 output register

The output register gets and stores the value stored in the accumulator usually after the performance of an arithmetic operation. The answer that is stored in the accumulator is loaded into the output register through the W bus. This is done in the next positive clock edge when Ea is high and Lo is low. The processed data can now be displayed to the outside world.

2.8.1 RTL viewer

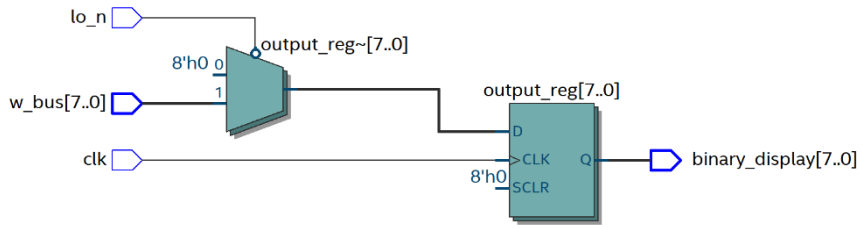


Figure 1-0-17 output register

2.8.2 simulation

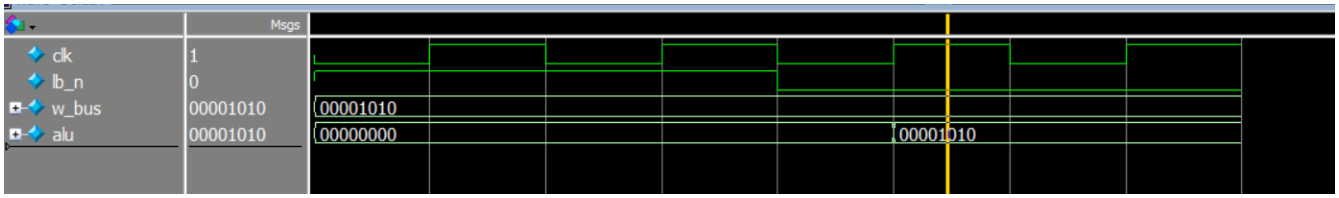
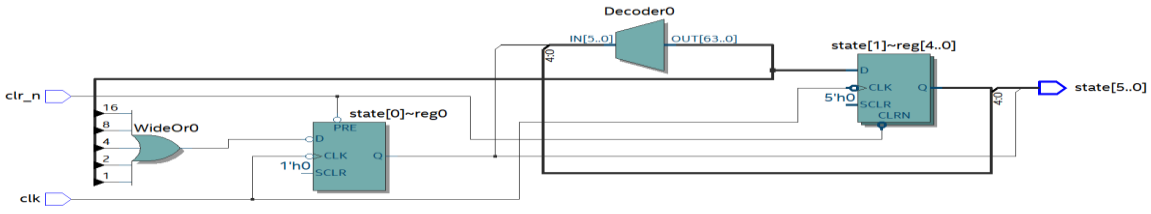


Figure 1-0-18 output register simulation

2.9 controller

The controller-sequencer sends out signals that control the computer and makes sure things happen only when they are supposed to. The 12 bit output signals from controller-sequencer is called the control word which determines how the registers will react to the next positive clock edge. The controller control other block by FSM. In controller we must use ring counter because the sap use six clock cycles and counter will use to determine the state. the ring counter RTL appear at the figure 1-19



- Those control signals are:
- Cp = Increment PC
- Ep = Enable PC ouput to WBUS (1 = Enable)
- lm_n = load in memory address register(0 = load)
- ce_n = Enable output of RAM data to WBUS (0 = enable)
- Li_n = Load Instruction Register from WBUS (0 = load)
- Ei_n = Enable ouput of address from Instruction Register to lower 4 bits of WBUS
- La_n = Load data into the accumulator from WBUS (0 = load)
- Ea = put ouput of accumulator to WBUS (1 = enable)
- Su = ALU operation (0 = Add, 1 = Subtract)
- Eu = Enable output of ALU to WBUS (1 = enable)
- Lb_n = Load data into Register B from WBUS (0 = load)
- Lo_n = Load data into Output Register (0 = load)
- enio = Enable output from Input Register (1 = enable)
- HLT_n = Inverse Halt operation. (0 = HALT)

2.9.1 simulation

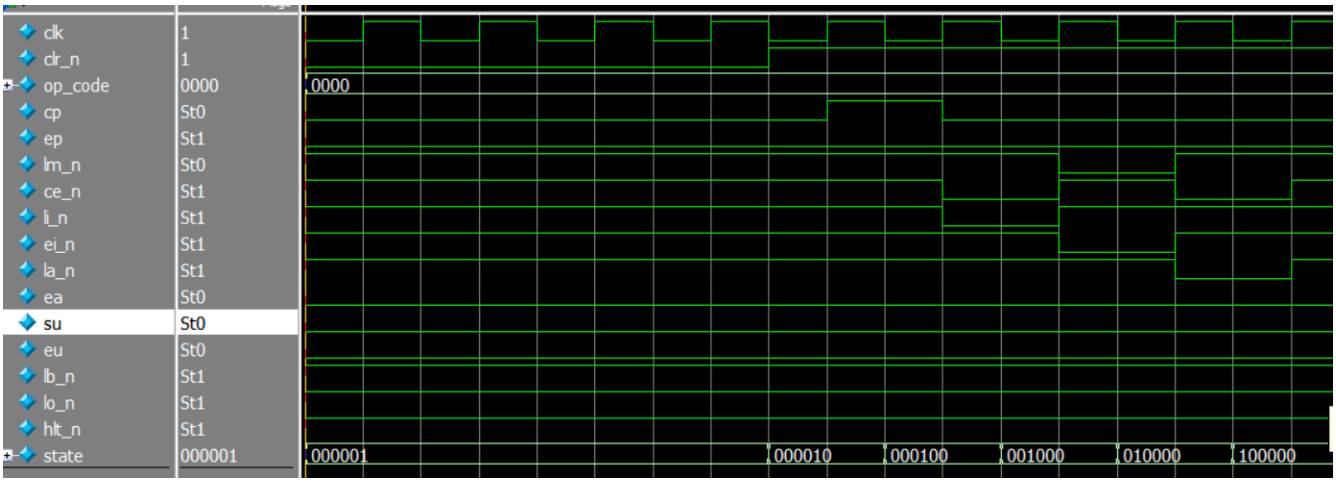


Figure 1-0-19 controller simulation

2.9.2 RTL viewer

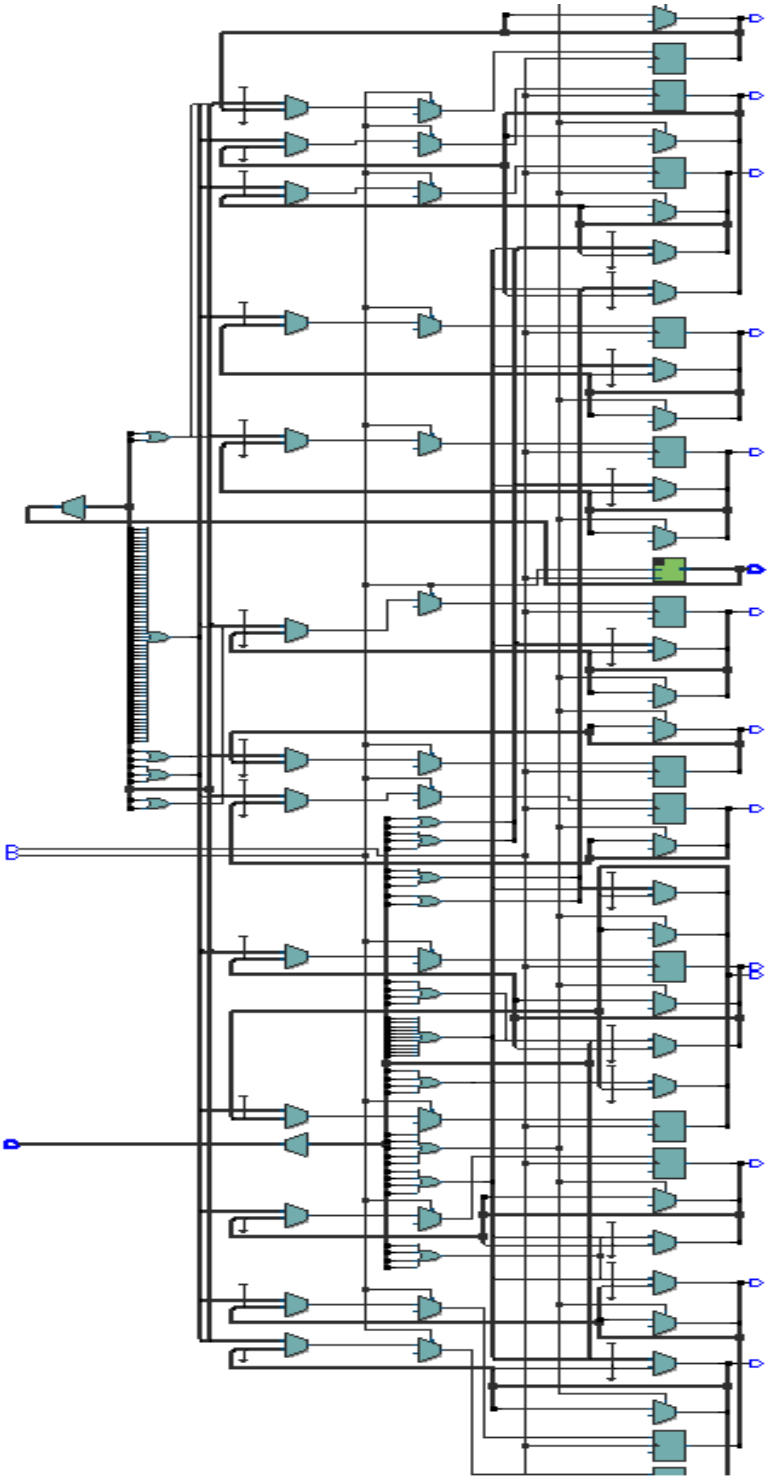


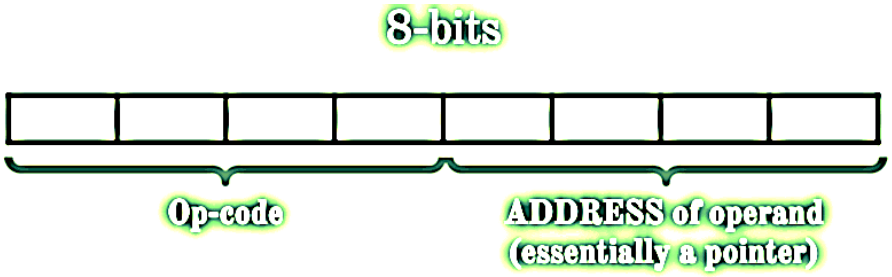
Figure 1-20 controller RTL

3.Instruction set

The SAP-1 has four operations that it can perform:

- [0000] LDA $X: Load the value at memory location $X into Accumulator.
- [0001] ADD $X: Load the value at memory location $X into register b and Add the value at memory location $X to Accumulator and store the sum in Accumulator.
- [0010] SUB $X: Load the value at memory location $X into register b and sub the value at memory location $X to Accumulator and store the sum in Accumulator.
- [1110] OUT: Load accumulator data into output register
- [1111] HLT: Halt execution of the program.

4.Instruction format



Complete code includes opcode and operand.

The instruction format of SAP-1 Computer is (XXXX) (XXXX)
The first four bits make the op-code while the last four bits make the operand (address).
Example:

LDA FH = 0000 1111
ADD EH = 0001 1110
HLT = 1111 XXXX

5.Machine cycle and Instruction cycle

SAP1 has six T-states (three fetch and three execute cycles). All instructions require all the six T-states for execution. The unused T- state is marked as No Operation (NOP) cycle. Each T-state is called a machine cycle for SAP1. A ring counter is used to generate a T-state at every falling edge of clock pulse. The ring counter show at figure 2.0.

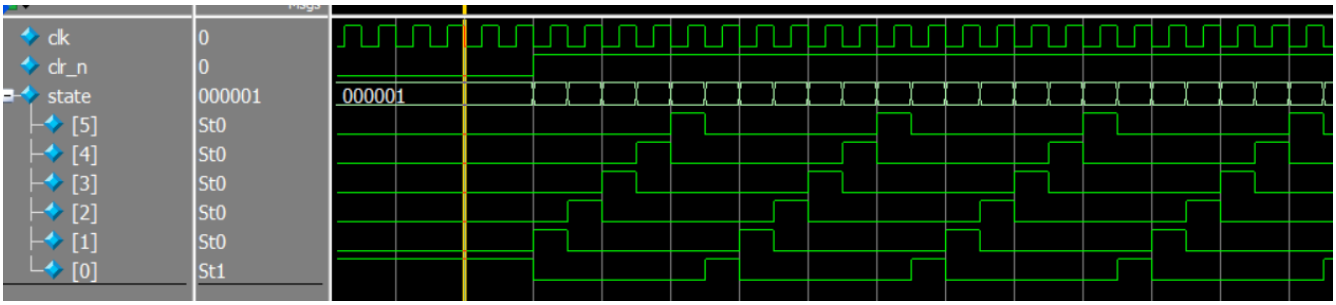


Figure 20 ring counter

The ring counter output is reset after the 6th T-state. the instruction cycles shows in the figure 2.0.

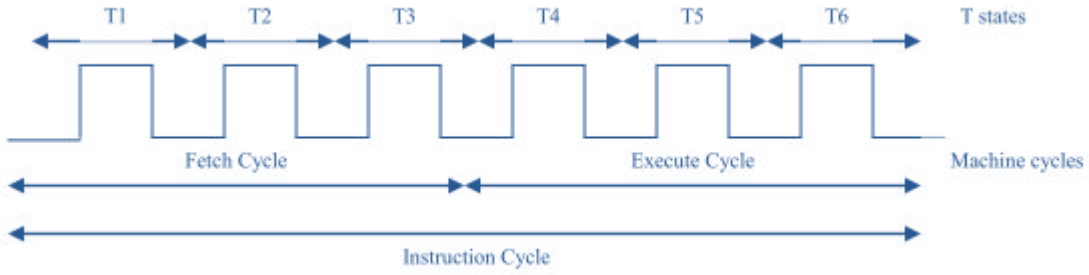


Figure 21 instruction cycle

For SAP-1:	FETCH CYCLE – T1, T2, T3 machine cycle
	EXECUTE CYCLE - T4, T5, T6 machine cycle.
	Instruction cycle = Machine cycle
	Instruction cycle = Fetch cycle + Execution cycle
	One instruction is executed in one instruction cycle

5.1FETCH CYCLE

- **All Instructions**

Stage 0	Stage 1	Stage 2
▪ Put the PC onto the bus. ▪ Load that value into the MAR.	▪ Increment the PC.	▪ Put whatever is in memory at the MAR address onto the bus. ▪ Load it into the IR.

After the first three stages, the actions performed during the next three differ depending on the instruction, and some of the instructions do nothing at all.

5.2EXECUTE CYCLE

- **LDA**

Stage 3	Stage 4	Stage 5
▪Put the instruction operand onto the bus ▪Load that value into the MAR	▪Put whatever is in memory at the MAR address onto the bus ▪Load that value into accumulator	▪NOP

- **ADD**

Stage 3	Stage 4	Stage 5
▪Put the instruction operand onto the bus. ▪Load that value into the MAR.	▪Put whatever is in memory at the MAR address onto the bus. ▪Load that value into Register B.	▪Put the value in the adder onto the bus. ▪Load that value into accumulator.

- **SUB**

Stage 3	Stage 4	Stage 5
▪Put the instruction operand onto the bus. ▪Load that value into the MAR.	▪Put whatever is in memory at the MAR address onto the bus. ▪Load that value into Register B.	▪Do subtraction rather than addition. ▪Put the value in the adder onto the bus. ▪Load that value into accumulator.

- **Out**

Stage 3	Stage 4	Stage 5
▪ put the accumulator in bus . ▪ load the bus value in output register	▪ NOP	○ NOP

- **HLT**

Stage 3	Stage 4	Stage 5
▪ Halt the clock (**hlt**)	▪ NOP	○ NOP

6.results

6.1RTL viewer results

The figure below shows the RTL netlist results for the sap1.

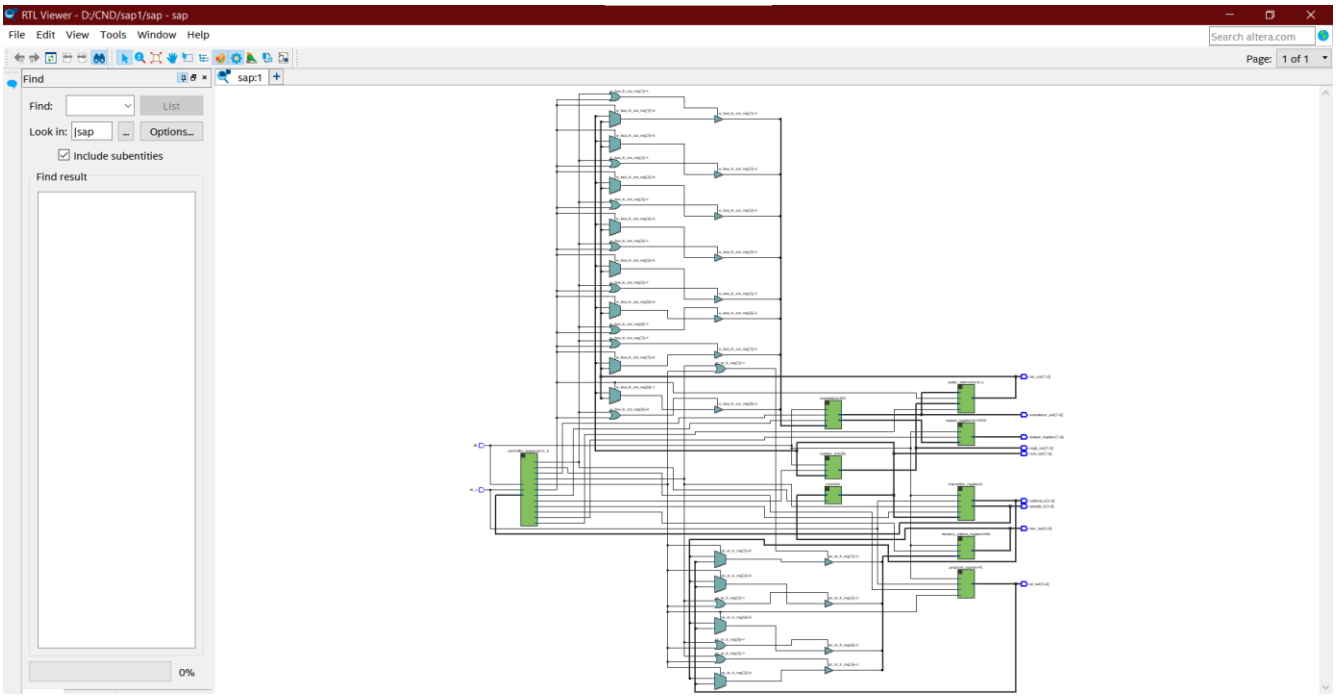
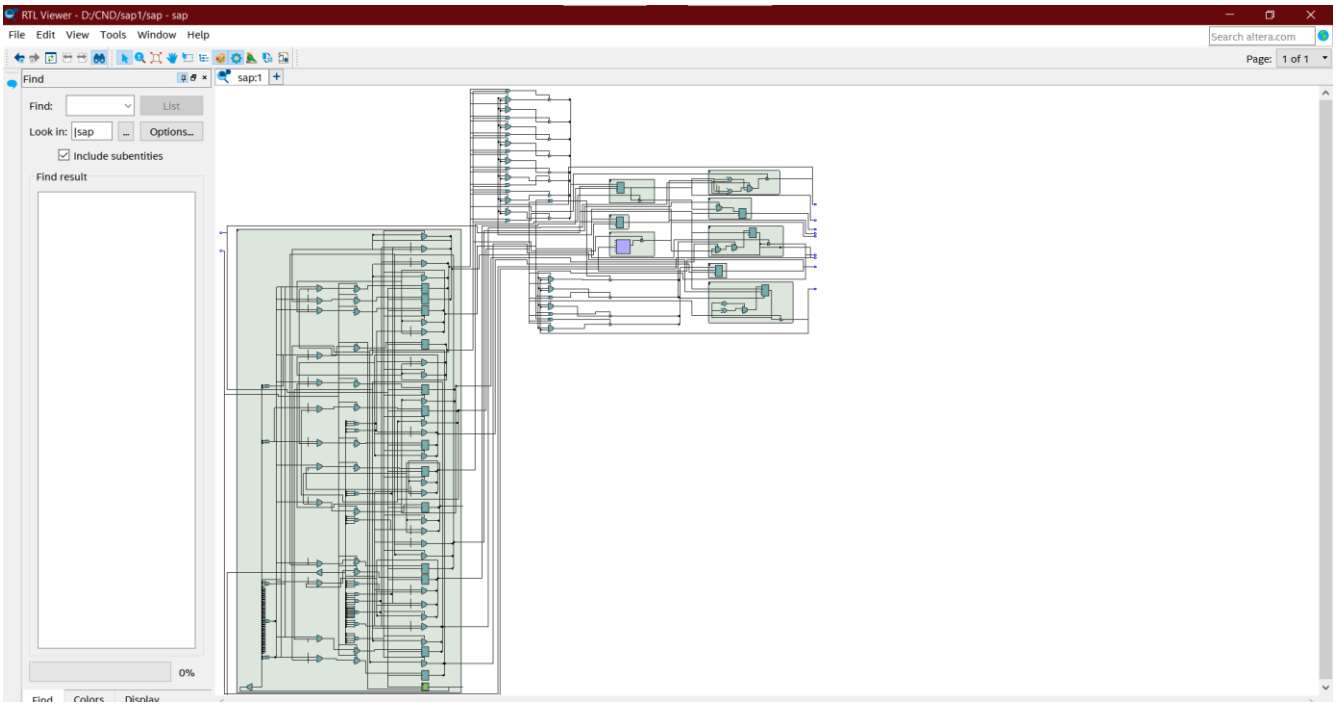
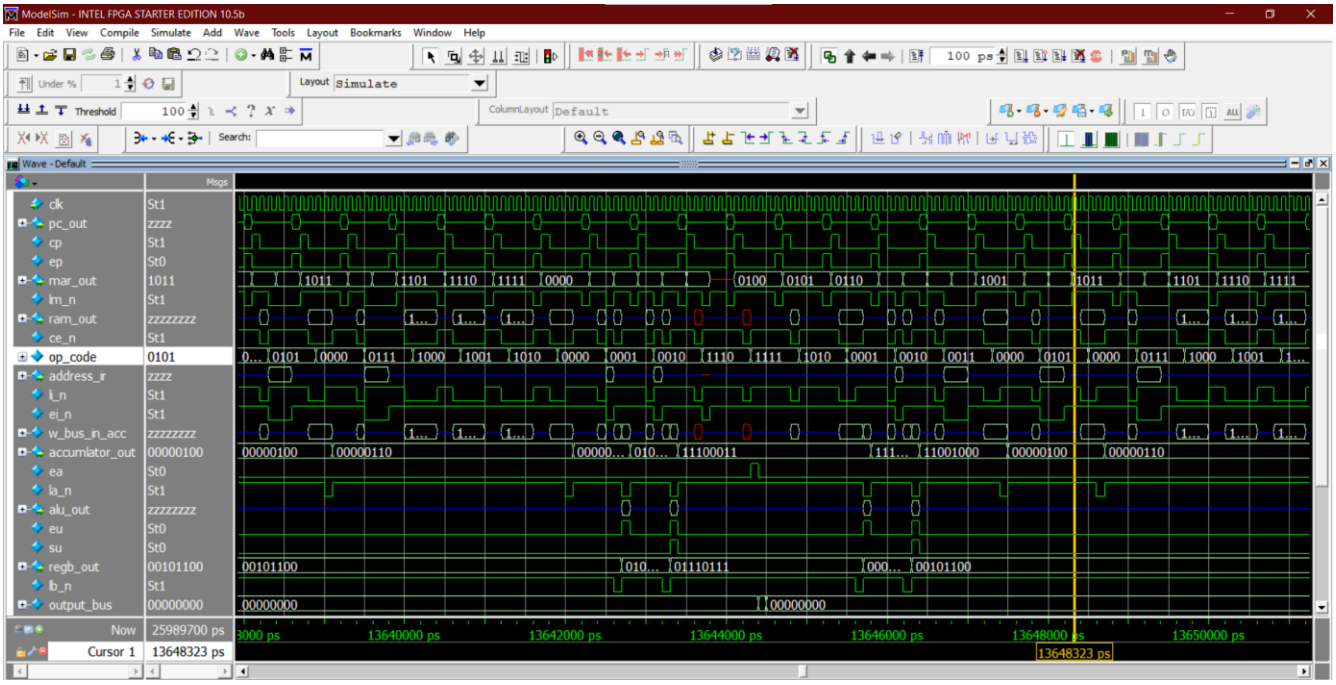


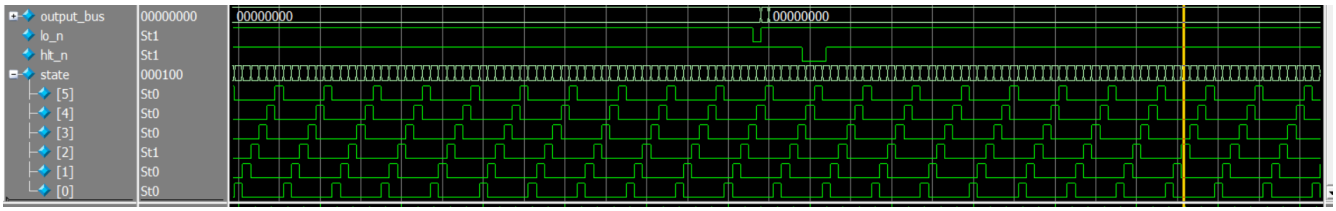
Figure 22 rtl netlist sap1

After expand all block the figure is

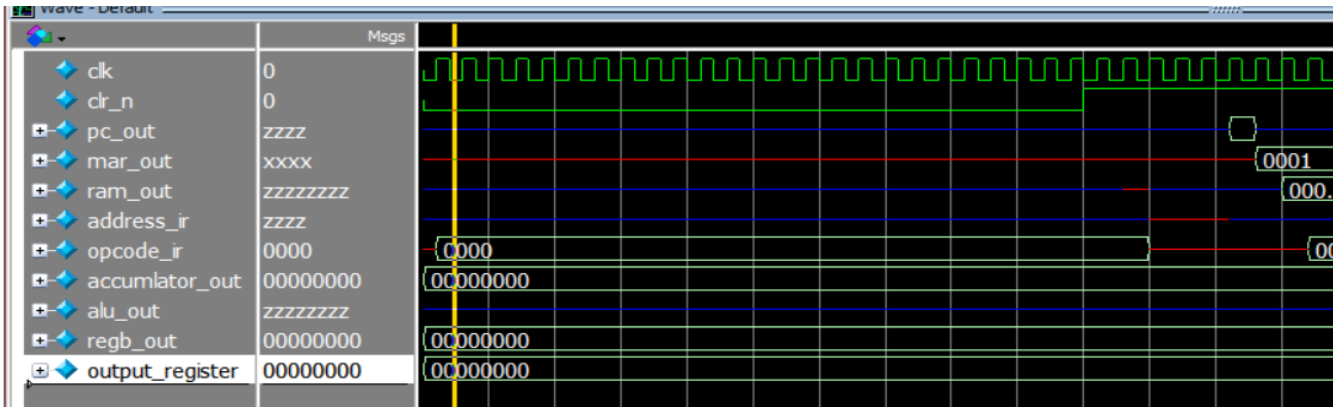


6.2RTL simulation results





The figure below shows the RTL simulation results for the sap1 when clear is low.



And when clear is high. Output is 10100011 which is in accumulator and appear at opcode 1110 .

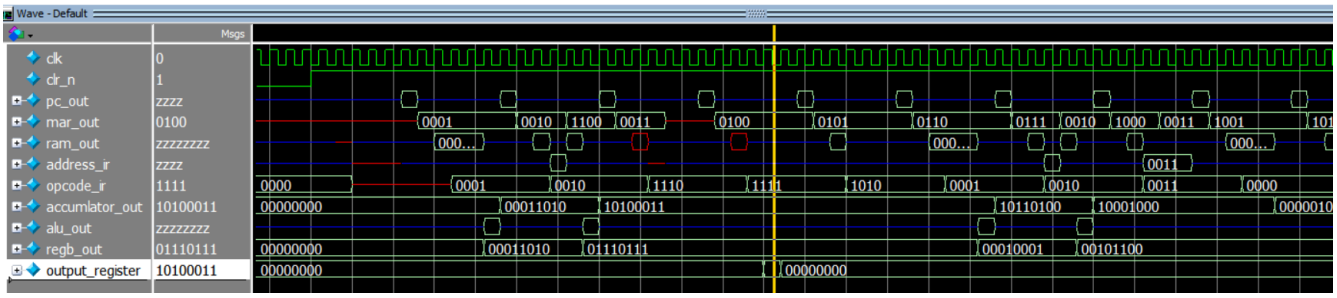
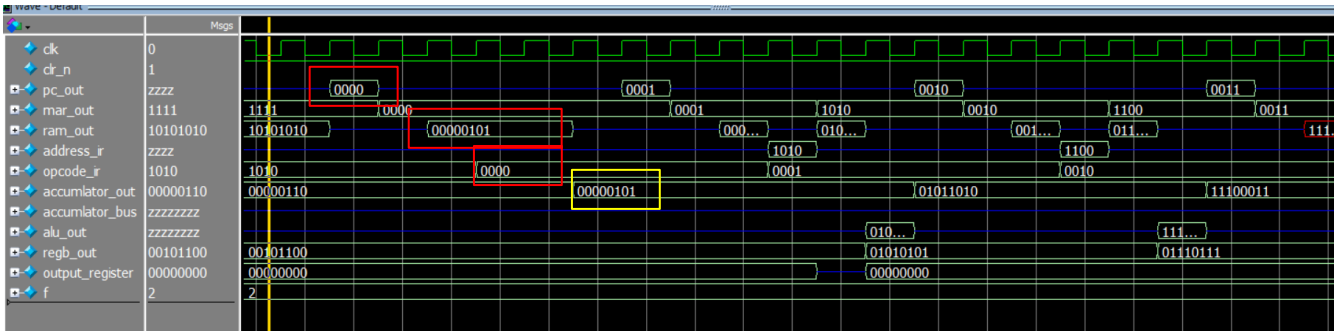


Figure 23 RTL simulation

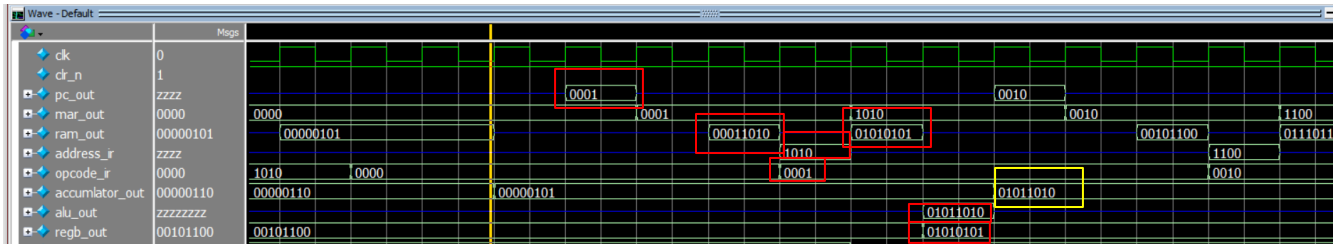
6.2.1The different cases:

When pc =0000 which pointer to the first location in ram and the out of ram is 0000101, The instruction register divide it to op code=0000 which indicate to load the data from ram to accumulator.



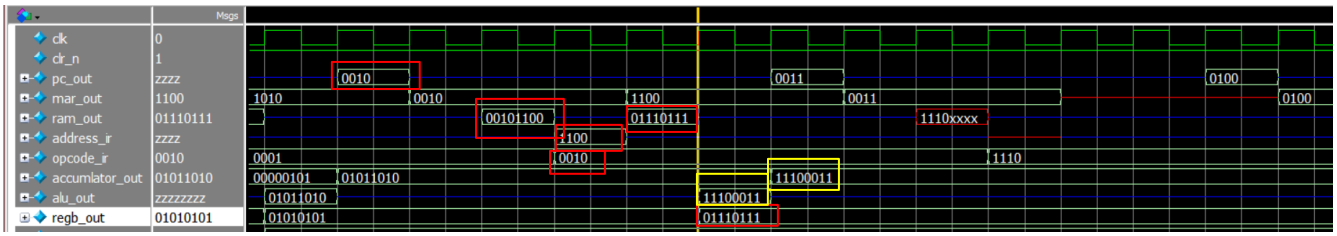
When pc =0001 which pointer to the second location in ram and the out of ram is 00011010, The instruction register divide it to op code=0001 which indicate to add.

So the data at location 1010 load to reg b and added with accumulator data in alu and then store output in accumulator



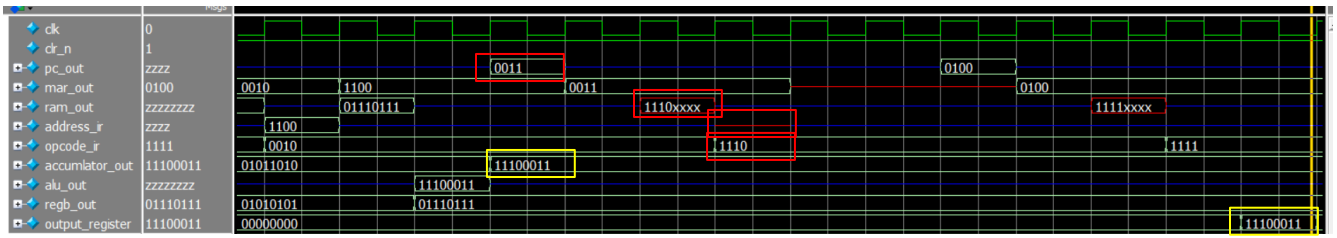
When pc =0010 which pointer to the third location in ram and the out of ram is 0010_1100, The instruction register divide it to op code=0010 which indicate to sub.

So the data at location 1100 load to reg b and subtract with accumulator data in alu and then store output in accumulator



When pc =0011 which pointer to the fourth location in ram and the out of ram is 1110_xxxx, The instruction register divide it to op code=1110 which indicate to output.

So the data at accumulator will be go to output register.



The output file:

```
time=  165, output_register =00000000
time=  355, ram output =1111xxxx
time=  345, program_counter =zzzz
time=  345, memory address register output =0100
time=  355, ram output =1111xxxx
time=  315, address =zzzz
time=  305, op code =1110
time=  275, accumlator_to alu =10100011
time=   10, accumlator_bus =zzzzzzzz
time=  275, alu_out     =zzzzzzzz
time=  265, regb_out =01110111
time=  165, output_register =00000000
time=  365, op code =1111
time=  365, accumlator_bus =10100011
time=  365, ram output =zzzzzzzz
time=  345, program_counter =zzzz
time=  345, memory address register output =0100
time=  365, ram output =zzzzzzzz
time=  315, address =zzzz
time=  365, op code =1111
time=  275, accumlator_to alu =10100011
time=  365, accumlator_bus =10100011
time=  275, alu_out     =zzzzzzzz
time=  265, regb_out =01110111
time=  165, output_register =00000000
time=  375, output_register =10100011
time=  375, accumlator_bus =zzzzzzzz
time=  345, program_counter =zzzz
time=  345, memory address register output =0100
time=  365, ram output =zzzzzzzz
time=  315, address =zzzz
time=  365, op code =1111
time=  275, accumlator_to alu =10100011
time=  375, accumlator_bus =zzzzzzzz
time=  275, alu_out     =zzzzzzzz
time=  265, regb_out =01110111
time=  375, output_register =10100011
```

```
time=  455, program_counter =0110
time=  405, memory address register output =0101
time=  425, ram output =zzzzzzzz
time=  315, address =zzzz
time=  425, op code =1010
time=  275, accumlator_to alu =10100011
time=  375, accumlator_bus =zzzzzzzz
time=  275, alu_out     =zzzzzzzz
time=  265, regb_out =01110111
time=  385, output_register =00000000
time=  465, memory address register output =0110
time=  465, program_counter =zzzz
time=  465, program_counter =zzzz
time=  465, memory address register output =0110
time=  425, ram output =zzzzzzzz
time=  315, address =zzzz
time=  425, op code =1010
time=  275, accumlator_to alu =10100011
time=  375, accumlator_bus =zzzzzzzz
time=  275, alu_out     =zzzzzzzz
time=  265, regb_out =01110111
time=  385, output_register =00000000
time=  475, ram output =00010001
time=  465, program_counter =zzzz
time=  465, memory address register output =0110
time=  475, ram output =00010001
time=  315, address =zzzz
time=  425, op code =1010
time=  275, accumlator_to alu =10100011
time=  375, accumlator_bus =zzzzzzzz
time=  275, alu_out     =zzzzzzzz
```

```
time=  575, accumlator_to alu =10001000
time=  375, accumlator_bus =zzzzzzzz
time=  575, alu_out     =zzzzzzzz
time=  565, regb_out =00101100
time=  385, output_register =00000000
time=  585, memory address register output =100
time=  585, program_counter =zzzz
time=  585, program_counter =zzzz
time=  585, memory address register output =100
time=  565, ram output =zzzzzzzz
time=  555, address =zzzz
time=  545, op code =0010
time=  575, accumlator_to alu =10001000
time=  375, accumlator_bus =zzzzzzzz
time=  575, alu_out     =zzzzzzzz
time=  565, regb_out =00101100
time=  385, output_register =00000000
time=  595, ram output =00110011
time=  585, program_counter =zzzz
time=  585, memory address register output =100
time=  595, ram output =00110011
time=  555, address =zzzz
time=  545, op code =0010
time=  575, accumlator_to alu =10001000
time=  375, accumlator_bus =zzzzzzzz
time=  575, alu_out     =zzzzzzzz
```

```
time=  575, program_counter =1000
time=  555, memory address register output =0010
time=  565, ram output =zzzzzzzz
time=  555, address =zzzz
time=  545, op code =0010
time=  575, accumlator_to alu =10001000
time=  375, accumlator_bus =zzzzzzzz
time=  575, alu_out     =zzzzzzzz
time=  565, regb_out =00101100
time=  385, output_register =00000000
time=  585, memory address register output =1000
time=  585, program_counter =zzzz
time=  585, program_counter =zzzz
time=  585, memory address register output =1000
time=  565, ram output =zzzzzzzz
time=  555, address =zzzz
time=  545, op code =0010
time=  575, accumlator_to alu =10001000
time=  375, accumlator_bus =zzzzzzzz
time=  575, alu_out     =zzzzzzzz
time=  565, regb_out =00101100
time=  385, output_register =00000000
time=  595, ram output =00110011
time=  585, program_counter =zzzz
time=  585, memory address register output =1000
time=  595, ram output =00110011
time=  555, address =zzzz
time=  545, op code =0010
time=  575, accumlator_to alu =10001000
time=  375, accumlator_bus =zzzzzzzz
time=  575, alu_out     =zzzzzzzz
time=  565, regb_out =00101100
time=  385, output_register =00000000
time=  605, op code =0011
time=  605, address =0011
time=  605, ram output =zzzzzzzz
time=  585, program_counter =zzzz
time=  585, memory address register output =1000
time=  605, ram output =zzzzzzzz
time=  605, address =0011
time=  605, op code =0011
time=  575, accumlator_to alu =10001000
time=  375, accumlator_bus =zzzzzzzz
time=  575, alu_out     =zzzzzzzz
time=  565, regb_out =00101100
time=  385, output_register =00000000
time=  615, memory address register output =0011
time=  585, program_counter =zzzz
time=  615, memory address register output =0011
time=  605, ram output =zzzzzzzz
time=  605, address =0011
time=  605, op code =0011
time=  575, accumlator_to alu =10001000
time=  375, accumlator_bus =zzzzzzzz
time=  575, alu_out     =zzzzzzzz
```

```
time=  595, ram output =00110011
time=  555, address =zzzz
time=  545, op code =0010
time=  575, accumlator_to alu =10001000
time=  375, accumlator_bus =zzzzzzzz
time=  575, alu_out     =zzzzzzzz
time=  565, regb_out =00101100
time=  385, output_register =00000000
time=  605, op code =0011
time=  605, address =0011
time=  605, ram output =zzzzzzzz
time=  585, program_counter =zzzz
time=  585, memory address register output =1000
time=  605, ram output =zzzzzzzz
time=  605, address =0011
time=  605, op code =0011
time=  575, accumlator_to alu =10001000
time=  375, accumlator_bus =zzzzzzzz
time=  575, alu_out     =zzzzzzzz
time=  565, regb_out =00101100
time=  385, output_register =00000000
time=  615, memory address register output =0011
time=  585, program_counter =zzzz
time=  615, memory address register output =0011
time=  605, ram output =zzzzzzzz
time=  605, address =0011
time=  605, op code =0011
time=  575, accumlator_to alu =10001000
time=  375, accumlator_bus =zzzzzzzz
time=  575, alu_out     =zzzzzzzz
time=  565, regb_out =00101100
```

```
time=  635, program_counter =1001
time=  635, address =zzzz
time=  635, program_counter =1001
time=  615, memory address register output =0011
time=  605, ram output =zzzzzzzz
time=  635, address =zzzz
time=  605, op code =0011
time=  575, accumlator_to alu =10001000
time=  375, accumlator_bus =zzzzzzzz
time=  575, alu_out     =zzzzzzzz
time=  565, regb_out =00101100
time=  385, output_register =00000000
time=  645, memory address register output =1001
time=  645, program_counter =zzzz
time=  645, program_counter =zzzz
time=  645, memory address register output =1001
time=  605, ram output =zzzzzzzz
time=  635, address =zzzz
time=  605, op code =0011
time=  575, accumlator_to alu =10001000
time=  375, accumlator_bus =zzzzzzzz
time=  575, alu_out     =zzzzzzzz
time=  565, regb_out =00101100
time=  385, output_register =00000000
time=  655, ram output =00000100
time=  645, program_counter =zzzz
time=  645, memory address register output =1001
time=  655, ram output =00000100
time=  635, address =zzzz
time=  605, op code =0011
time=  575, accumlator_to alu =10001000
time=  375, accumlator_bus =zzzzzzzz
time=  575, alu_out     =zzzzzzzz
time=  565, regb_out =00101100
time=  385, output_register =00000000
time=  665, op code =0000
time=  645, program_counter =zzzz
time=  645, memory address register output =1001
time=  655, ram output =00000100
time=  635, address =zzzz
time=  665, op code =0000
time=  575, accumlator_to alu =10001000
time=  375, accumlator_bus =zzzzzzzz
time=  575, alu_out     =zzzzzzzz
time=  565, regb_out =00101100
time=  385, output_register =00000000
time=  645, program_counter =zzzz
time=  645, memory address register output =1001
time=  655, ram output =00000100
time=  635, address =zzzz
time=  665, op code =0000
time=  575, accumlator_to alu =10001000
time=  375, accumlator_bus =zzzzzzzz
time= 1265, op code =1110
time= 1235, accumlator_to alu =11100011
time=  375, accumlator_bus =zzzzzzzz
time= 1235, alu_out     =zzzzzzzz
time= 1225, regb_out =01110111
time= 1165, output_register =00000000
time= 1305, memory address register output =0100
time= 1305, program_counter =zzzz
time= 1305, program_counter =zzzz
time= 1305, memory address register output =0100
time= 1265, ram output =zzzzzzzz
time= 1275, address =zzzz
time= 1265, op code =1110
time= 1235, accumlator_to alu =11100011
time=  375, accumlator_bus =zzzzzzzz
time= 1235, alu_out     =zzzzzzzz
time= 1225, regb_out =01110111
time= 1165, output_register =00000000
time= 1315, ram output =1111xxxx
time= 1305, program_counter =zzzz
time= 1305, memory address register output =0100
time= 1315, ram output =1111xxxx
time= 1275, address =zzzz
time= 1265, op code =1110
time= 1235, accumlator_to alu =11100011
time=  375, accumlator_bus =zzzzzzzz
time= 1235, alu_out     =zzzzzzzz
time= 1225, regb_out =01110111
time= 1165, output_register =00000000
time= 1325, op code =1111
time= 1325, accumlator_bus =11100011
time= 1325, ram output =zzzzzzzz
time= 1305, program_counter =zzzz
time= 1305, memory address register output =0100
time= 1325, ram output =zzzzzzzz
time= 1275, address =zzzz
time= 1325, op code =1111
time= 1235, accumlator_to alu =11100011
time= 1325, accumlator_bus =11100011
time= 1235, alu_out     =zzzzzzzz
time= 1225, regb_out =01110111
time= 1165, output_register =00000000
time= 1335, output_register =11100011
time= 1335, accumlator_bus =zzzzzzzz
time= 1305, program_counter =zzzz
time= 1305, memory address register output =0100
time= 1325, ram output =zzzzzzzz
time= 1275, address =zzzz
time= 1325, op code =1111
time= 1235, accumlator_to alu =11100011
time= 1335, accumlator_bus =zzzzzzzz
time= 1235, alu_out     =zzzzzzzz
time= 1225, regb_out =01110111
time= 1335, output_register =11100011
```

