BANGLADESH UNIVERSITY OF ENGINEERING AND TECHNOLOGY

Department of Electrical and Electronic Engineering

Course No: EEE 468

Course Title: VLSI Laboratory

Title of the Project: 32 Bit Multicycle RISC-V Processor Design.

Section: G2

Group No: 01

Group Member's ID:	**Group Member's Name:**
1706117	Rasin Mohammed Ihtemam
1706121	A.F.M. Iftekhar Rasul
1706134	Khaled Mahmud
1706157	Mahmadul Hassan Robin
1706159	Mahmudul Hasan
1706194	Md. Siratul Mostakim Ifty

Goals we have achieved:

1) We have designed RISC-V 32-bit multicycle processor.

2) We have used this core to execute instructions written in C for detecting a prime or non-prime number.

3) Our designed core is compatible with rv32i instruction set.

4) We have used input global clock (positive edge triggered: clk), global reset (negative edge triggered: rst_n) and core_select signal.

5) We have created memory controller with industry standard interface SPI to load the instruction data to the memory when the core_select signal is 0.

6) Also when the core_select signal is 1 then core side interface of the memory is active to read and write data into the memory.

7) We simulate our core using Cadence NCSim then the waveforms showing the value of register [12] is 1 for the benchmarking instructions.

Core :

Our core is multicycle and we have used von neumann architecture . **Due to synchronous Memory, it needs two clock cycles to read something from memory and so multicycle core has been implemented**.

Datapath of Core: The datapath of core has been taken mainly from **"Sarah Harris, David Harris RISC-V.pdf** ". But the datapath does not include all riscV instructions. So,the datapath has been modified for implementing all instructions from **"RV32 I Base Instruction Set".**

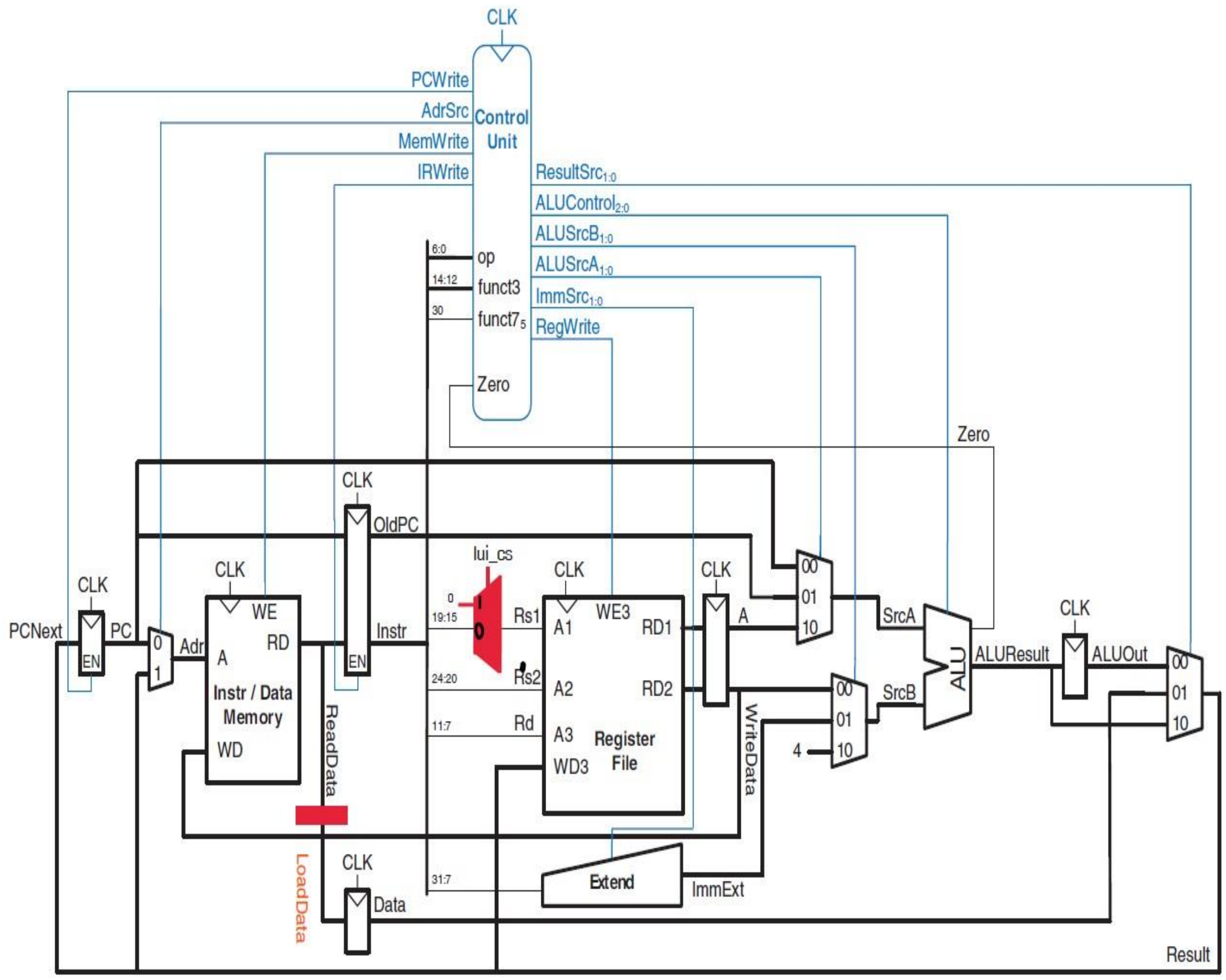
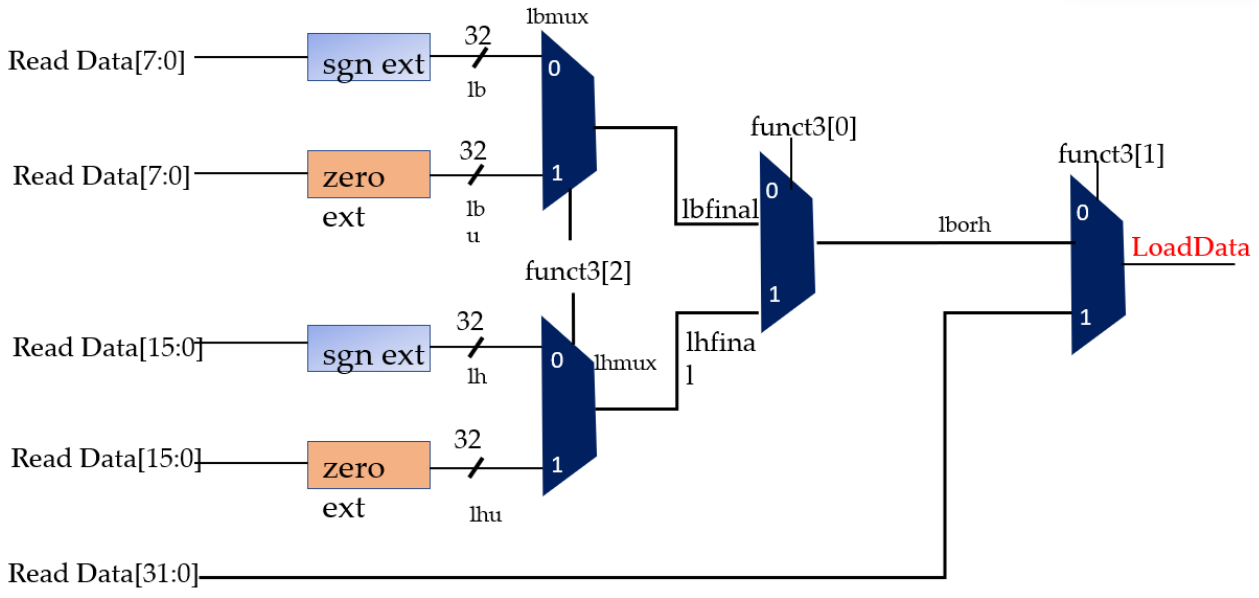
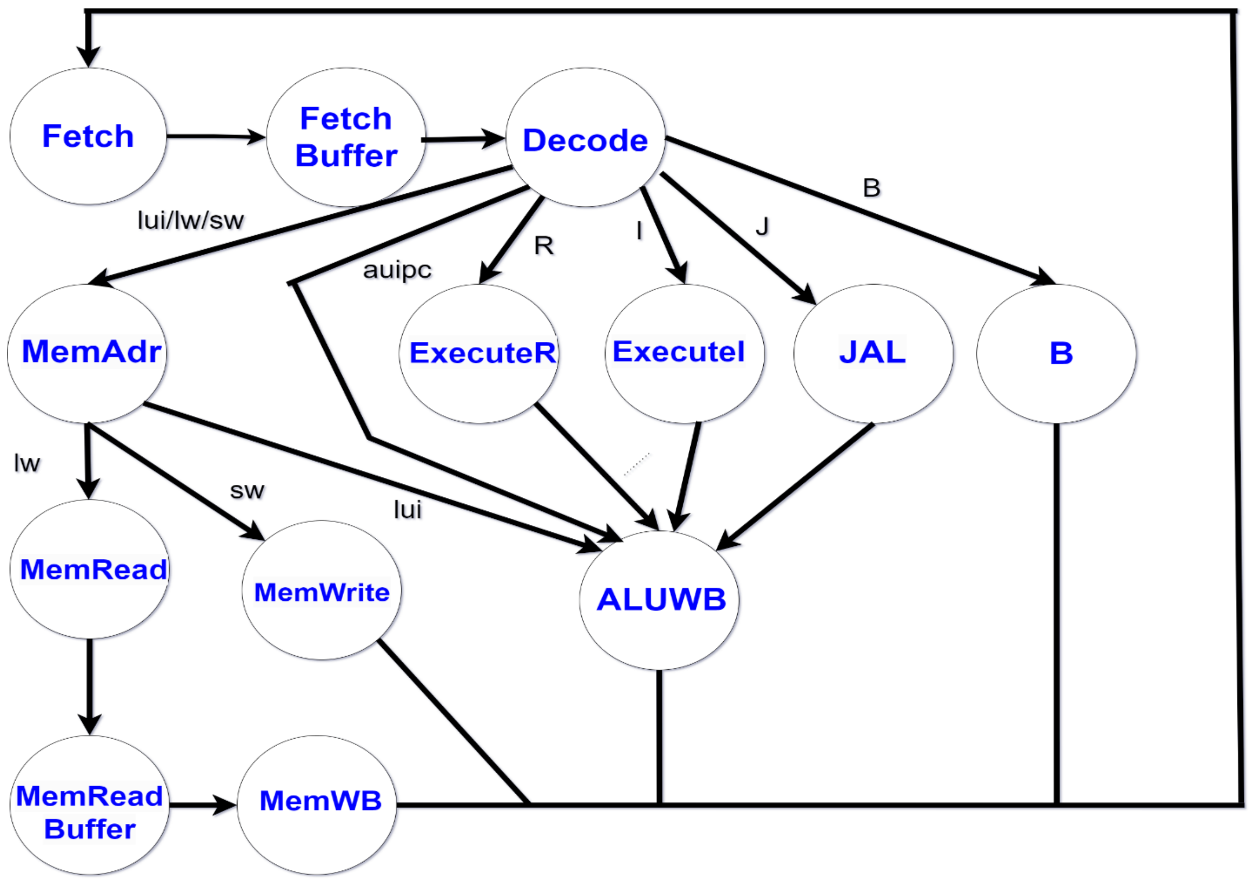


Figure: Datapath of Multicycle Core

To implement different types of load instructions(lb,lh,lw,lbu,lhu) we have added this block before loading in the register file :



FSM Chart of the Multicycle Core:



We have added two buffer state. Because memory is sequential we need two cycles for reading from memory and after memory write back stage a delay was needed for fetching of new instruction.

We have added all 37 instructions from “RV32 I Base Instruction Set”.

No of cycles required for various types of instructions

Load – 7 cycle

Store-5 cycle

U type-5 cycle

R type-5 cycle

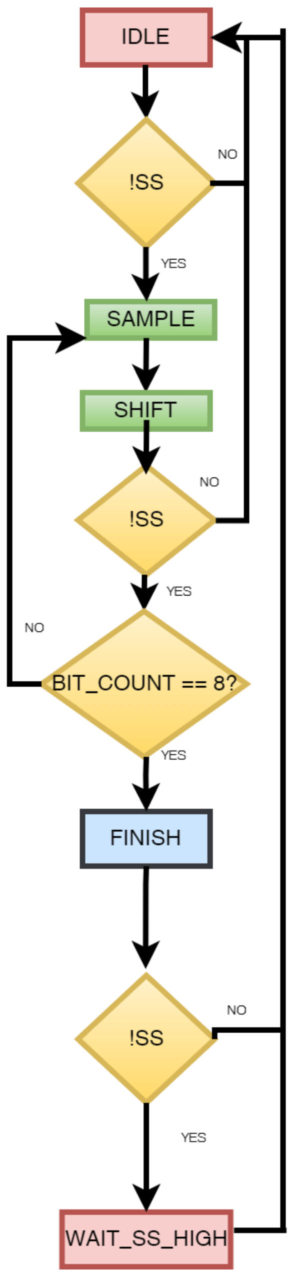
I type-5 cycle

J type-5 cycle

B type-4 cycle

Interface: We have used SPI for interface

FSM of SPI:



1. **RESET**
2. Slave Select **(SS)** pulled low (Active).
3. **SAMPLE** on positive edge, send the value of shift register on negative edge.
4. Check bit count, if bit count is 8, **FINISH**, else continue transfer.
5. Continue until **SS** is pulled high.

Memory Interface

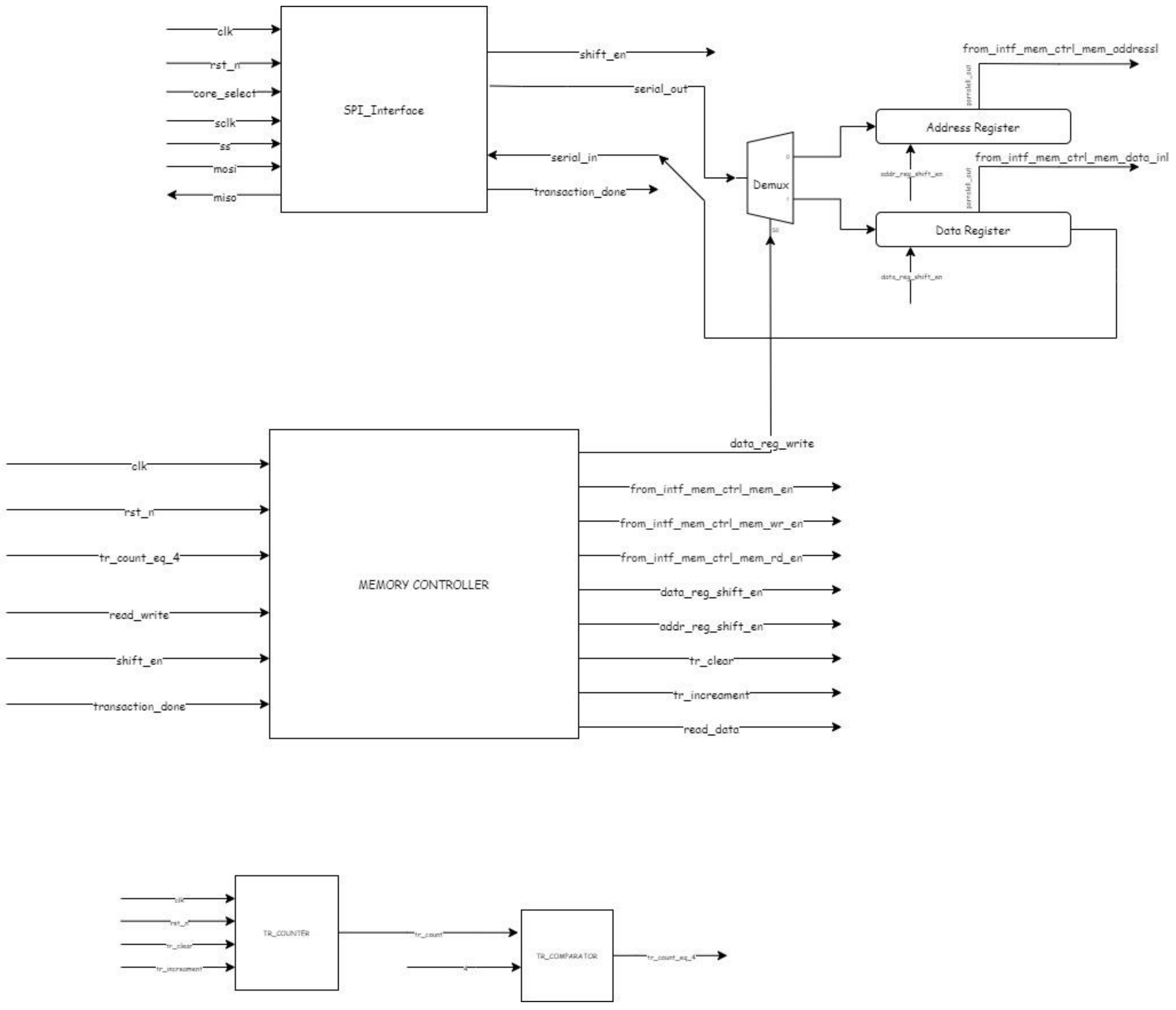


Figure: Microarchitecture of Memory Interface

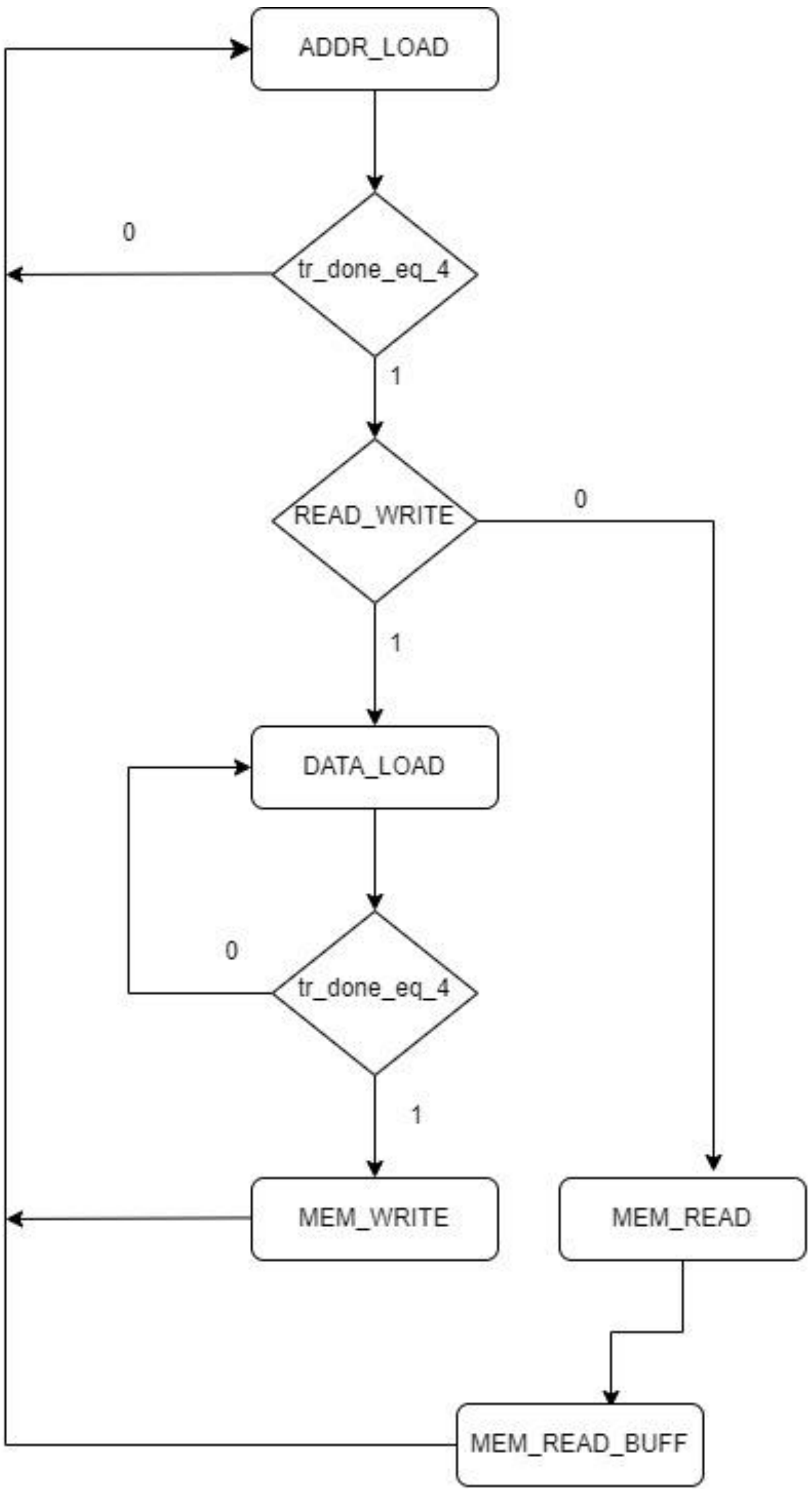


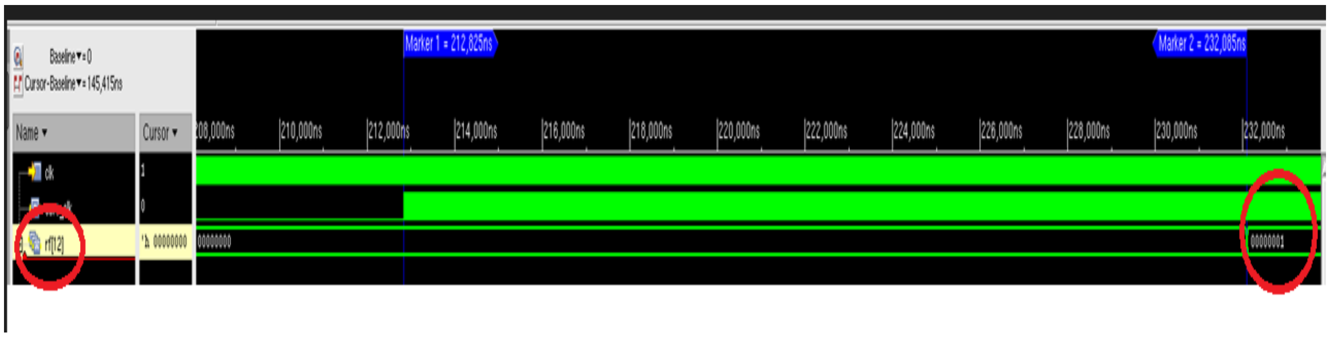
Figure: Flow Chart of Memory Controller

State Description

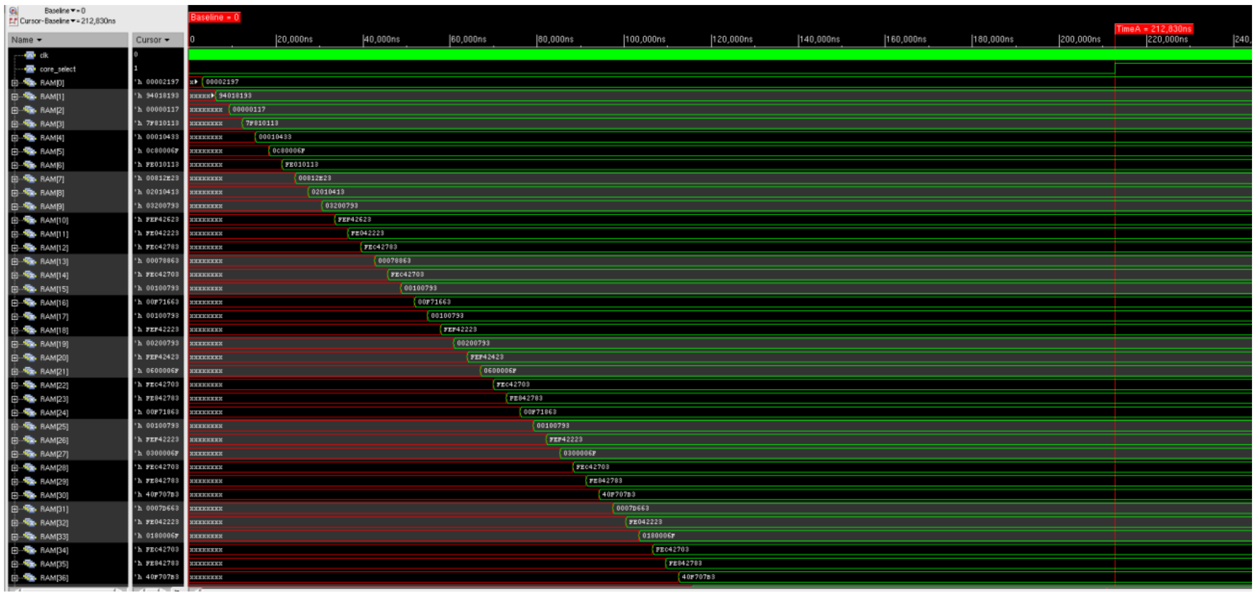
ADDR_LOAD	: 32-bit address is shifted to address register.
DATA_LOAD	: If this is a write request. Then Data is loaded in data register serially.
MEM_WRITE	: If Address register and data register is full then address and data is sent to memory. Memory enable signal and memory write signal is enabled in this state.
MEM_READ	: If a read request is got then memory read enable signal is generated and sent to memory.
MEM_READ_BUFF	: Data output from memory is loaded into data register which is then serially out via miso pin.

Output Waveform:

At the end, register r[12]=1 for the benchmarking instructions.



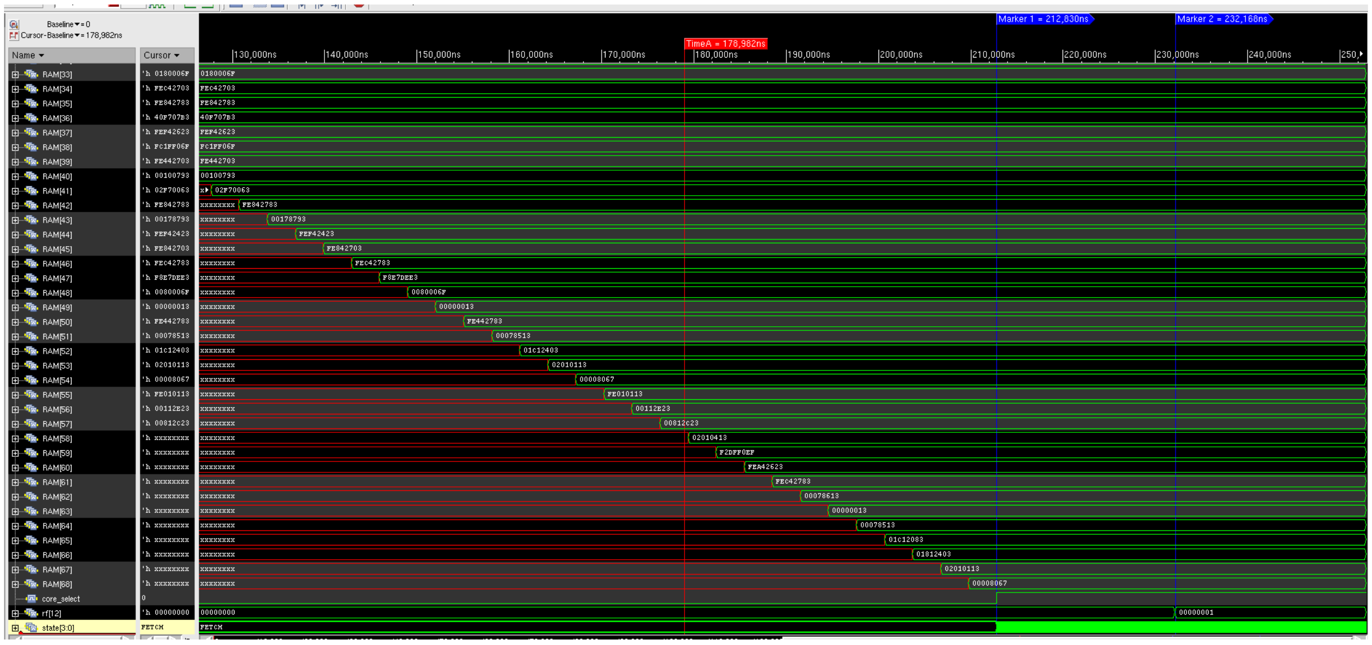
Uploading Instructions into the memory(core_select =0)



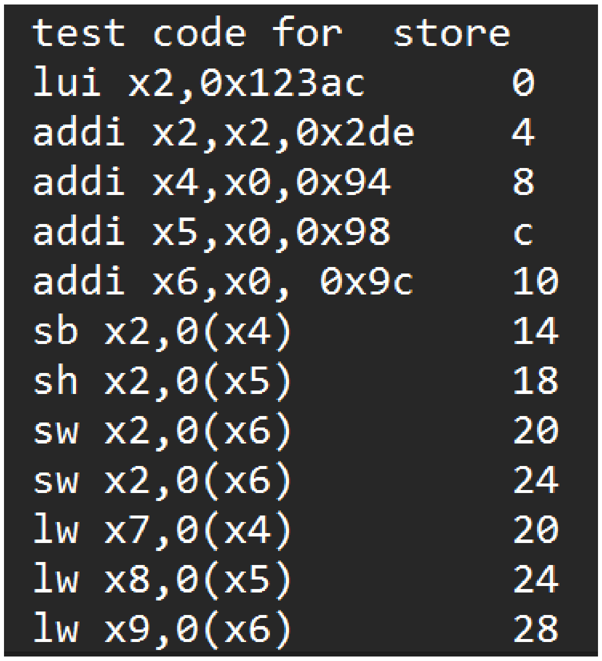
Total time required to execute benchmark instructions- 232168 ns or 0.232 ms

Core_select is high after 212,830 ns

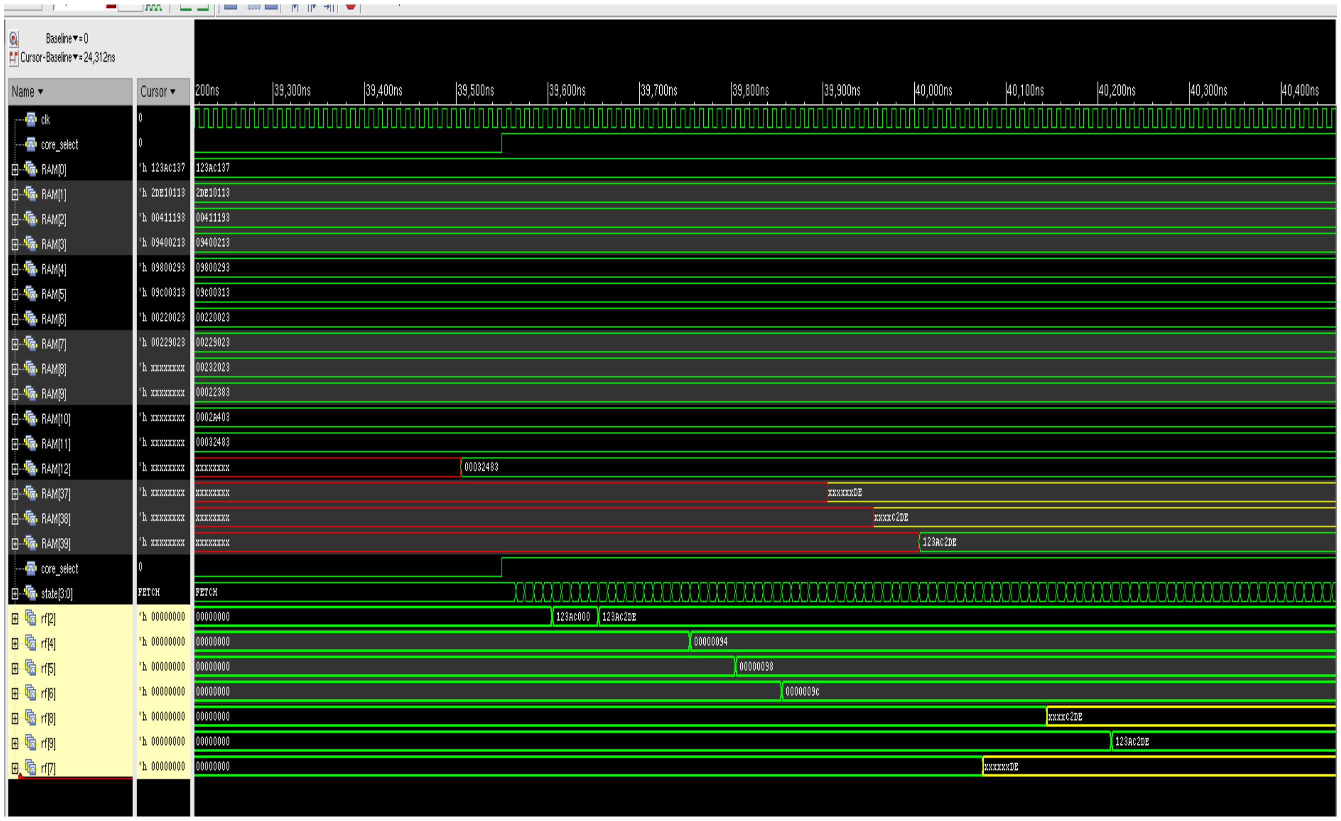
Core needs 19338 ns or 19.338 us



We have run some additional instructions



```
test code for  store
lui x2,0x123ac       0
addi x2,x2,0x2de     4
addi x4,x0,0x94      8
addi x5,x0,0x98      c
addi x6,x0, 0x9c     10
sb x2,0(x4)          14
sh x2,0(x5)          18
sw x2,0(x6)          20
sw x2,0(x6)          24
lw x7,0(x4)          20
lw x8,0(x5)          24
lw x9,0(x6)          28
```


Baseline▼=0
Cursor-Baseline▼=24,312ns
Name ▼
Cursor ▼
200ns
39,300ns
39,400ns
39,500ns
39,600ns
39,700ns
39,800ns
39,900ns
40,000ns
40,100ns
40,200ns
40,300ns
40,400ns
clk 0
core_select 0
RAM[0] 'h 123AC137 123AC137
RAM[1] 'h 2DE10113 2DE10113
RAM[2] 'h 00411193 00411193
RAM[3] 'h 09400213 09400213
RAM[4] 'h 09800293 09800293
RAM[5] 'h 09C00313 09C00313
RAM[6] 'h 00220023 00220023
RAM[7] 'h 00229023 00229023
RAM[8] 'h xxxxxxxx 00232023
RAM[9] 'h xxxxxxxx 00022383
RAM[10] 'h xxxxxxxx 0002A403
RAM[11] 'h xxxxxxxx 00032483
RAM[12] 'h xxxxxxxx xxxxxxxx 00032483
RAM[37] 'h xxxxxxxx xxxxxxxx xxxxxxDE
RAM[38] 'h xxxxxxxx xxxxxxxx xxxxC2DE
RAM[39] 'h xxxxxxxx xxxxxxxx 123AC2DE
core_select 0
state[3:0] FETCH FETCH
rf[2] 'h 00000000 00000000 123AC000 123AC2DE
rf[4] 'h 00000000 00000000 00000094
rf[5] 'h 00000000 00000000 00000098
rf[6] 'h 00000000 00000000 0000009c
rf[8] 'h 00000000 00000000 xxxxC2DE
rf[9] 'h 00000000 00000000 123AC2DE
rf[7] 'h 00000000 00000000 xxxxxxDE

Fibonacci series: After writing the assembly code from the C code, we have used an online tool to get the machines code.

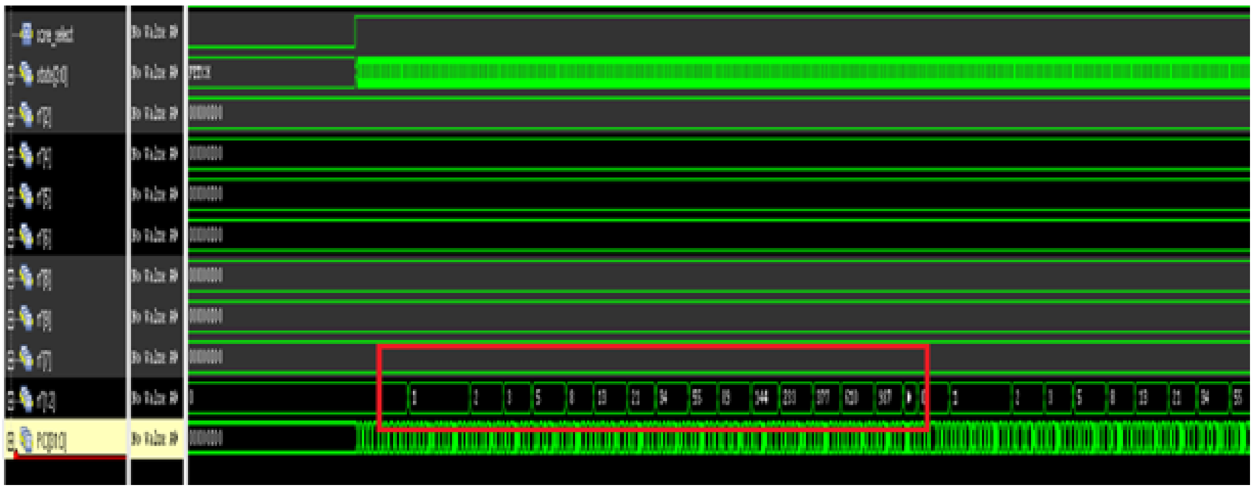
The website name:

https://riscvasm.lucasteske.dev/?fbclid=IwAR0iTMDT0CicpErRBfkV_5ZwE1Tv40GPUpSf07vOPMVB2emC3VVieIcCJOo

The machine codes are:

```
01100F13
01E02023
00000693
00000613
00000593
00100593
00002783
00060533
00168693
00B50633
FED780E3
0016F713
FE0706E3
000605B3
FE9FF0EF
```

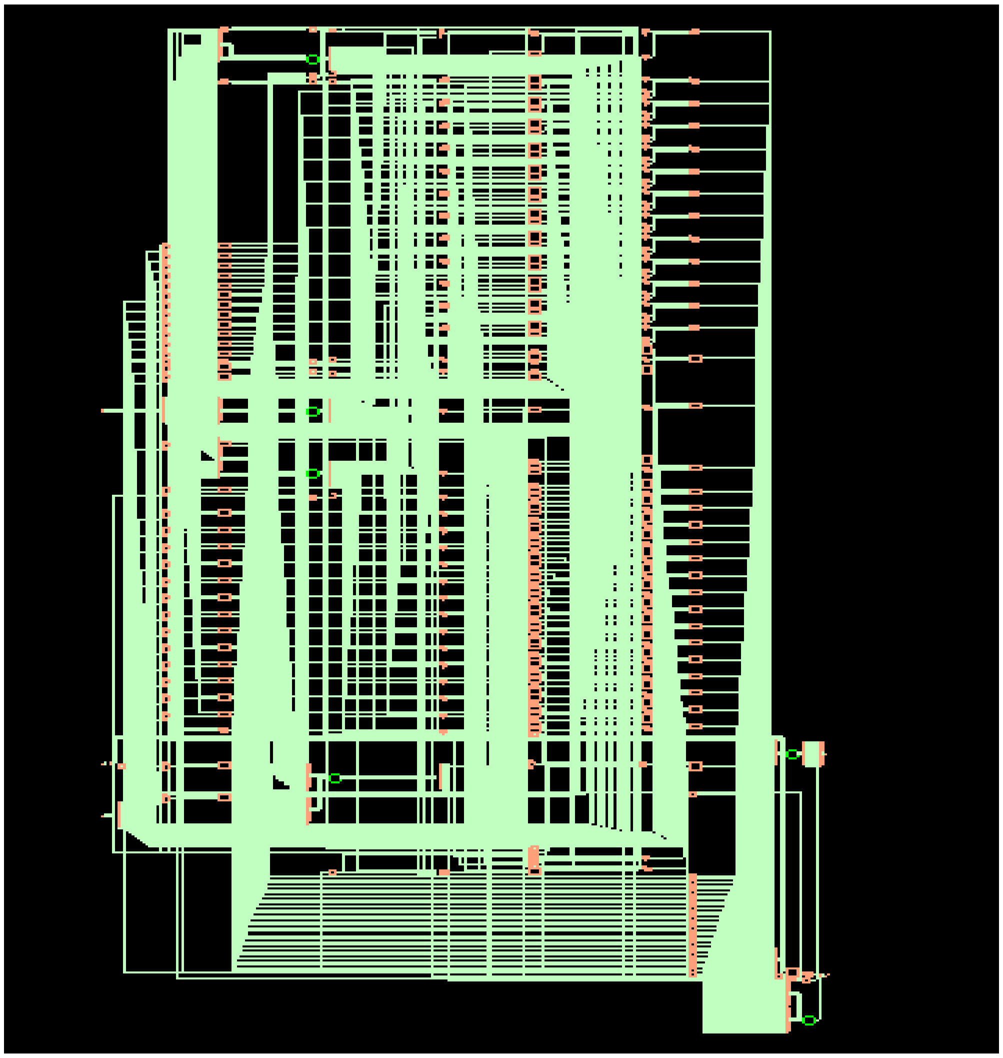
Output waveform:



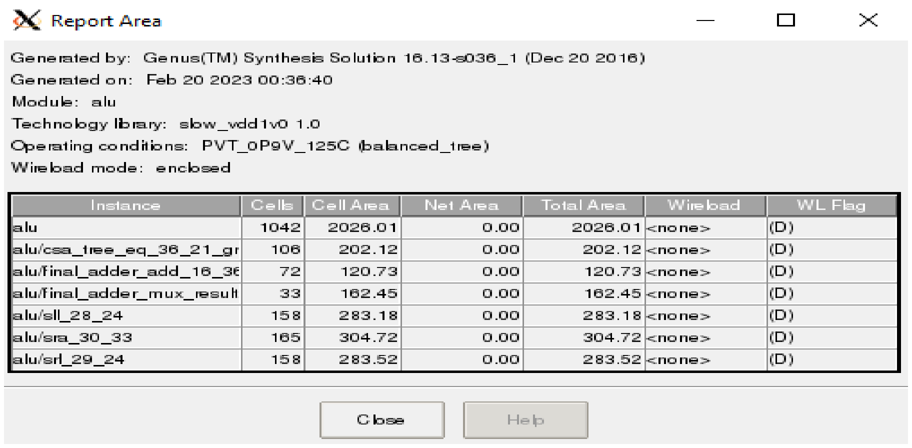
Synthesis:

We have synthesized some parts of the core rather than core. Due to unavailability of server for few days, we could not try to synthesis the whole core

ALU:



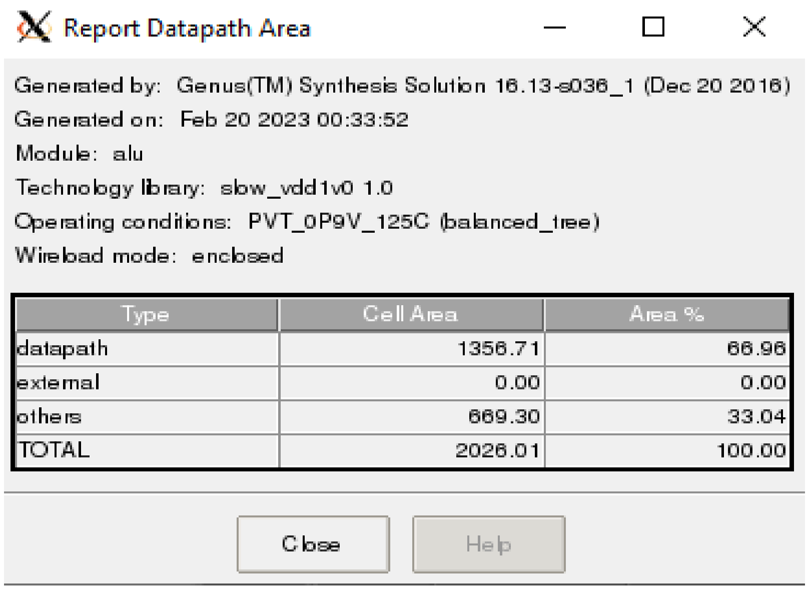
Report:



Report Area

Generated by: Genus(TM) Synthesis Solution 16.13-s036_1 (Dec 20 2016)
Generated on: Feb 20 2023 00:36:40
Module: alu
Technology library: slow_vdd1v0 1.0
Operating conditions: PVT_0P9V_125C (balanced_tree)
Wireload mode: enclosed

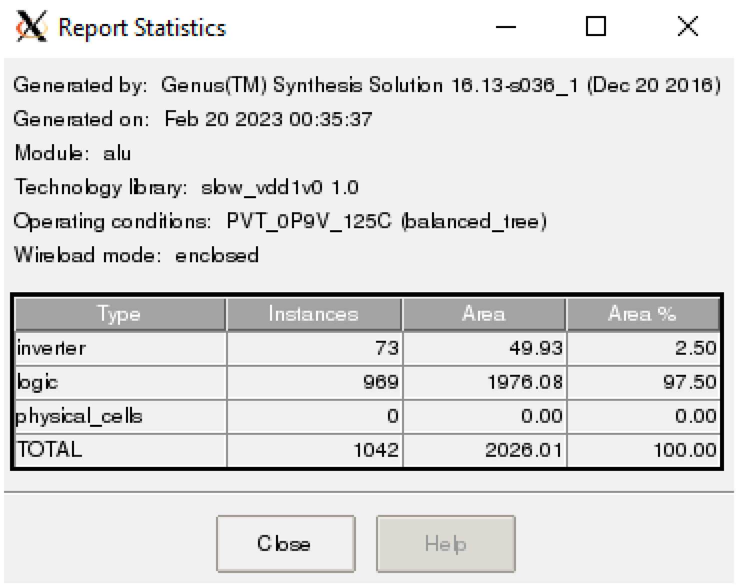
Instance	Cells	Cell Area	Net Area	Total Area	Wireload	WL Flag
alu	1042	2026.01	0.00	2026.01	<none>	(D)
alu/csa_tree_eq_36_21_gr	106	202.12	0.00	202.12	<none>	(D)
alu/final_adder_add_16_36	72	120.73	0.00	120.73	<none>	(D)
alu/final_adder_mux_result	33	162.45	0.00	162.45	<none>	(D)
alu/sll_28_24	158	283.18	0.00	283.18	<none>	(D)
alu/sra_30_33	165	304.72	0.00	304.72	<none>	(D)
alu/srl_29_24	158	283.52	0.00	283.52	<none>	(D)



Report Datapath Area

Generated by: Genus(TM) Synthesis Solution 16.13-s036_1 (Dec 20 2016)
Generated on: Feb 20 2023 00:33:52
Module: alu
Technology library: slow_vdd1v0 1.0
Operating conditions: PVT_0P9V_125C (balanced_tree)
Wireload mode: enclosed

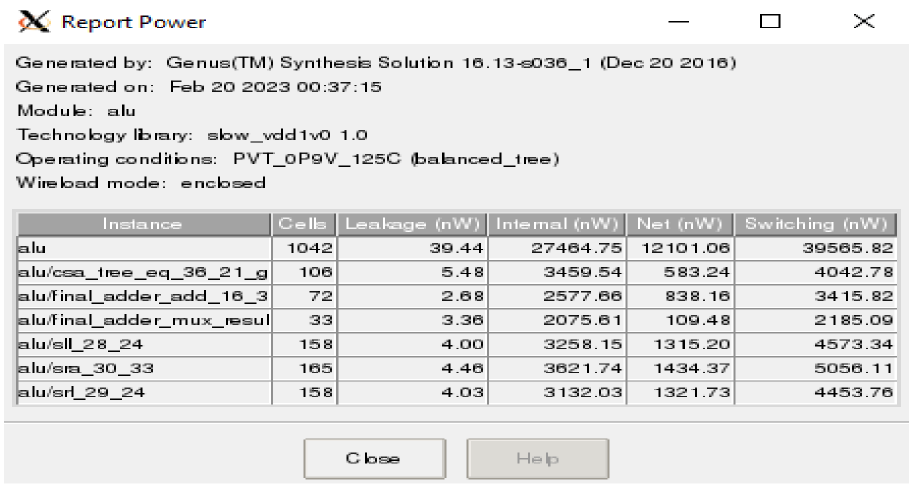
Type	Cell Area	Area %
datapath	1356.71	66.96
external	0.00	0.00
others	669.30	33.04
TOTAL	2026.01	100.00



Report Statistics

Generated by: Genus(TM) Synthesis Solution 16.13-s036_1 (Dec 20 2016)
Generated on: Feb 20 2023 00:35:37
Module: alu
Technology library: slow_vdd1v0 1.0
Operating conditions: PVT_0P9V_125C (balanced_tree)
Wireload mode: enclosed

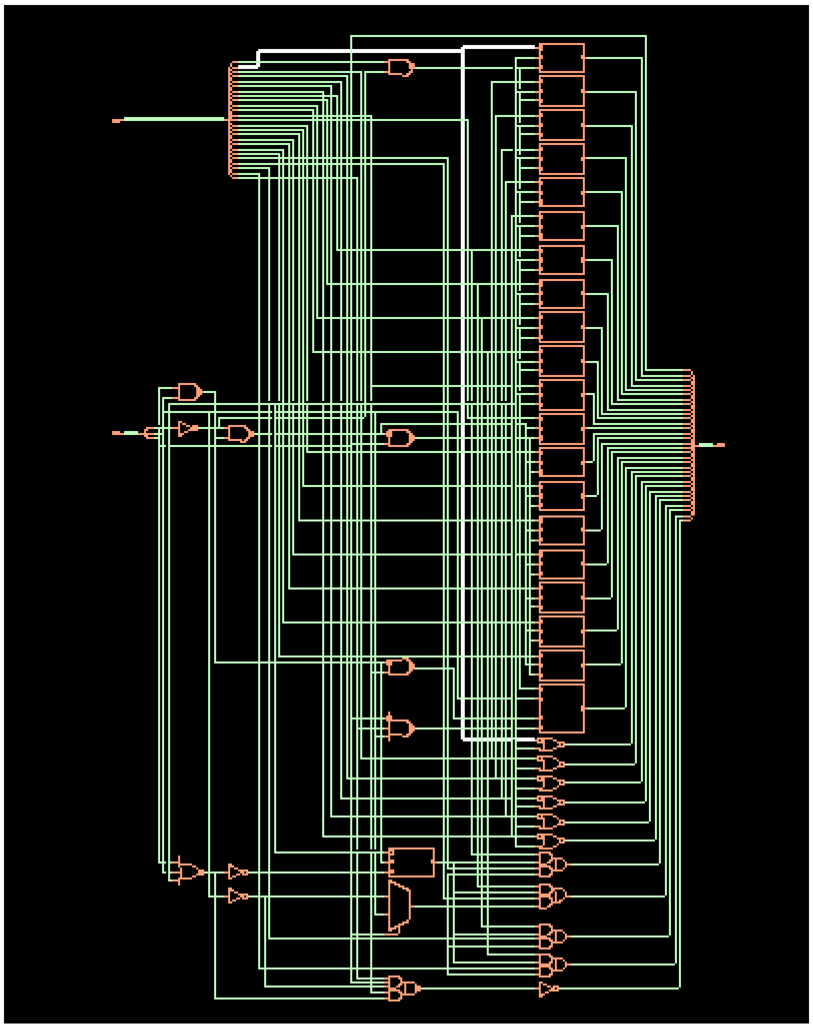
Type	Instances	Area	Area %
inverter	73	49.93	2.50
logic	969	1976.08	97.50
physical_cells	0	0.00	0.00
TOTAL	1042	2026.01	100.00



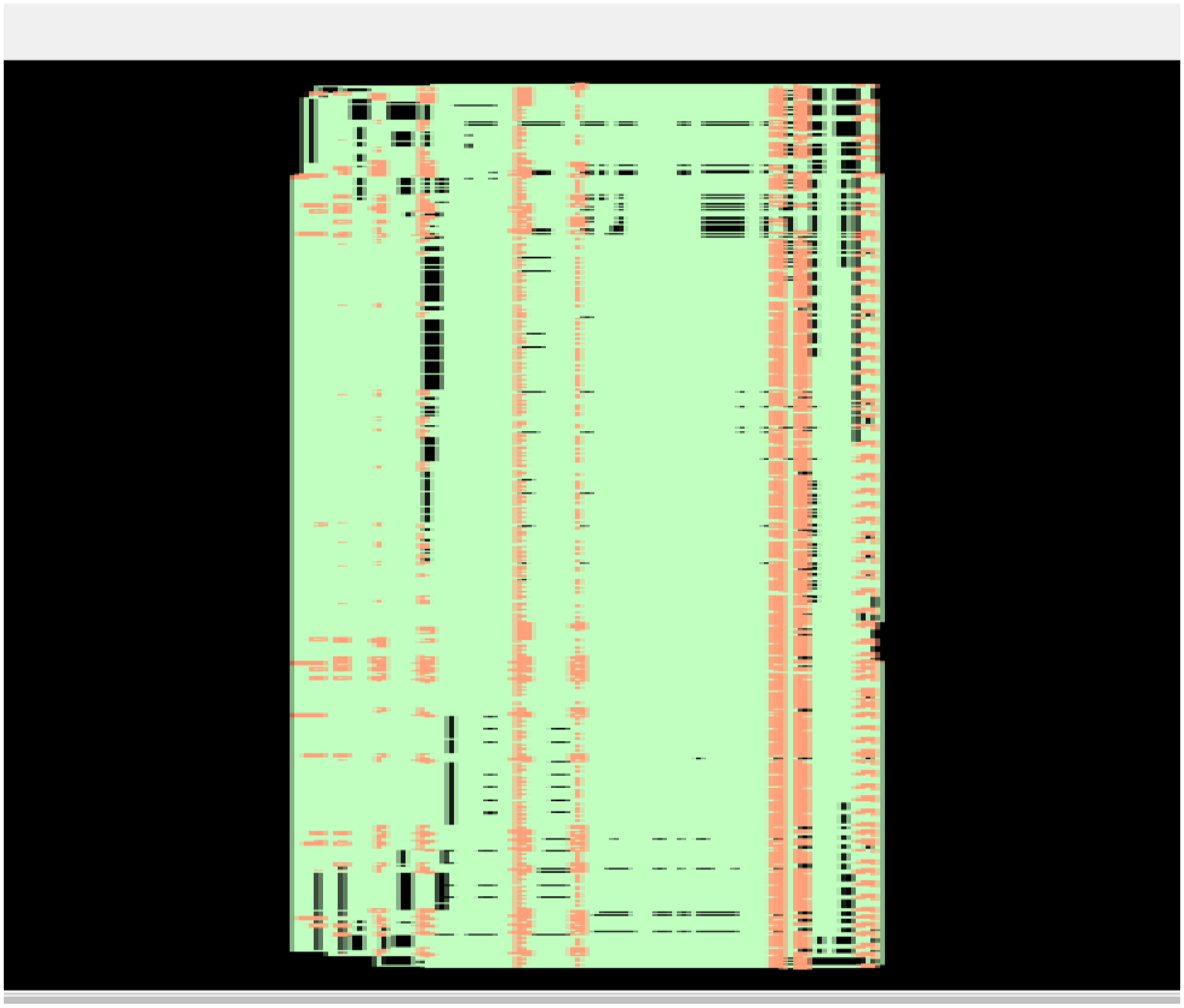
Generated by: Genus(TM) Synthesis Solution 16.13-s036_1 (Dec 20 2016)
Generated on: Feb 20 2023 00:37:15
Module: alu
Technology library: slow_vdd1v0 1.0
Operating conditions: PVT_0P9V_125C (balanced_tree)
Wireload mode: enclosed

