VLSI Circuit Design II– EHB 425E

HOMEWORK V

Yiğit Bektaş GÜRSOY

040180063

Rana TİLKİ

040180741

Class Lecturer: Sıddıka Berna Örs Yalçın

Class Assistant:
Yasin Fırat Kula

1- Instructions and Descriptions

Mnemonic	Name	Description
LUI	Load Upper Immediate	rd 🡐 imm << 12
AUIPC	Add Upper Immediate To Pc	rd 🡐 program counter(pc) + imm << 12
JAL	Jump And Link	rd 🡐 {imm[20], imm[10:1], imm[11], imm[19:12]}
JALR	Jump and Link Register	rs1 🡐 imm[11:0] then rs1[0] 🡐 0 rd 🡐 pc + 4 pc 🡐 pc + imm

BEQ	Branch If Equal	if rs1 = rs2, pc 🡐 pc + imm
		else pc 🡐 pc + 4

BNE	Branch If Not Equal	if rs1 != rs2, pc 🡐 pc + imm
		else pc = pc + 4

BLT	Branch If Less Than	if rs1 < rs2, pc 🡐 pc + imm
		else pc = pc + 4

BGE	Branch If Greater or Equal	if rs1 >= rs2, pc 🡐 pc + imm
		else pc = pc + 4

BLTU	Branch If Less Than Unsigned	if rs1 < rs2, pc 🡐 pc + imm
		else pc = pc + 4

BGEU	Branch If Greater Than Unsigned	if rs1 >= rs2, pc 🡐 pc + imm
		else pc = pc + 4

LB	Load Byte	address 🡐 rs1 + imm[7:0] byte_value 🡐 Memory[address] rd 🡐 sxt(byte_value)
LH	Load Halfword	address 🡐 rs1 + imm[15:0] byte_value 🡐 Memory[address] rd 🡐 sxt(byte_value)
LW	Load Word	address 🡐 rs1 + imm[31:0] byte_value 🡐 Memory[address] rd 🡐 sxt(byte_value)
LBU	Load Byte Unsigned	address 🡐 rs1 + imm[7:0] byte_value 🡐 Memory[address] rd 🡐 zxt(byte_value) (zxt - unsigned extended
LHU	Load Halfword Unsigned	address 🡐 rs1 + imm[15:0] byte_value 🡐 Memory[address] rd 🡐 zxt(byte_value)
SB	Store Byte	address 🡐 rs1 + imm Memory[address] 🡐rs2[7:0]
SH	Store Halfword	address 🡐 rs1 + imm Memory[address] 🡐 rs2[15:0]
SW	Store Word	address 🡐rs1 + imm Memory[address] 🡐rs2[31:0]
ADDI	Add Immediate	rd 🡐rs1 + sxt(imm)

SLTI	Set Less Than Immediate	if rs1<sxt(imm) then rd 🡐1
		else rs 🡐 0

SLTIU	Set Less Than Immediate Unsigned	if rs1 🡐 zxt(imm), rd 🡐1
		else rd 🡐 0

XORI	XOR Immediate	rd 🡐 rs1 ^ imm
ORI	OR Immediate	rd 🡐 rs1 | imm
ANDI	AND Immediate	rd 🡐 rs1 & imm
SLLI	Shift Left Logical Immediate	rd 🡐 rs1 << shamt
SRLI	Shift Right Logical Immediate	rd 🡐 rs1 >> shamt
SRAI	Shift Right Arithmetic Immediate	rd 🡐 rs1 >>> shamt
ADD	Addition	rd 🡐 rs1 + rs2
SUB	Substraction	rd 🡐 rs1 – rs2
SLL	Shift Left Logical	rd 🡐 rs1 << rs2[4:0]

SLT	Set Less Than	if rs1 < rs2 then rd 🡐 1
		else rd 🡐 0

SLTU	Set Less Than Unsigned	if rs1[31:0] < rs2[31:0] then rd 🡐 1
		else rd 🡐 0

XOR	Exclusive OR	rd 🡐 rs1 ^ rs2
SRL	Shift Right Logical	rd 🡐 rs1 >> rs2[4:0]
SRA	Shift Right Arithmetic	rd 🡐 rs1 >>> rs2[4:0]
OR	OR	rd 🡐 rs1 | rs2
AND	AND	rd 🡐 rs1 & rs2

2- Machine Codes

1-)ADDI: 000000000000_00000_000_00001_0010011

I-TYPE imm[11:0] rs1 funct3 rd opcode

imm[11:0] = 000000000000, rs1 = 00000, funct3 = 000, rd = 00001, opcode = 0010011

Hexadecimal : imm[11:0] = 0, rs1 = 0, rd=1

2-)BEQ: 0000000_00010_00011_000_00010_0001100

B-TYPE imm[12|10:5] rs2 rs1 funct3 imm[4:1|11] opcode

imm[12:1] = 000000000001

imm[12|10:5] = 0000000, rs2 = 00010, rs1 = 00011, funct3 = 000, imm[4:1|11] = 00010, opcode = 1100011

Hexadecimal : imm[12:1] = 1, rs2 = 2, rs1=3

3-)LW : 000000000010_00100_010_00101_0000011

I-TYPE imm[11:0] rs1 funct3 rd opcode

imm[11:0] = 000000000010, rs1 = 00100, funct3 = 010, rd = 00101, opcode = 0000011

Hexadecimal: imm[11:0] = 2, rs1=4, rd=5

4-)SW : 0000000_00110_00111_010_00011_0100011

S-TYPE imm[11:5] rs2 rs1 funct3 imm[4:0] opcode

imm[11:0] = 000000000011

imm[11:5] = 0000000, rs2 = 00110, rs1 = 00111, funct3 = 010, imm[4:0] = 00011, opcode = 0100011

Hexadecimal: imm[11:0] = 3, rs2=6, rs1=7

5-)BNE: 0000000_01000_01001_001_01000_1100011

B-TYPE imm[12|10:5] rs2 rs1 funct3 imm[4:1|11] opcode imm[12:1] = 000000000100

imm[12|10:5] = 0000000, rs2 = 01000, rs1 = 01001, funct3 = 001, imm[4:1|11] = 01000, opcode = 1100011

Hexadecimal : imm[12:1] = 4, rs2 = 8, rs1=9

6-)BLT: 0000000_01010_01011_100_01010_1100011

B-TYPE imm[12|10:5] rs2 rs1 funct3 imm[4:1|11] opcode

imm[12:1] = 000000000101

imm[12|10:5] = 0000000, rs2 = 01010, rs1 = 01011, funct3 = 100, imm[4:1|11] = 01010, opcode = 1100011

Hexadecimal: imm[12:1] = 5, rs2 = 10, rs1=11

7-)BGE: 0000000_01100_01101_101_01100_1100011

B-TYPE imm[12|10:5] rs2 rs1 funct3 imm[4:1|11] opcode

imm[12:1] = 000000000110

imm[12|10:5] = 0000000, rs2 = 01100, rs1 = 01101, funct3 = 101, imm[4:1|11] = 01100, opcode = 1100011

Hexadecimal: imm[12:1] = 6, rs2 = 12, rs1=13

8-)BLTU: 0000000_01110_01111_110_01110_1100011

B-TYPE imm[12|10:5] rs2 rs1 funct3 imm[4:1|11] opcode

imm[12:1] = 000000000111

imm[12|10:5] = 0000000, rs2 = 01110, rs1 = 01111, funct3 = 110, imm[4:1|11] = 01110, opcode = 1100011

Hexadecimal: imm[12:1] = 7, rs2 = 14, rs1=15

9-)BGEU: 0000000_10000_10001_111_10000_1100011

B-TYPE imm[12|10:5] rs2 rs1 funct3 imm[4:1|11] opcode

imm[12:1] = 000000001000

imm[12|10:5] = 0000000, rs2 = 10000, rs1 = 10001, funct3 = 111, imm[4:1|11] = 10000, opcode = 1100011

Hexadecimal: imm[12:1] = 8, rs2 = 16, rs1=17

10-)XOR: 0000000_10010_10011_100_10100_0110011

R-TYPE func7 rs2 rs1 func3 rd opcode

func7 = 0000000, rs2 = 10010, rs1 = 10011, func3 = 100, rd = 10100, opcode = 0110011

Hexadecimal: rs2 = 18, rs1=19, rd=20

11-)OR: 0000000_10101_10110_110_10111_0110011

R-TYPE func7 rs2 rs1 func3 rd opcode

func7 = 0000000, rs2 = 10101, rs1 = 10110, func3 = 110, rd = 10111, opcode = 0110011

Hexadecimal: rs2 = 21, rs1=22, rd=23

12-)AND: 0000000_11000_11001_111_11010_0110011

R-TYPE func7 rs2 rs1 func3 rd opcode

func7 = 0000000, rs2 = 11000, rs1 = 11001, func3 = 111, rd = 11010, opcode = 0110011

Hexadecimal: rs2 = 24, rs1=25, rd=26

13-)SUB: 0100000_11011_11100_000_11101_0110011

R-TYPE func7 rs2 rs1 func3 rd opcode

func7 = 0100000, rs2 = 11011, rs1 = 11100, func3 = 000, rd = 11101, opcode = 0110011

Hexadecimal: rs2 = 27, rs1=28, rd=29

14-)JAL: 00000001001000000000_11110_1101111

J-TYPE imm[20:1], rd, opcode

imm[20:1]= 00000000000000001001, rd = 11110, opcode = 1101111

Hexadecimal: imm[20:1]=9, rd=30

15-)AUIPC: 00000000000000001010_11111_0010111

U-TYPE imm[20:1], rd, opcode

imm[31:12]= 00000000000000001010, rd = 11111, opcode = 0010111

Hexadecimal: imm[31:12]=10, rd=31

16-)SRA: 0100000_00000_00001_101_00010_0110011

R-TYPE func7 rs2 rs1 func3 rd opcode

func7 = 0100000, rs2 = 00000, rs1 = 00001, func3 = 101, rd = 00010, opcode = 0110011

Hexadecimal: rs2 = 0, rs1=1, rd=2

In RISC-V, the NOP (no operation) instruction is performed using the ADDI (add immediate) instruction with both source operands set to the zero register (x0). Since adding zero to a register doesn't change its value, the ADDI instruction effectively performs no operation.

NOP: 00000000000000000000000000010011

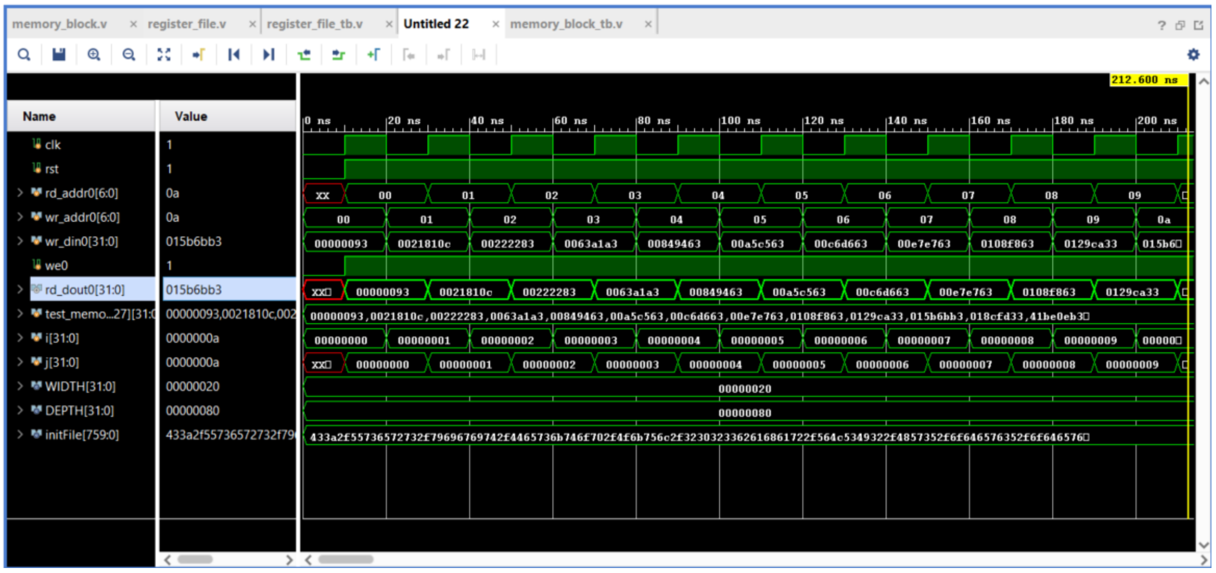
imm: 0000000000000000, rd: 00000, funct3: 000, rs1: 00000, opcode:0010011

In this encoding, the opcode field is `0010011`, which corresponds to the ADDI instruction. The rs1 and rs2 fields are both set to the zero register (x0), and the immediate field is set to zero. Since adding zero to a register doesn't change its value, this instruction effectively performs no operation.

3- Behavioral Simulation Results

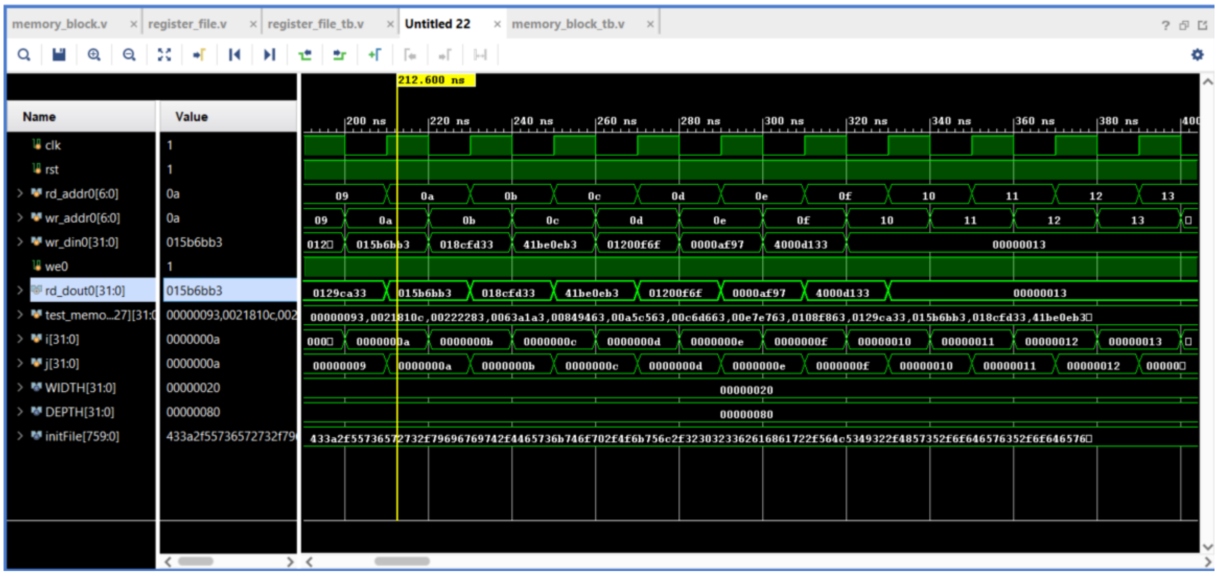
The codes are in the zip file. Only simulation results and comments are available here.