6.3 reporting

6.3.1RTL Flow Summary results

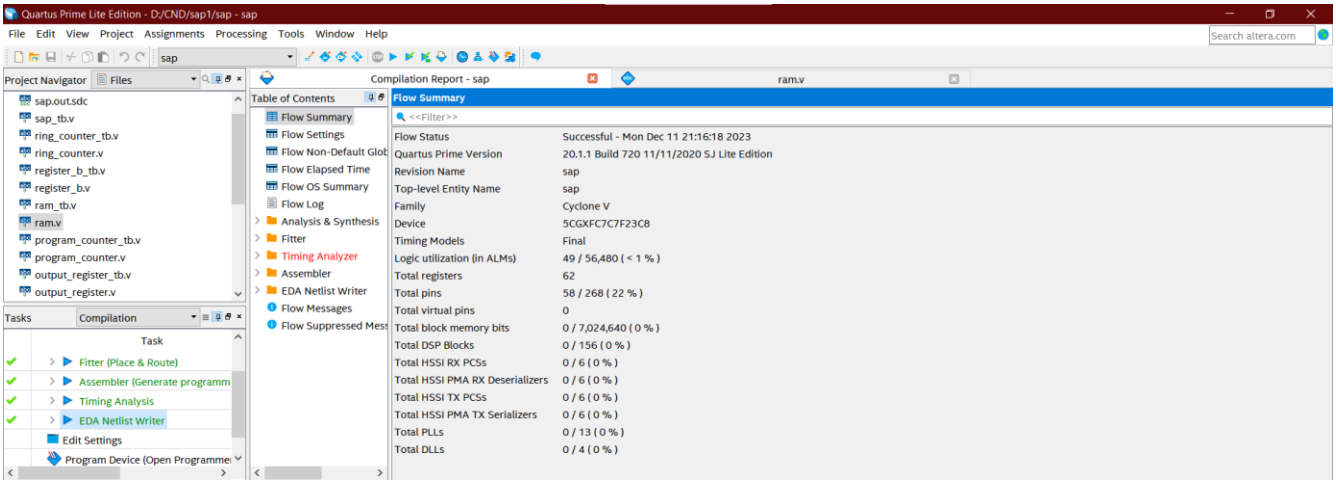
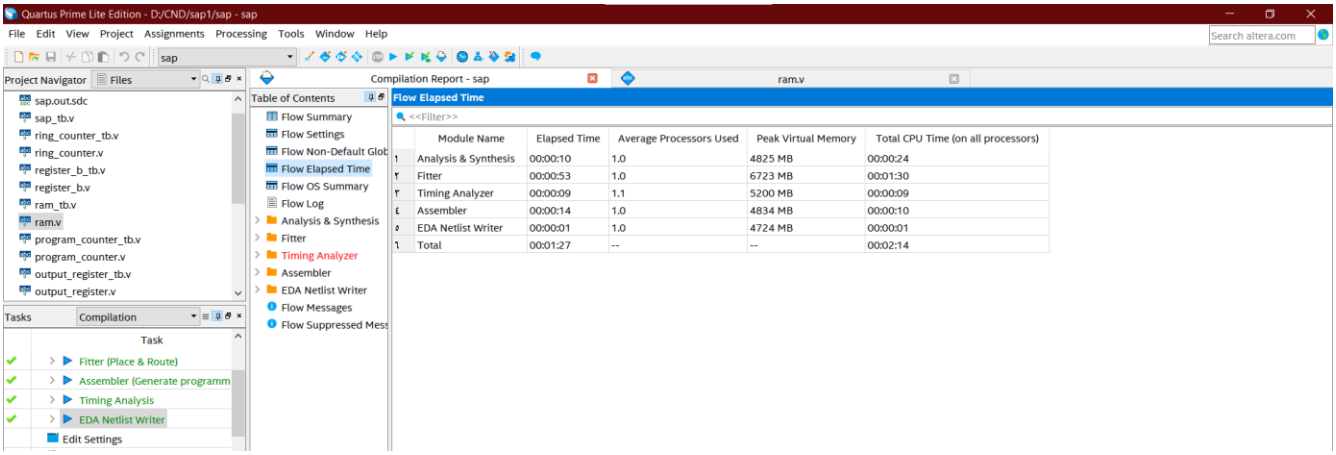
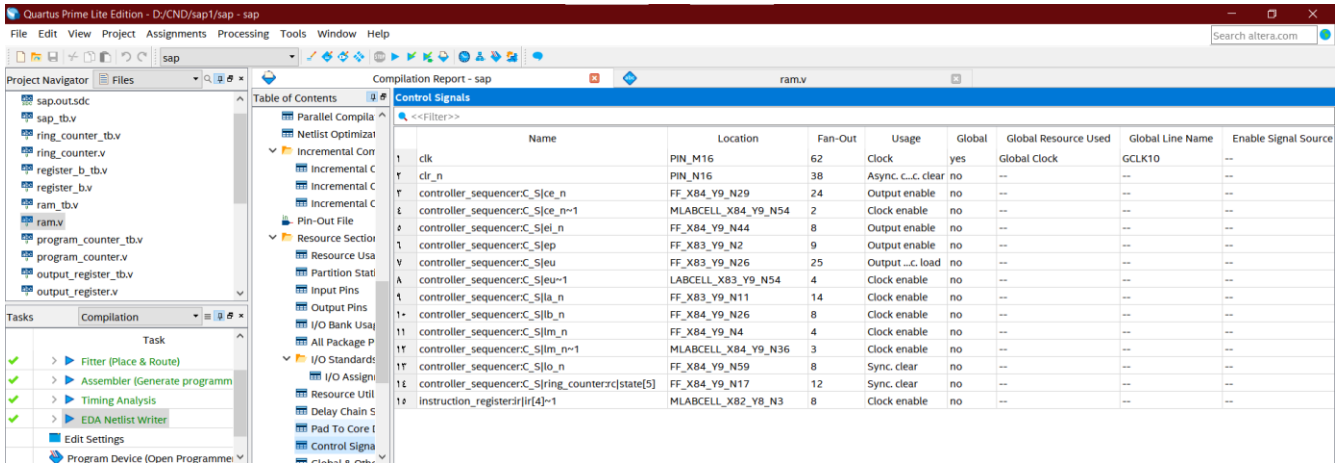


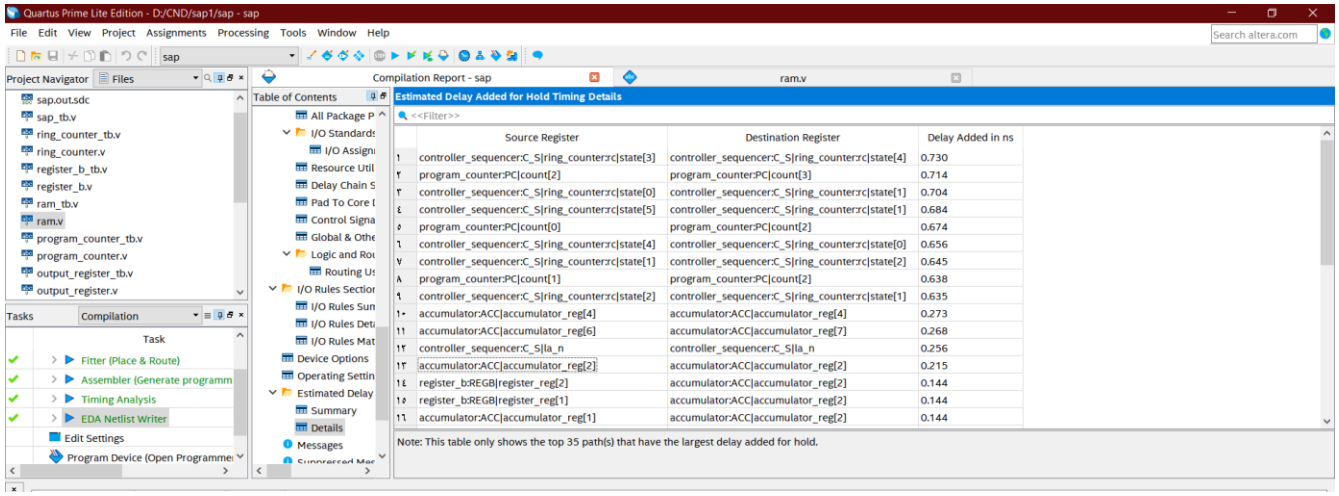
Figure 24 flow summary results

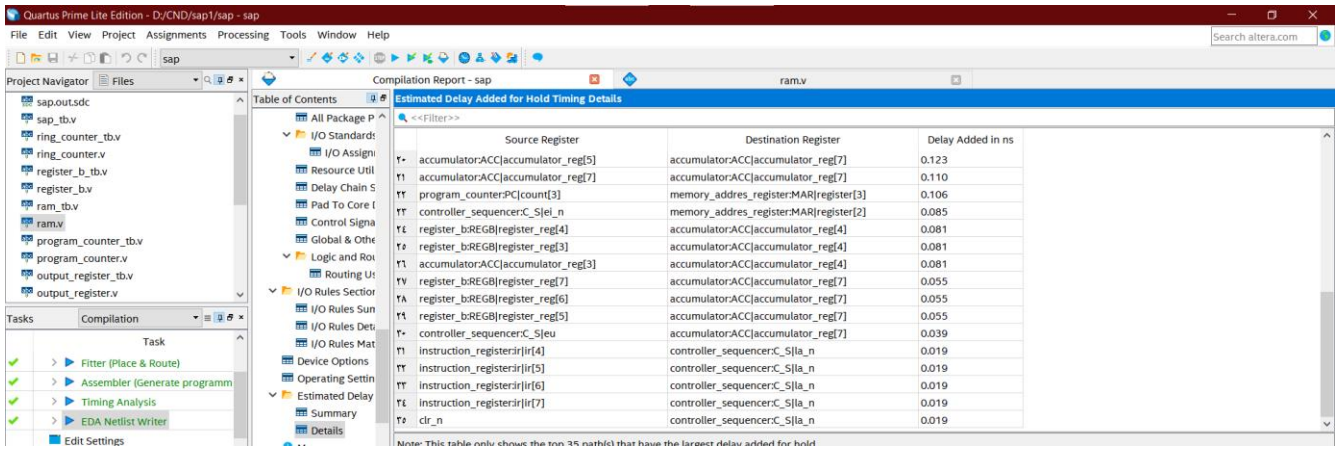


6.3.2fan out of control signal.

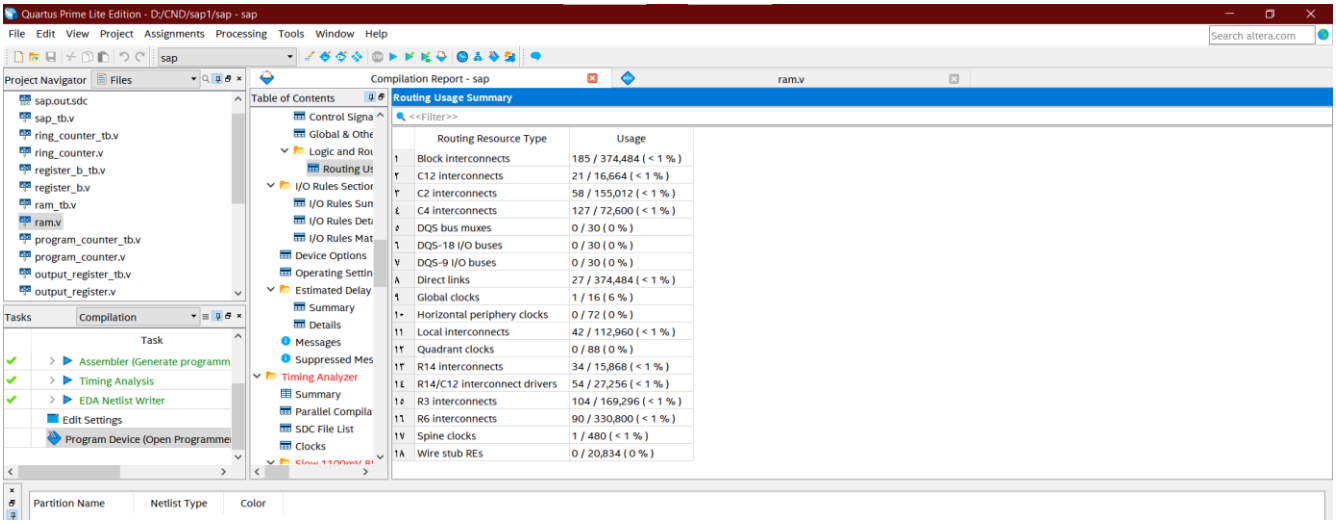


6.3.4delay due to register

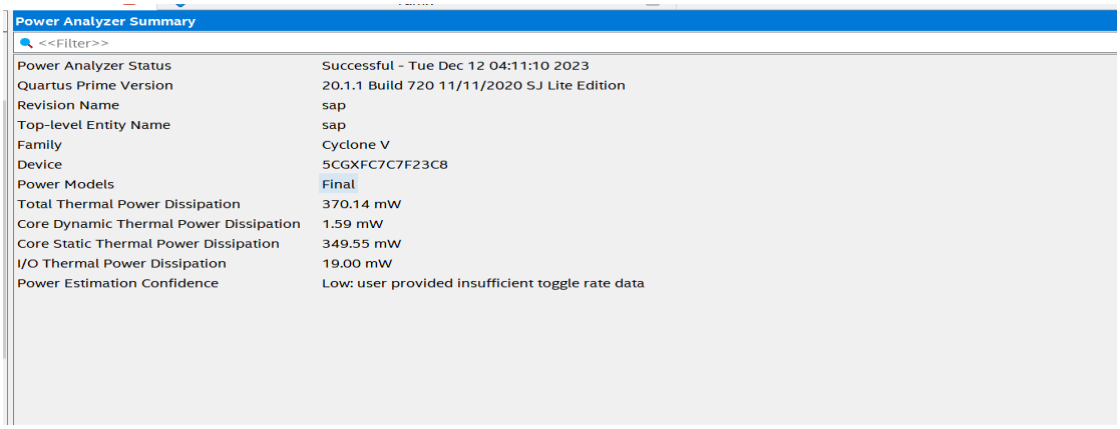




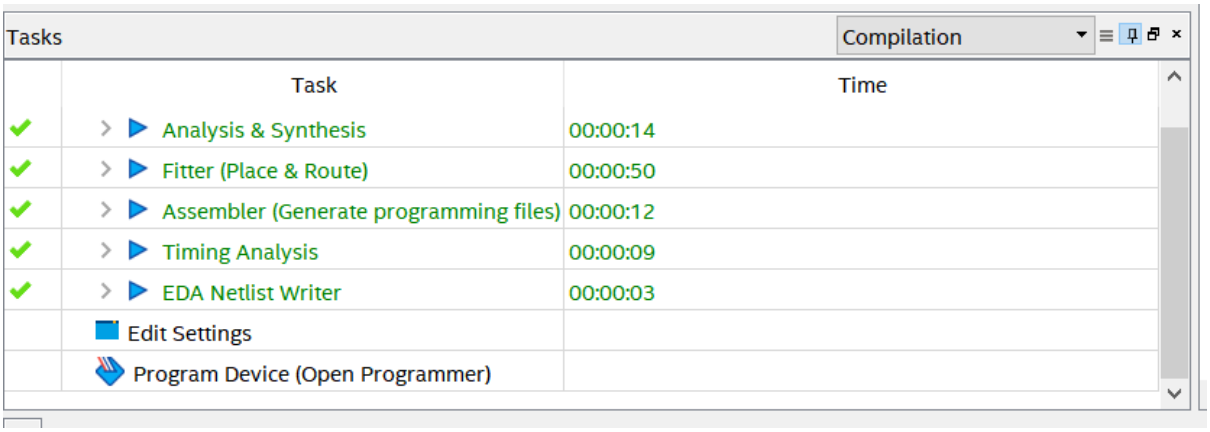
6.3.5the routing summary.



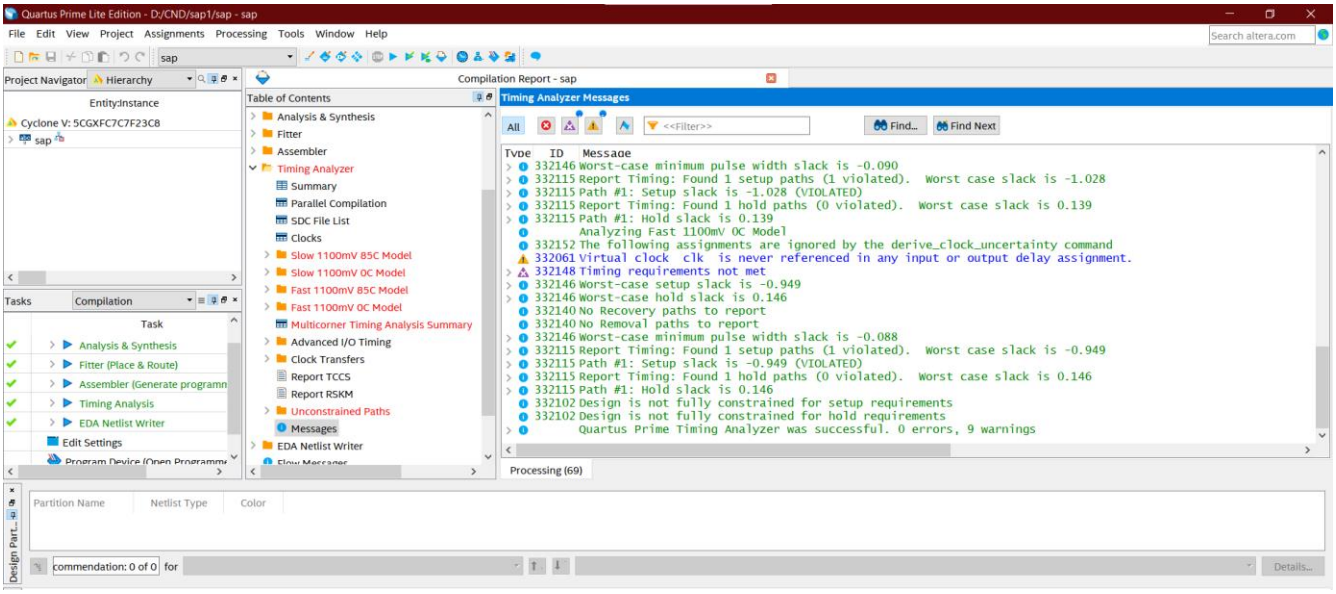
6.3.6 the power report:



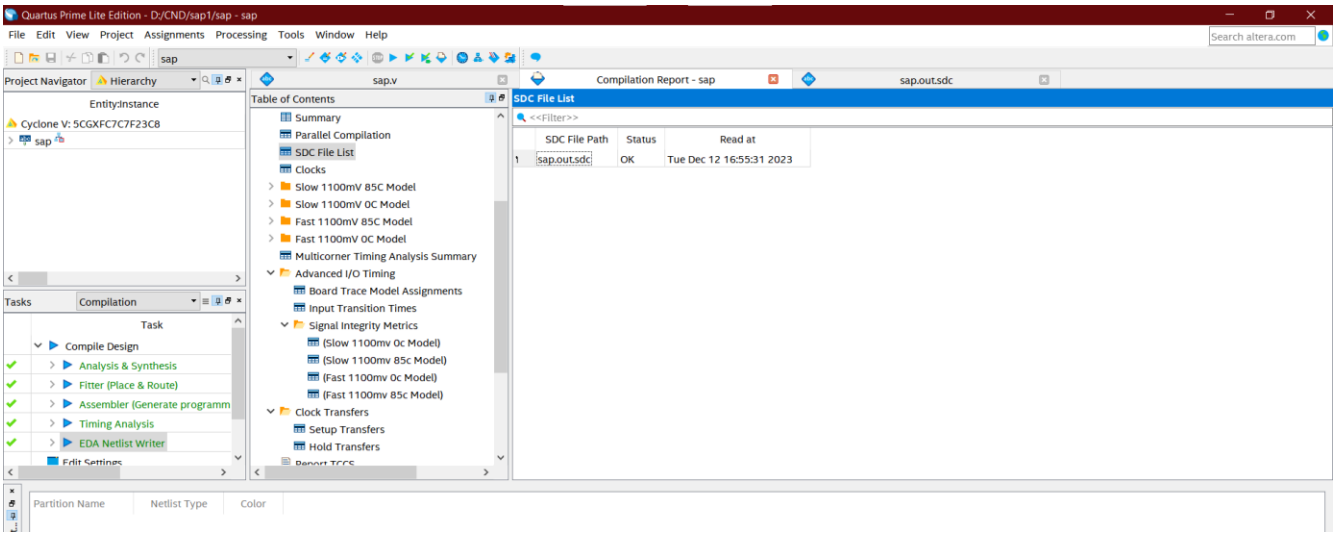
6.3.7 timing report



Timing analyzer before put constrain in the design show at next figure:



After put constrains:



The maximum frequency to work in slow 1100mv 85c model show in the next figure:

Slow 1100mV 85C Model Fmax Summary

<<Filter>>

	Fmax	Restricted Fmax	Clock Name	Note
1	124.53 MHz	124.53 MHz	clk	

The maximum frequency to work in slow 1100mv 0c model show in the next figure:

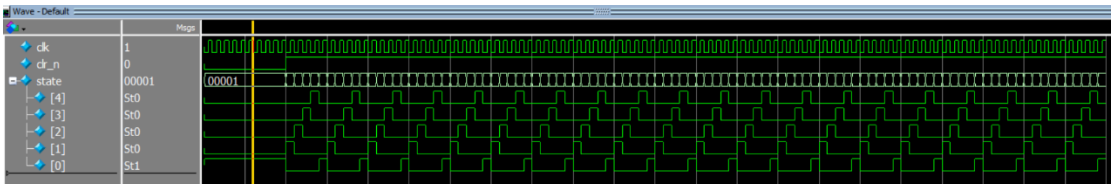
Slow 1100mV 0C Model Fmax Summary

<<Filter>>

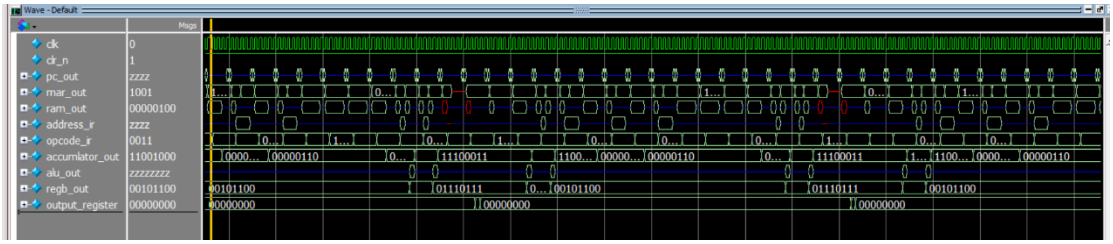
	Fmax	Restricted Fmax	Clock Name	Note
1	128.04 MHz	128.04 MHz	clk	

7.design improvement

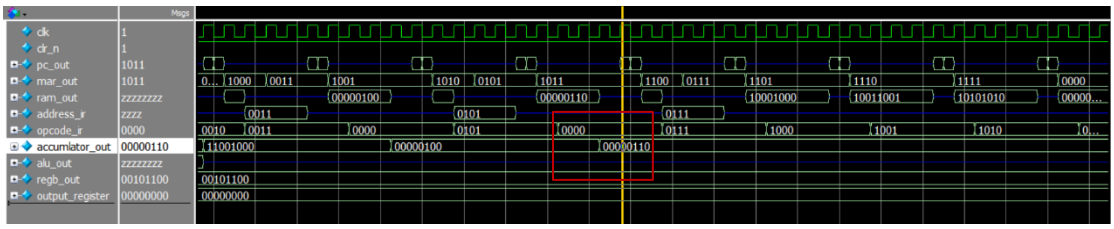
I work to make the design work only in five clock cycle instead of six clocks so the ring counter will be as in next figure



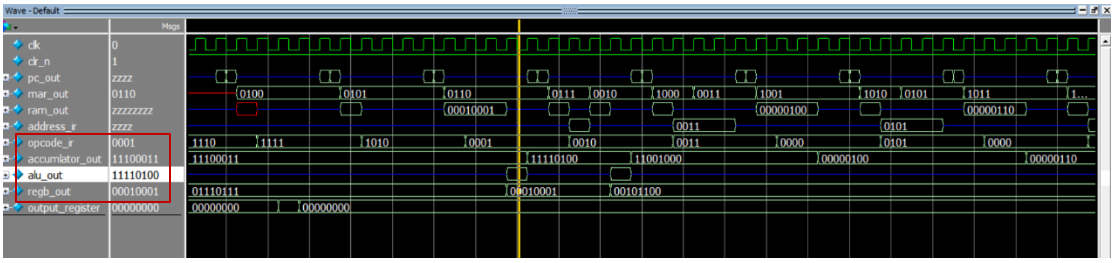
The figure below shows the RTL simulation results for the sap1 .



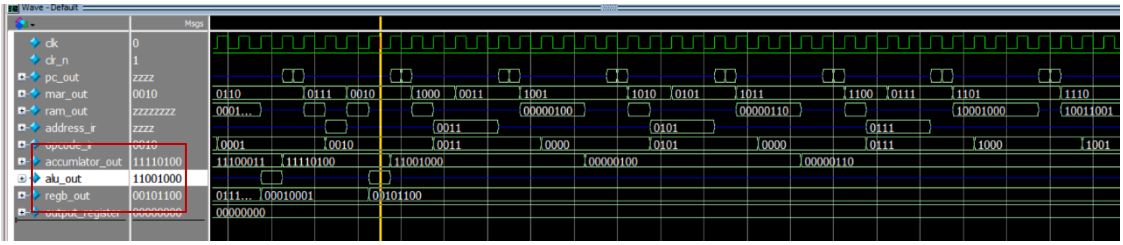
When pc =0000 which pointer to the first location in ram and the out of ram is 00000110, The instruction register divide it to op code=0000 which indicate to load the data from ram to accumulator like previous simulation.



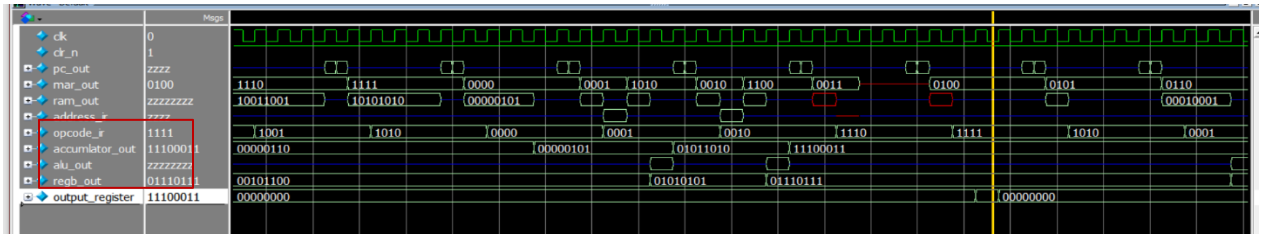
When pc =0001 which pointer to the second location in ram and the out of ram is 00010001, The instruction register divide it to op code=0001 which indicate to add.



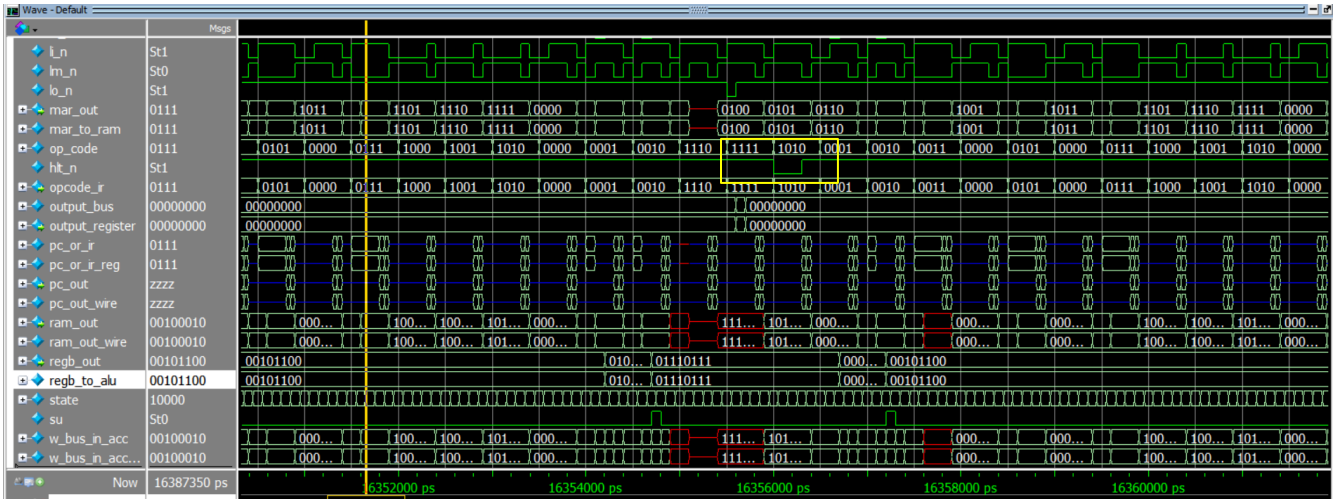
And when op code= 0010 which indicate to sub.



When op code =1110 to output data



When op code =1111 , this indicates to HLT processor



In the flow summary, we find the total register =61 instead of 62 .

Flow Summary

<<Filter>>

Flow Status	Successful - Wed Dec 13 19:08:45 2023
Quartus Prime Version	20.1.1 Build 720 11/11/2020 SJ Lite Edition
Revision Name	sap
Top-level Entity Name	sap
Family	Cyclone V
Device	5CGXFC7C7F23C8
Timing Models	Final
Logic utilization (in ALMs)	54 / 56,480 (< 1 %)
Total registers	61
Total pins	66 / 268 (25 %)
Total virtual pins	0
Total block memory bits	0 / 7,024,640 (0 %)
Total DSP Blocks	0 / 156 (0 %)
Total HSSI RX PCSs	0 / 6 (0 %)
Total HSSI PMA RX Deserializers	0 / 6 (0 %)
Total HSSI TX PCSs	0 / 6 (0 %)
Total HSSI PMA TX Serializers	0 / 6 (0 %)
Total PLLs	0 / 13 (0 %)
Total DLLs	0 / 4 (0 %)

The timing to do simulation at the five cycles.

	Task	Time
✔	Analysis & Synthesis	00:00:12
✔	Fitter (Place & Route)	00:00:51
✔	Assembler (Generate programming files)	00:00:14
✔	Timing Analysis	00:00:10
✔	EDA Netlist Writer	00:00:03

At six clock cycles the maximum frequency in technique slow 1100mv 85c =124 MHZ And in technique slow 1100mv 0c =128 MHZ.

After making enhancement:

At five clock cycles the maximum frequency in technique slow 1100mv 85c =132 MHZ

And in technique slow 1100mv 0c =136 MHZ

Slow 1100mV 85C Model Fmax Summary

<<Filter>>

	Fmax	Restricted Fmax	Clock Name	Note
1	132.24 MHz	132.24 MHz	clk	

Slow 1100mV 0C Model Fmax Summary

<<Filter>>

	Fmax	Restricted Fmax	Clock Name	Note
1	136.35 MHz	136.35 MHz	clk	