Instance	Cells	Leakage (nW)	Internal (nW)	Net (nW)	Switching (nW)
alu	1042	39.44	27464.75	12101.06	39565.82
alu/csa_tree_eq_36_21_g	106	5.48	3459.54	583.24	4042.78
alu/final_adder_add_16_3	72	2.68	2577.66	838.16	3415.82
alu/final_adder_mux_resul	33	3.36	2075.61	109.48	2185.09
alu/sll_28_24	158	4.00	3258.15	1315.20	4573.34
alu/sra_30_33	165	4.46	3621.74	1434.37	5056.11
alu/srl_29_24	158	4.03	3132.03	1321.73	4453.76

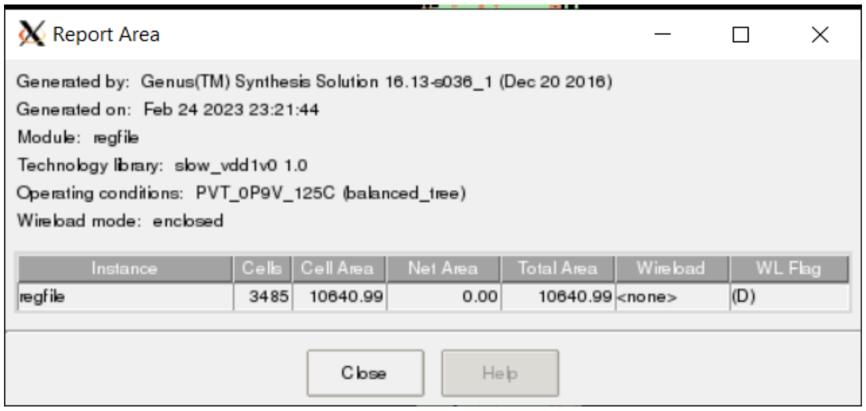
Extend:



Register File:

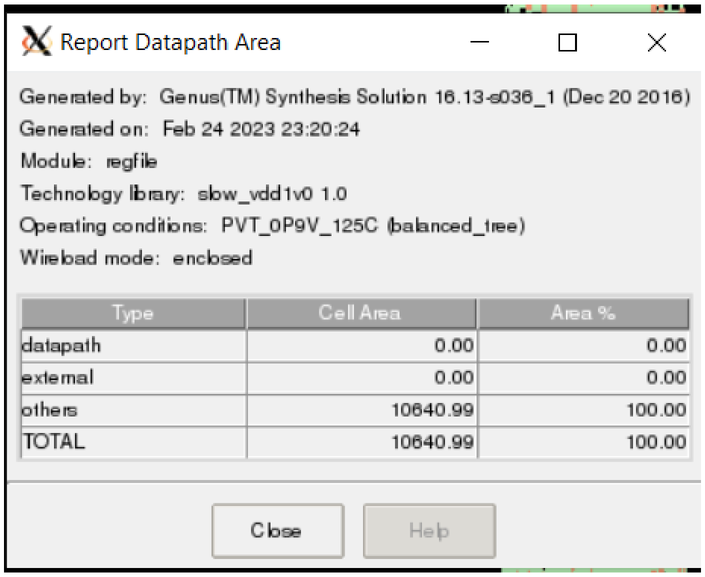


Report:



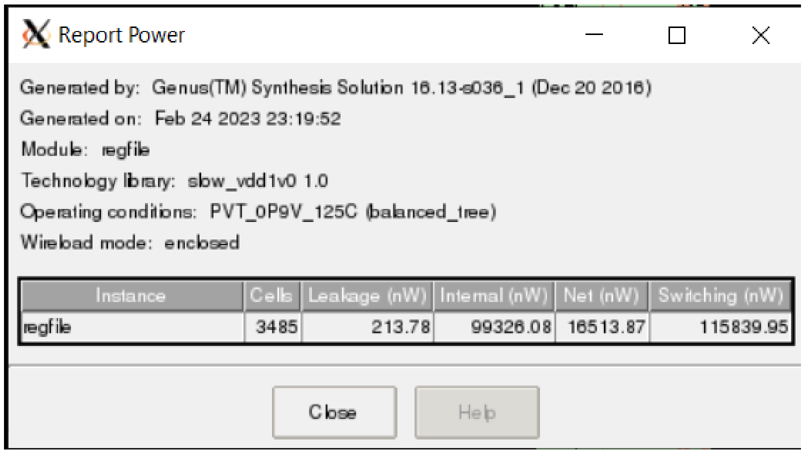
```
Generated by:  Genus(TM) Synthesis Solution 16.13-s036_1 (Dec 20 2016)
Generated on:  Feb 24 2023 23:21:44
Module:  regfile
Technology library:  slow_vdd1v0 1.0
Operating conditions:  PVT_0P9V_125C (balanced_tree)
Wireload mode:  enclosed
```

Instance	Cells	Cell Area	Net Area	Total Area	Wireload	WL Flag
regfile	3485	10640.99	0.00	10640.99	<none>	(D)


Report Datapath Area

Generated by: Genus(TM) Synthesis Solution 16.13-s036_1 (Dec 20 2016)
Generated on: Feb 24 2023 23:20:24
Module: regfile
Technology library: slow_vdd1v0 1.0
Operating conditions: PVT_0P9V_125C (balanced_tree)
Wireload mode: enclosed

Type	Cell Area	Area %
datapath	0.00	0.00
external	0.00	0.00
others	10640.99	100.00
TOTAL	10640.99	100.00


Report Power

Generated by: Genus(TM) Synthesis Solution 16.13-s036_1 (Dec 20 2016)
Generated on: Feb 24 2023 23:19:52
Module: regfile
Technology library: slow_vdd1v0 1.0
Operating conditions: PVT_0P9V_125C (balanced_tree)
Wireload mode: enclosed

Instance	Cells	Leakage (nW)	Internal (nW)	Net (nW)	Switching (nW)
regfile	3485	213.78	99326.08	16513.87	115839.95