- **Memory Block**
  The simulation results and TCL console outputs are shown below. 32bit data consisting of 128 lines in the "machine_code.txt" file is read through the memory_block.v file. With our "we0" signal being 1, the data starts to be written to the memory block. When wr_din0 and rd_dout signals are followed, it appears to be read successfully. After the first 16 lines are read, the remaining 112 lines are filled with "no action" machine code. The results are verified when comparing the memory written with the Tcl console outputs.

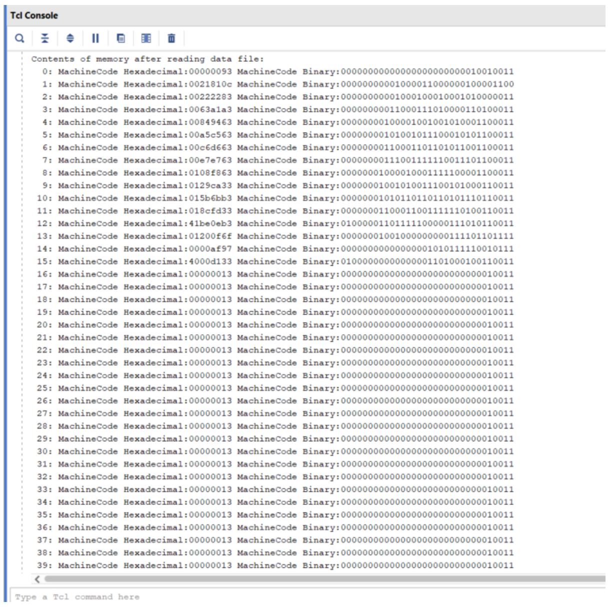


Name	Value
clk	1
rst	1
rd_addr0[6:0]	0a
wr_addr0[6:0]	0a
wr_din0[31:0]	015b6bb3
we0	1
rd_dout0[31:0]	015b6bb3
test_memo...27][31:0	00000093,0021810c,002
i[31:0]	0000000a
j[31:0]	0000000a
WIDTH[31:0]	00000020
DEPTH[31:0]	00000080
initFile[759:0]	433a2f55736572732f79

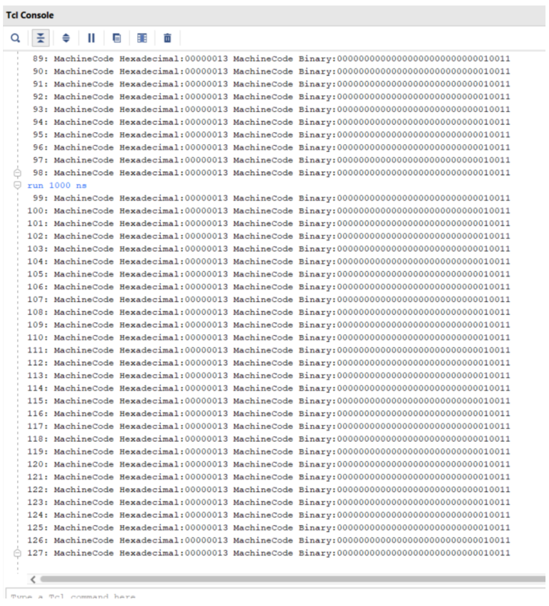
```
rd_addr0:  XX 00 01 02 03 04 05 06 07 08 09
wr_addr0:  00 01 02 03 04 05 06 07 08 09 0a
wr_din0:   00000093 0021810c 00222283 0063a1a3 00849463 00a5c563 00c6d663 00e7e763 0108f863 0129ca33 015b6
rd_dout0:  XX 00000093 0021810c 00222283 0063a1a3 00849463 00a5c563 00c6d663 00e7e763 0108f863 0129ca33
test_memo: 00000093,0021810c,00222283,0063a1a3,00849463,00a5c563,00c6d663,00e7e763,0108f863,0129ca33,015b6bb3,018cfd33,41be0eb3
i:         00000000 00000001 00000002 00000003 00000004 00000005 00000006 00000007 00000008 00000009 00000
j:         XX 00000000 00000001 00000002 00000003 00000004 00000005 00000006 00000007 00000008 00000009
WIDTH:     00000020
DEPTH:     00000080
initFile:  433a2f55736572732f79696769742f4465736b746f702f4f6b756c2f32303233362616861722f564c5349322f4857352f6f646576352f6f646576
```



```
Name                    Value
clk                     1
rst                     1
rd_addr0[6:0]           0a
wr_addr0[6:0]           0a
wr_din0[31:0]           015b6bb3
we0                     1
rd_dout0[31:0]          015b6bb3
test_memo...27][31:0    00000093,0021810c,002
i[31:0]                 0000000a
j[31:0]                 0000000a
WIDTH[31:0]             00000020
DEPTH[31:0]             00000080
initFile[759:0]         433a2f55736572732f79
```

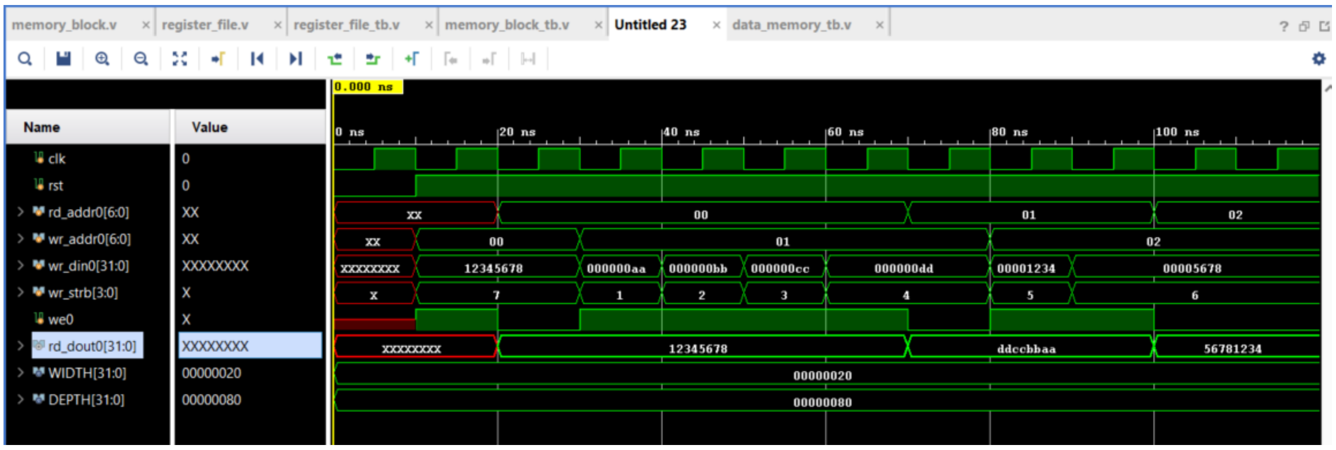


```
Contents of memory after reading data file:
  0: MachineCode Hexadecimal:00000093 MachineCode Binary:00000000000000000000000010010011
  1: MachineCode Hexadecimal:0021810c MachineCode Binary:00000000001000011000000100001100
  2: MachineCode Hexadecimal:00222283 MachineCode Binary:00000000001000100010001010000011
  3: MachineCode Hexadecimal:0063a1a3 MachineCode Binary:00000000011000111010000110100011
  4: MachineCode Hexadecimal:00849463 MachineCode Binary:00000000100001001001010001100011
  5: MachineCode Hexadecimal:00a5c563 MachineCode Binary:00000000101001011100010101100011
  6: MachineCode Hexadecimal:00c6d663 MachineCode Binary:00000000110001101101011001100011
  7: MachineCode Hexadecimal:00e7e763 MachineCode Binary:00000000111001111110011101100011
  8: MachineCode Hexadecimal:0108f863 MachineCode Binary:00000001000010001111100001100011
  9: MachineCode Hexadecimal:0129ca33 MachineCode Binary:00000001001010011100101000110011
 10: MachineCode Hexadecimal:015b6bb3 MachineCode Binary:00000001010110110110101110110011
 11: MachineCode Hexadecimal:018cfd33 MachineCode Binary:00000001100011001111110100110011
 12: MachineCode Hexadecimal:41be0eb3 MachineCode Binary:01000001101111100000111010110011
 13: MachineCode Hexadecimal:01200f6f MachineCode Binary:00000001001000000000111101101111
 14: MachineCode Hexadecimal:0000af97 MachineCode Binary:00000000000000001010111110010111
 15: MachineCode Hexadecimal:4000d133 MachineCode Binary:01000000000000001101000100110011
 16: MachineCode Hexadecimal:00000013 MachineCode Binary:00000000000000000000000000010011
 17: MachineCode Hexadecimal:00000013 MachineCode Binary:00000000000000000000000000010011
 18: MachineCode Hexadecimal:00000013 MachineCode Binary:00000000000000000000000000010011
 19: MachineCode Hexadecimal:00000013 MachineCode Binary:00000000000000000000000000010011
 20: MachineCode Hexadecimal:00000013 MachineCode Binary:00000000000000000000000000010011
 21: MachineCode Hexadecimal:00000013 MachineCode Binary:00000000000000000000000000010011
 22: MachineCode Hexadecimal:00000013 MachineCode Binary:00000000000000000000000000010011
 23: MachineCode Hexadecimal:00000013 MachineCode Binary:00000000000000000000000000010011
 24: MachineCode Hexadecimal:00000013 MachineCode Binary:00000000000000000000000000010011
 25: MachineCode Hexadecimal:00000013 MachineCode Binary:00000000000000000000000000010011
 26: MachineCode Hexadecimal:00000013 MachineCode Binary:00000000000000000000000000010011
 27: MachineCode Hexadecimal:00000013 MachineCode Binary:00000000000000000000000000010011
 28: MachineCode Hexadecimal:00000013 MachineCode Binary:00000000000000000000000000010011
 29: MachineCode Hexadecimal:00000013 MachineCode Binary:00000000000000000000000000010011
 30: MachineCode Hexadecimal:00000013 MachineCode Binary:00000000000000000000000000010011
 31: MachineCode Hexadecimal:00000013 MachineCode Binary:00000000000000000000000000010011
 32: MachineCode Hexadecimal:00000013 MachineCode Binary:00000000000000000000000000010011
 33: MachineCode Hexadecimal:00000013 MachineCode Binary:00000000000000000000000000010011
 34: MachineCode Hexadecimal:00000013 MachineCode Binary:00000000000000000000000000010011
 35: MachineCode Hexadecimal:00000013 MachineCode Binary:00000000000000000000000000010011
 36: MachineCode Hexadecimal:00000013 MachineCode Binary:00000000000000000000000000010011
 37: MachineCode Hexadecimal:00000013 MachineCode Binary:00000000000000000000000000010011
 38: MachineCode Hexadecimal:00000013 MachineCode Binary:00000000000000000000000000010011
 39: MachineCode Hexadecimal:00000013 MachineCode Binary:00000000000000000000000000010011
Type a Tcl command here
```

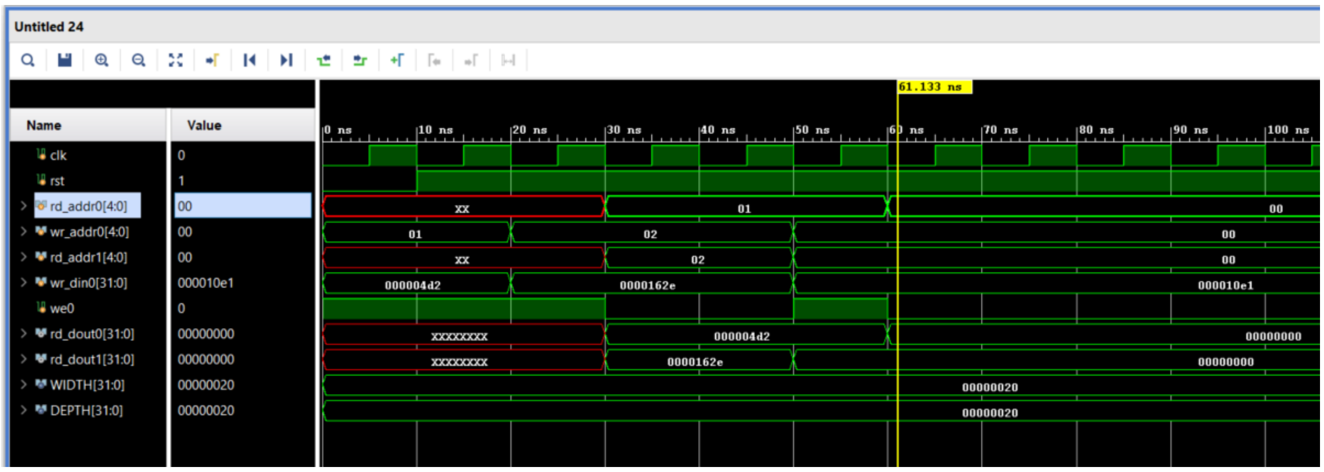


```
 89: MachineCode Hexadecimal:00000013 MachineCode Binary:00000000000000000000000000010011
 90: MachineCode Hexadecimal:00000013 MachineCode Binary:00000000000000000000000000010011
 91: MachineCode Hexadecimal:00000013 MachineCode Binary:00000000000000000000000000010011
 92: MachineCode Hexadecimal:00000013 MachineCode Binary:00000000000000000000000000010011
 93: MachineCode Hexadecimal:00000013 MachineCode Binary:00000000000000000000000000010011
 94: MachineCode Hexadecimal:00000013 MachineCode Binary:00000000000000000000000000010011
 95: MachineCode Hexadecimal:00000013 MachineCode Binary:00000000000000000000000000010011
 96: MachineCode Hexadecimal:00000013 MachineCode Binary:00000000000000000000000000010011
 97: MachineCode Hexadecimal:00000013 MachineCode Binary:00000000000000000000000000010011
 98: MachineCode Hexadecimal:00000013 MachineCode Binary:00000000000000000000000000010011
run 1000 ns
 99: MachineCode Hexadecimal:00000013 MachineCode Binary:00000000000000000000000000010011
100: MachineCode Hexadecimal:00000013 MachineCode Binary:00000000000000000000000000010011
101: MachineCode Hexadecimal:00000013 MachineCode Binary:00000000000000000000000000010011
102: MachineCode Hexadecimal:00000013 MachineCode Binary:00000000000000000000000000010011
103: MachineCode Hexadecimal:00000013 MachineCode Binary:00000000000000000000000000010011
104: MachineCode Hexadecimal:00000013 MachineCode Binary:00000000000000000000000000010011
105: MachineCode Hexadecimal:00000013 MachineCode Binary:00000000000000000000000000010011
106: MachineCode Hexadecimal:00000013 MachineCode Binary:00000000000000000000000000010011
107: MachineCode Hexadecimal:00000013 MachineCode Binary:00000000000000000000000000010011
108: MachineCode Hexadecimal:00000013 MachineCode Binary:00000000000000000000000000010011
109: MachineCode Hexadecimal:00000013 MachineCode Binary:00000000000000000000000000010011
110: MachineCode Hexadecimal:00000013 MachineCode Binary:00000000000000000000000000010011
111: MachineCode Hexadecimal:00000013 MachineCode Binary:00000000000000000000000000010011
112: MachineCode Hexadecimal:00000013 MachineCode Binary:00000000000000000000000000010011
113: MachineCode Hexadecimal:00000013 MachineCode Binary:00000000000000000000000000010011
114: MachineCode Hexadecimal:00000013 MachineCode Binary:00000000000000000000000000010011
115: MachineCode Hexadecimal:00000013 MachineCode Binary:00000000000000000000000000010011
116: MachineCode Hexadecimal:00000013 MachineCode Binary:00000000000000000000000000010011
117: MachineCode Hexadecimal:00000013 MachineCode Binary:00000000000000000000000000010011
118: MachineCode Hexadecimal:00000013 MachineCode Binary:00000000000000000000000000010011
119: MachineCode Hexadecimal:00000013 MachineCode Binary:00000000000000000000000000010011
120: MachineCode Hexadecimal:00000013 MachineCode Binary:00000000000000000000000000010011
121: MachineCode Hexadecimal:00000013 MachineCode Binary:00000000000000000000000000010011
122: MachineCode Hexadecimal:00000013 MachineCode Binary:00000000000000000000000000010011
123: MachineCode Hexadecimal:00000013 MachineCode Binary:00000000000000000000000000010011
124: MachineCode Hexadecimal:00000013 MachineCode Binary:00000000000000000000000000010011
125: MachineCode Hexadecimal:00000013 MachineCode Binary:00000000000000000000000000010011
126: MachineCode Hexadecimal:00000013 MachineCode Binary:00000000000000000000000000010011
127: MachineCode Hexadecimal:00000013 MachineCode Binary:00000000000000000000000000010011
Type a Tcl command here
```

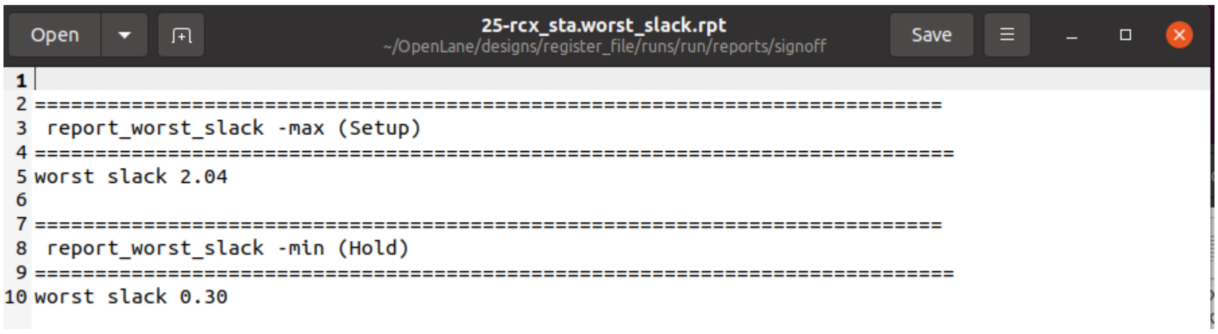
- **Data Memory**
  In this section, it is aimed to show 8bit, 16bit and 32bit values. An 8-bit (1 byte) value is displayed in 4 ranges, a 16-bit (half word) value in 2 ranges, and a 32-bit (word) value in 1 range. This part is coded in the upper module with the case structure. Then a testbench was designed for this module. Word, 1byte and halfword are specified in the simulation, respectively. The wr_strb signal represents the case states, there are 7 states in total. After the wr_din0 signal reaches 32bit and the we01 signal becomes 1, the output appears in the rd_dout0 signal.



- **Register File**
  Below is the simulation result of the register file module created using the memory_block.v top module. In this module, 2 inputs named rd_addr0 and rd_addr1 and two outputs named rd_dout0 and rd_dout1 have been added. The values given here appear in the relevant outputs, respectively. It is said that when rd_addr0 and rd_addr1 are given 0 as stated in the assignment, 0 should be output. It is stated that when both inputs return 0, 0 is read at the output. Incoming values other than 0 are shown to be output directly. The circuit is working correctly.



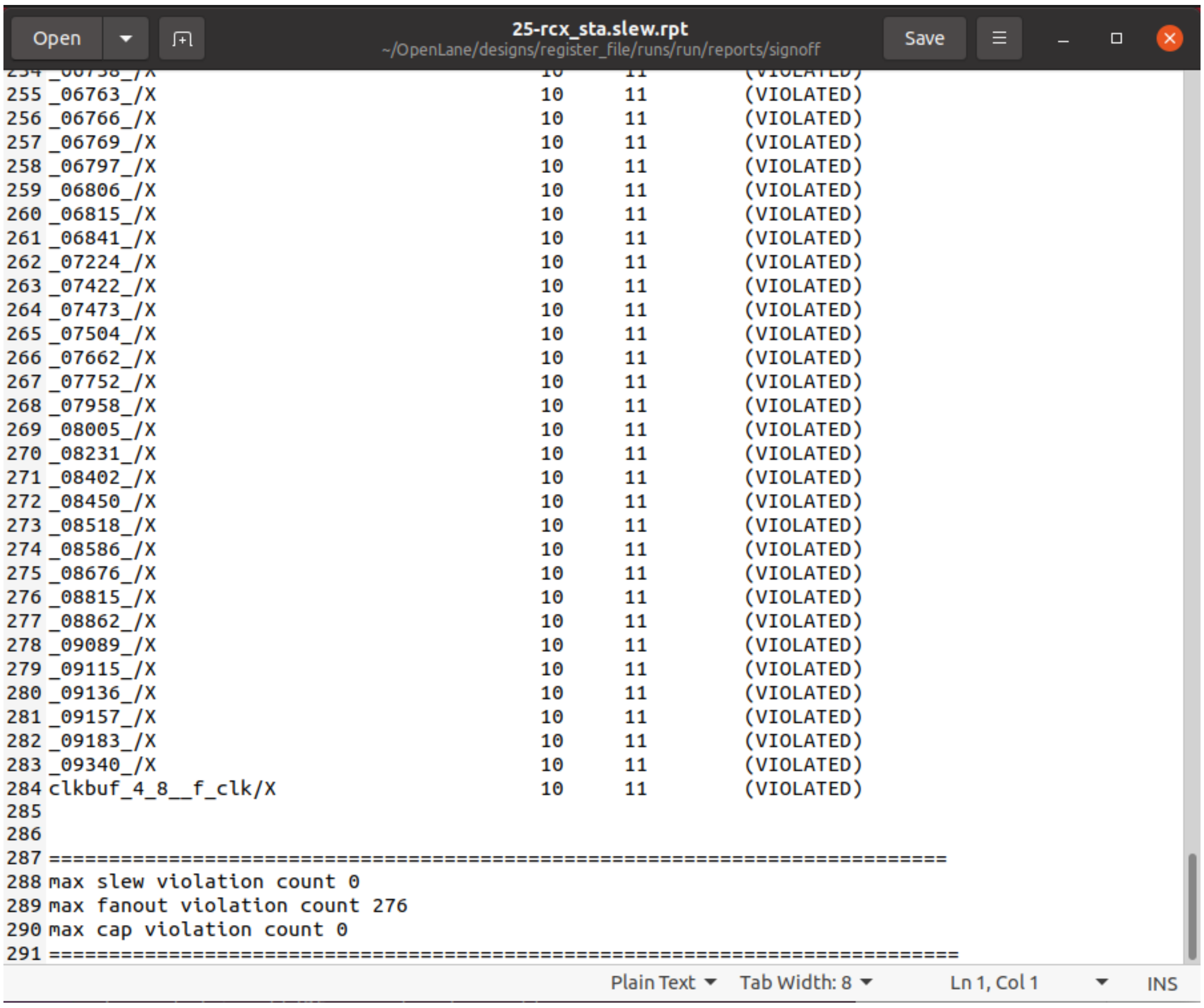
✓ Setup and Hold Violations


Open
25-rcx_sta.worst_slack.rpt
~/OpenLane/designs/register_file/runs/run/reports/signoff
Save

```

===========================================================================
 report_worst_slack -max (Setup)
============================================================================
worst slack 2.04

===========================================================================
 report_worst_slack -min (Hold)
============================================================================
worst slack 0.30
```

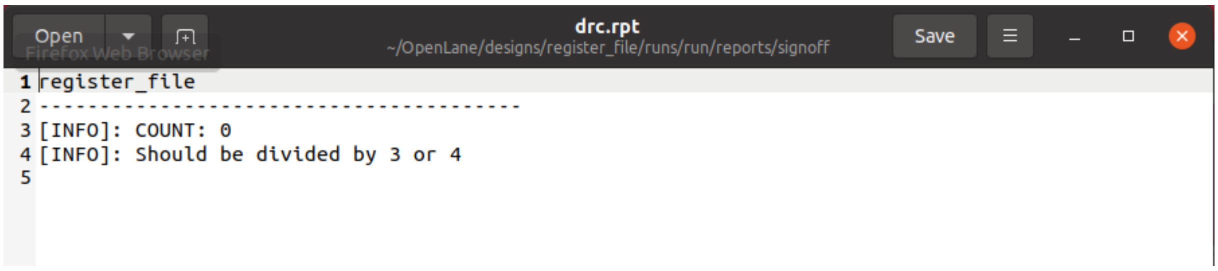
Maximum frequency 1/2.04*10^-9 = 490.196 Mhz

✓ DRV



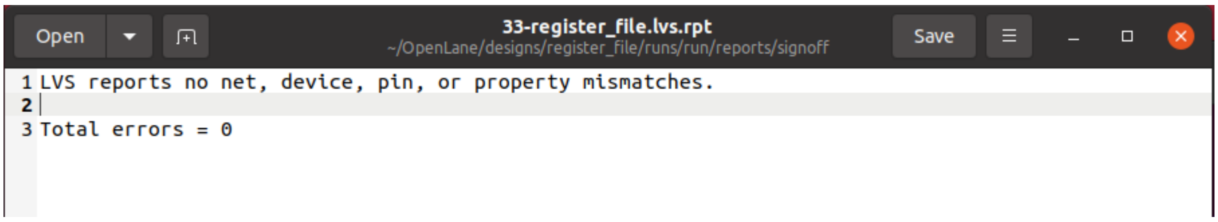
```
_06763_/X                                 10    11        (VIOLATED)
_06766_/X                                 10    11        (VIOLATED)
_06769_/X                                 10    11        (VIOLATED)
_06797_/X                                 10    11        (VIOLATED)
_06806_/X                                 10    11        (VIOLATED)
_06815_/X                                 10    11        (VIOLATED)
_06841_/X                                 10    11        (VIOLATED)
_07224_/X                                 10    11        (VIOLATED)
_07422_/X                                 10    11        (VIOLATED)
_07473_/X                                 10    11        (VIOLATED)
_07504_/X                                 10    11        (VIOLATED)
_07662_/X                                 10    11        (VIOLATED)
_07752_/X                                 10    11        (VIOLATED)
_07958_/X                                 10    11        (VIOLATED)
_08005_/X                                 10    11        (VIOLATED)
_08231_/X                                 10    11        (VIOLATED)
_08402_/X                                 10    11        (VIOLATED)
_08450_/X                                 10    11        (VIOLATED)
_08518_/X                                 10    11        (VIOLATED)
_08586_/X                                 10    11        (VIOLATED)
_08676_/X                                 10    11        (VIOLATED)
_08815_/X                                 10    11        (VIOLATED)
_08862_/X                                 10    11        (VIOLATED)
_09089_/X                                 10    11        (VIOLATED)
_09115_/X                                 10    11        (VIOLATED)
_09136_/X                                 10    11        (VIOLATED)
_09157_/X                                 10    11        (VIOLATED)
_09183_/X                                 10    11        (VIOLATED)
_09340_/X                                 10    11        (VIOLATED)
clkbuf_4_8__f_clk/X                       10    11        (VIOLATED)



===========================================================================
max slew violation count 0
max fanout violation count 276
max cap violation count 0
============================================================================
```

✓ DRC



```
register_file
----------------------------------------
[INFO]: COUNT: 0
[INFO]: Should be divided by 3 or 4

```

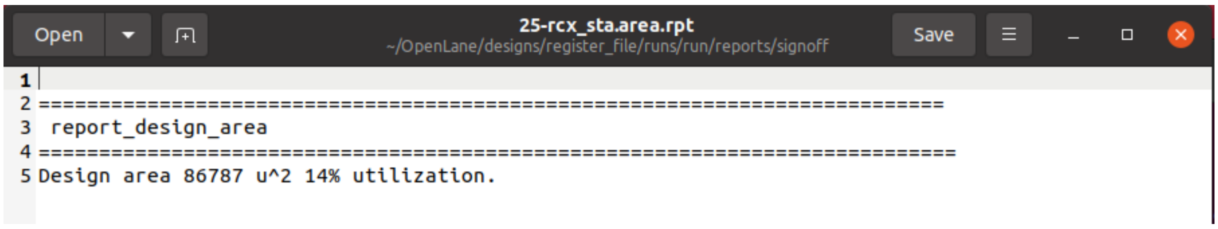
✓ LVS



```
LVS reports no net, device, pin, or property mismatches.

Total errors = 0
```

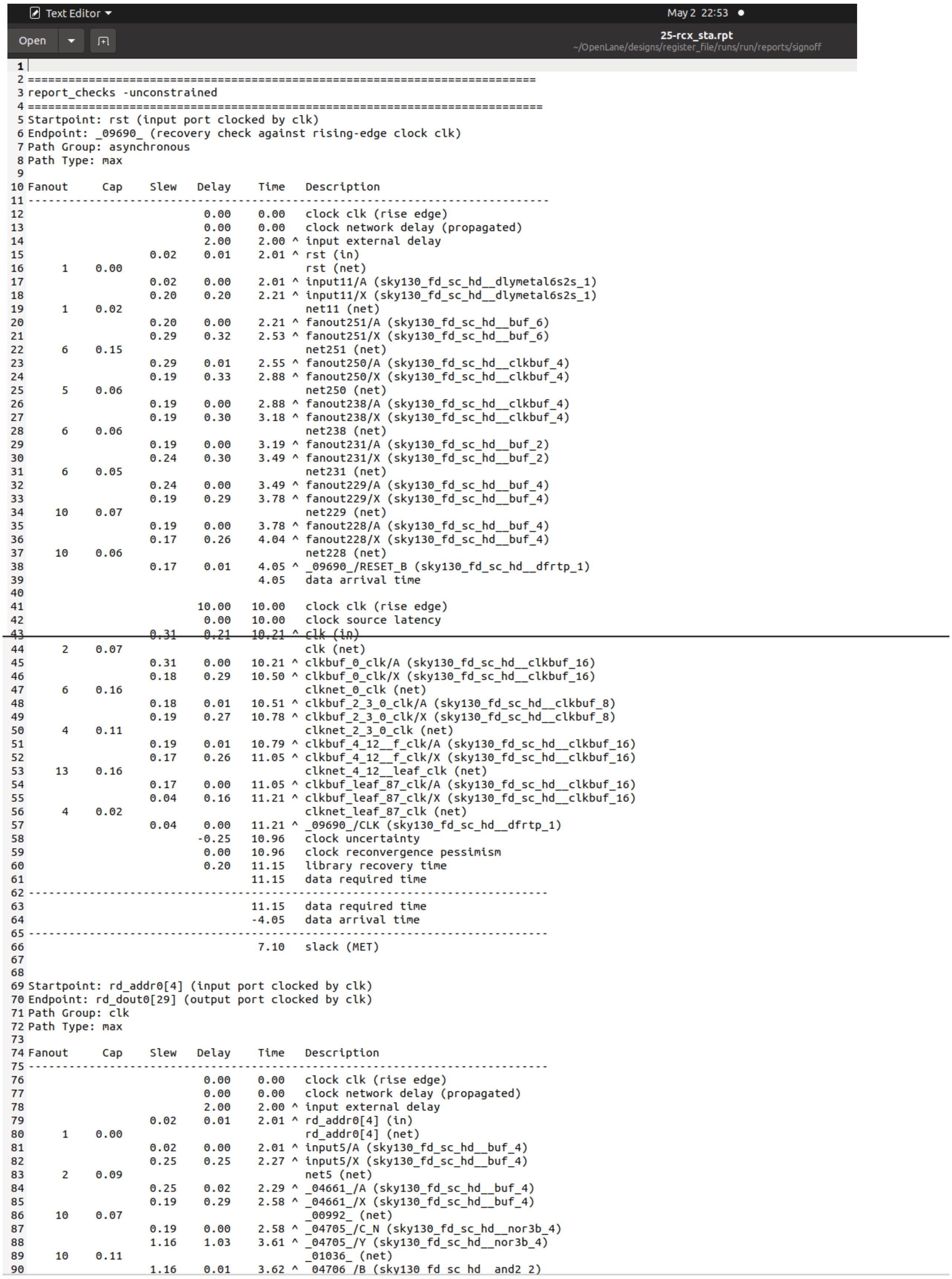
✓ Antennas outputs are in .zip file. It is not included here because it is too long.

✓ Final Area

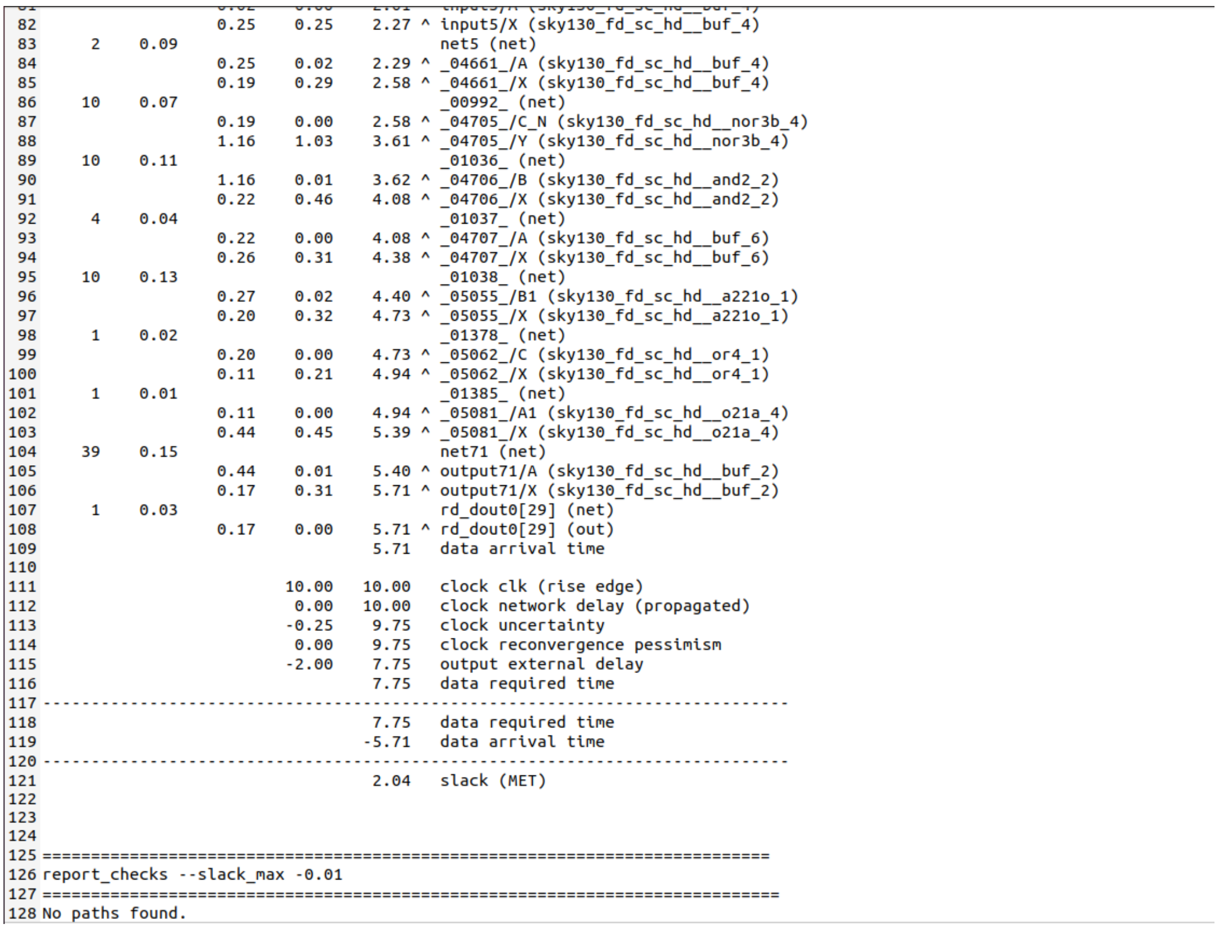


```

===========================================================================
 report_design_area
===========================================================================
Design area 86787 u^2 14% utilization.
```

✓ Path



```

===========================================================================
report_checks -unconstrained
============================================================================
Startpoint: rst (input port clocked by clk)
Endpoint: _09690_ (recovery check against rising-edge clock clk)
Path Group: asynchronous
Path Type: max

Fanout     Cap    Slew   Delay    Time   Description
-----------------------------------------------------------------------------
                          0.00    0.00   clock clk (rise edge)
                          0.00    0.00   clock network delay (propagated)
                          2.00    2.00 ^ input external delay
                  0.02    0.01    2.01 ^ rst (in)
     1    0.00                           rst (net)
                  0.02    0.00    2.01 ^ input11/A (sky130_fd_sc_hd__dlymetal6s2s_1)
                  0.20    0.20    2.21 ^ input11/X (sky130_fd_sc_hd__dlymetal6s2s_1)
     1    0.02                           net11 (net)
                  0.20    0.00    2.21 ^ fanout251/A (sky130_fd_sc_hd__buf_6)
                  0.29    0.32    2.53 ^ fanout251/X (sky130_fd_sc_hd__buf_6)
     6    0.15                           net251 (net)
                  0.29    0.01    2.55 ^ fanout250/A (sky130_fd_sc_hd__clkbuf_4)
                  0.19    0.33    2.88 ^ fanout250/X (sky130_fd_sc_hd__clkbuf_4)
     5    0.06                           net250 (net)
                  0.19    0.00    2.88 ^ fanout238/A (sky130_fd_sc_hd__clkbuf_4)
                  0.19    0.30    3.18 ^ fanout238/X (sky130_fd_sc_hd__clkbuf_4)
     6    0.06                           net238 (net)
                  0.19    0.00    3.19 ^ fanout231/A (sky130_fd_sc_hd__buf_2)
                  0.24    0.30    3.49 ^ fanout231/X (sky130_fd_sc_hd__buf_2)
     6    0.05                           net231 (net)
                  0.24    0.00    3.49 ^ fanout229/A (sky130_fd_sc_hd__buf_4)
                  0.19    0.29    3.78 ^ fanout229/X (sky130_fd_sc_hd__buf_4)
    10    0.07                           net229 (net)
                  0.19    0.00    3.78 ^ fanout228/A (sky130_fd_sc_hd__buf_4)
                  0.17    0.26    4.04 ^ fanout228/X (sky130_fd_sc_hd__buf_4)
    10    0.06                           net228 (net)
                  0.17    0.01    4.05 ^ _09690_/RESET_B (sky130_fd_sc_hd__dfrtp_1)
                                  4.05   data arrival time

                         10.00   10.00   clock clk (rise edge)
                          0.00   10.00   clock source latency
                  0.31    0.21   10.21 ^ clk (in)
     2    0.07                           clk (net)
                  0.31    0.00   10.21 ^ clkbuf_0_clk/A (sky130_fd_sc_hd__clkbuf_16)
                  0.18    0.29   10.50 ^ clkbuf_0_clk/X (sky130_fd_sc_hd__clkbuf_16)
     6    0.16                           clknet_0_clk (net)
                  0.18    0.01   10.51 ^ clkbuf_2_3_0_clk/A (sky130_fd_sc_hd__clkbuf_8)
                  0.19    0.27   10.78 ^ clkbuf_2_3_0_clk/X (sky130_fd_sc_hd__clkbuf_8)
     4    0.11                           clknet_2_3_0_clk (net)
                  0.19    0.01   10.79 ^ clkbuf_4_12__f_clk/A (sky130_fd_sc_hd__clkbuf_16)
                  0.17    0.26   11.05 ^ clkbuf_4_12__f_clk/X (sky130_fd_sc_hd__clkbuf_16)
    13    0.16                           clknet_4_12__leaf_clk (net)
                  0.17    0.00   11.05 ^ clkbuf_leaf_87_clk/A (sky130_fd_sc_hd__clkbuf_16)
                  0.04    0.16   11.21 ^ clkbuf_leaf_87_clk/X (sky130_fd_sc_hd__clkbuf_16)
     4    0.02                           clknet_leaf_87_clk (net)
                  0.04    0.00   11.21 ^ _09690_/CLK (sky130_fd_sc_hd__dfrtp_1)
                         -0.25   10.96   clock uncertainty
                          0.00   10.96   clock reconvergence pessimism
                          0.20   11.15   library recovery time
                                 11.15   data required time
-----------------------------------------------------------------------------
                                 11.15   data required time
                                 -4.05   data arrival time
-----------------------------------------------------------------------------
                                  7.10   slack (MET)


Startpoint: rd_addr0[4] (input port clocked by clk)
Endpoint: rd_dout0[29] (output port clocked by clk)
Path Group: clk
Path Type: max

Fanout     Cap    Slew   Delay    Time   Description
-----------------------------------------------------------------------------
                          0.00    0.00   clock clk (rise edge)
                          0.00    0.00   clock network delay (propagated)
                          2.00    2.00 ^ input external delay
                  0.02    0.01    2.01 ^ rd_addr0[4] (in)
     1    0.00                           rd_addr0[4] (net)
                  0.02    0.00    2.01 ^ input5/A (sky130_fd_sc_hd__buf_4)
                  0.25    0.25    2.27 ^ input5/X (sky130_fd_sc_hd__buf_4)
     2    0.09                           net5 (net)
                  0.25    0.02    2.29 ^ _04661_/A (sky130_fd_sc_hd__buf_4)
                  0.19    0.29    2.58 ^ _04661_/X (sky130_fd_sc_hd__buf_4)
    10    0.07                           _00992_ (net)
                  0.19    0.00    2.58 ^ _04705_/C_N (sky130_fd_sc_hd__nor3b_4)
                  1.16    1.03    3.61 ^ _04705_/Y (sky130_fd_sc_hd__nor3b_4)
    10    0.11                           _01036_ (net)
                  1.16    0.01    3.62 ^ _04706_/B (sky130_fd_sc_hd__and2_2)
```



```
                  0.25    0.25    2.27 ^ input5/X (sky130_fd_sc_hd__buf_4)
     2    0.09                           net5 (net)
                  0.25    0.02    2.29 ^ _04661_/A (sky130_fd_sc_hd__buf_4)
                  0.19    0.29    2.58 ^ _04661_/X (sky130_fd_sc_hd__buf_4)
    10    0.07                           _00992_ (net)
                  0.19    0.00    2.58 ^ _04705_/C_N (sky130_fd_sc_hd__nor3b_4)
                  1.16    1.03    3.61 ^ _04705_/Y (sky130_fd_sc_hd__nor3b_4)
    10    0.11                           _01036_ (net)
                  1.16    0.01    3.62 ^ _04706_/B (sky130_fd_sc_hd__and2_2)
                  0.22    0.46    4.08 ^ _04706_/X (sky130_fd_sc_hd__and2_2)
     4    0.04                           _01037_ (net)
                  0.22    0.00    4.08 ^ _04707_/A (sky130_fd_sc_hd__buf_6)
                  0.26    0.31    4.38 ^ _04707_/X (sky130_fd_sc_hd__buf_6)
    10    0.13                           _01038_ (net)
                  0.27    0.02    4.40 ^ _05055_/B1 (sky130_fd_sc_hd__a221o_1)
                  0.20    0.32    4.73 ^ _05055_/X (sky130_fd_sc_hd__a221o_1)
     1    0.02                           _01378_ (net)
                  0.20    0.00    4.73 ^ _05062_/C (sky130_fd_sc_hd__or4_1)
                  0.11    0.21    4.94 ^ _05062_/X (sky130_fd_sc_hd__or4_1)
     1    0.01                           _01385_ (net)
                  0.11    0.00    4.94 ^ _05081_/A1 (sky130_fd_sc_hd__o21a_4)
                  0.44    0.45    5.39 ^ _05081_/X (sky130_fd_sc_hd__o21a_4)
    39    0.15                           net71 (net)
                  0.44    0.01    5.40 ^ output71/A (sky130_fd_sc_hd__buf_2)
                  0.17    0.31    5.71 ^ output71/X (sky130_fd_sc_hd__buf_2)
     1    0.03                           rd_dout0[29] (net)
                  0.17    0.00    5.71 ^ rd_dout0[29] (out)
                                  5.71   data arrival time

                         10.00   10.00   clock clk (rise edge)
                          0.00   10.00   clock network delay (propagated)
                         -0.25    9.75   clock uncertainty
                          0.00    9.75   clock reconvergence pessimism
                         -2.00    7.75   output external delay
                                  7.75   data required time
-----------------------------------------------------------------------------
                                  7.75   data required time
                                 -5.71   data arrival time
-----------------------------------------------------------------------------
                                  2.04   slack (MET)



===========================================================================
report_checks --slack_max -0.01
============================================================================
No paths found.
```

For asynchronous group

Startpoint: rst (input port clocked by clk)

Endponit: _09690_ (recovery check against rising-edge clock clk)

Worst path is 7.1 ns (slack (MET)).

For clk group

Startpoint: rd_addr0[4] (input port clocked by clk)

Endpoint: rd_dout[29] (output port clocked by clk)

Worst path is 2.04 ns (slack (MET)).

The number whose maximum gives the value of the clock frequency is the number in the critical path. This is how the critical path is found. Based on this, the critical path can be found. The above results are written according to these inferences.

✓ Layout