Lab5 Report

Liron Cohen 207481268

Yuval Mor 209011543

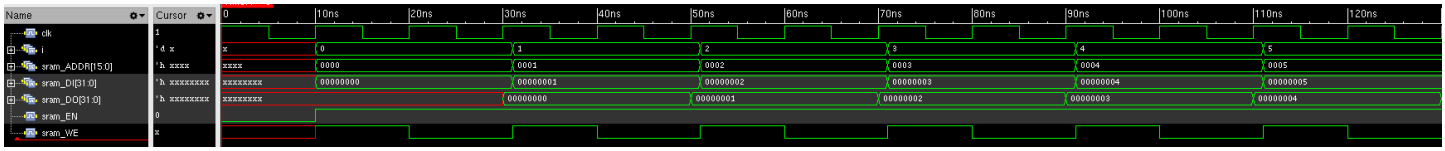
Question 1 - SRAM verification

We filled the missing code in the sram.v module and the sram_tb.v module.

We ran the xrun command the simulation finished successfully:

```
SRAM test finished successfully
Simulation complete via $finish(1) at time 1310731 NS + 0
./sram_tb.v:51  $finish;
xcelium> exit
TOOL:   xrun(64)        21.09-s006: Exiting on Dec 31, 2022 at 11:12:57 IST  (to
tal: 00:00:04)
[lironcohen3@micron-x01 lab5]$
```

The waveform is:



As we can see for example, on $t = 30ns$ the clock rises, and the integer 1 is written to address 0001 (shows in $SRAM\_DI$) and on $t = 50ns$ the clock rises and the output data that is read from address 0001 is the value 1 (shows in $SRAM\_DO$), so the read after write is working and the simulation succeeded.

Question 2 - processor implementation (no DMA)

We filled the missing code in the alu.v module, sp.v module and ctl.v module.

The required files are attached.

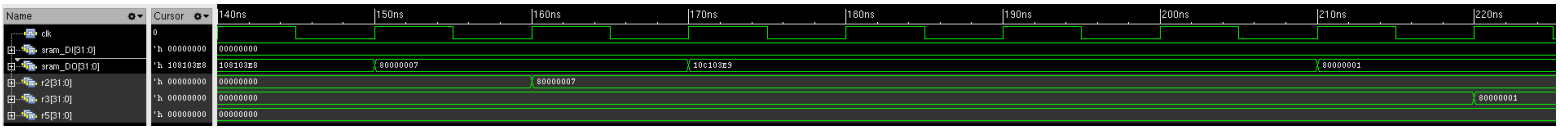
Question 3 - verification #1: example.bin

We ran the simulation of the provided example.bin and made sure the traces are matched.

Question 4 - verification #2: add

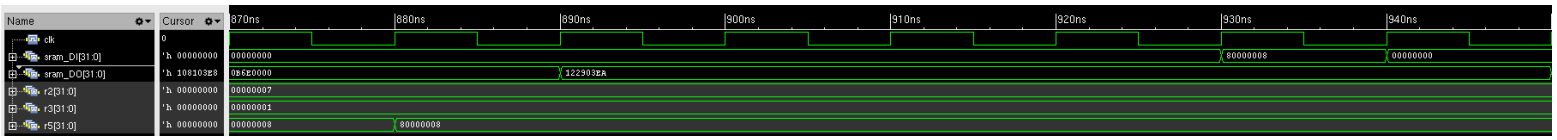
We ran the simulation of our add.bin program and made sure it's trace matched the lab 2 cycle trace.

The relevant waveforms:



We can see that in $t = 150ns$, the value of $SRAM\_DO$ is the first number (80000007) and it is read into $r2$ in $t = 160ns$.

We can also see that in $t = 210ns$, the value of $SRAM\_DI$ is the second number (80000001) and it is read into $r3$ in $t = 220ns$.

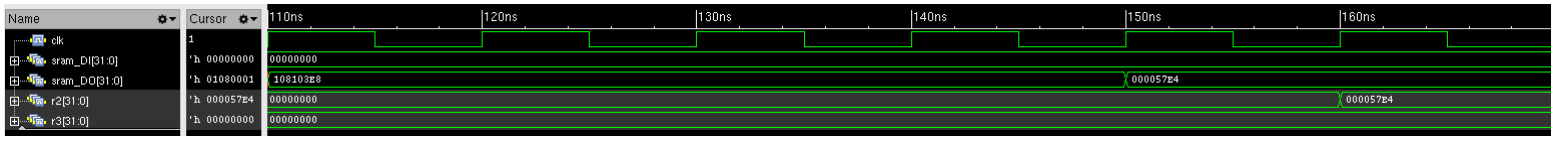


We can see that in $t = 880ns$, the value of $r5$ is the result (80000008) and it is written to $SRAM\_DO$ in $t = 930ns$.

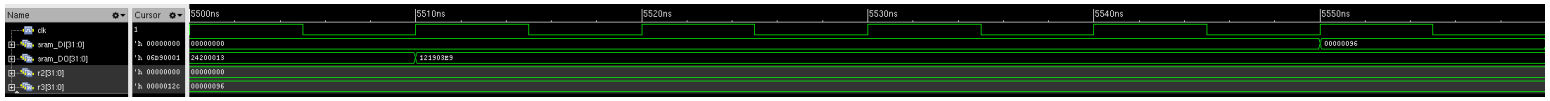
Question 5 - verification #3: sqrtq

We ran the simulation of our sqrtq.bin program and made sure it's trace matched the lab 2 cycle trace.

The relevant waveforms:



We can see that in $t = 150ns$, the value of $SRAM\_DO$ is the input to the program ($0x57E4$) and it is read into $r2$ in $t = 160ns$.



We can see that in $t = 5500ns$, the value of $r3$ is the result ($0x96$) and it is written to $SRAM\_DI$ in $t = 5550ns$.

Question 6 - DMA implementation

We added the DMA state machine that works in parallel to the main program while using the following added define statements:

`define DMA_STATE_IDLE 0

`define DMA_STATE_FETCH0 1

`define DMA_STATE_FETCH1 2

`define DMA_STATE_DEC0 3

`define DMA_STATE_DEC1 4

`define DMA_STATE_EXEC0 5

`define DMA_STATE_EXEC1 6

Question 7 - DMA verification

We verified our DMA design by running the DMA test program from the previous lab that contains memory access of both the assembly program and the DMA machine.

We got the following waveforms: