ELEC 374 Project Phase 1 Report

Group 42

- Alex O'Neill, 20043101, 16amon
- Jared McGrath, 20053313, 16jtm2

1. Design Overview

Our design was made entirely in Verilog, using no arithmetic operators (`+`, `-`, `/` or `*`), and also implementing some logical operators (left and right shifts and rotates) entirely from scratch. We also implemented various techniques for faster addition, including a Carry Lookahead Adder and Carry Save Adder, both of which are utilized in the Multiplier.

The structure of our design is based on the 3-bus architecture referenced in the lab reader. This allowed us to remove the now redundant `A`, `B`, `Y`, and `Z` registers, and greatly simplify interconnections between components of our datapath.

All of the modules we wrote have testbench modules included in the same module - for example, the `datapath` module has a `datapath_test` module both declared in the `datapath.v` file. We used a combination of a Makefile, and the ModelSim command line interface in order to run automatic tests. We use `$display()` calls to observe expected and actual outputs, and then report any differences by simulating the designs.

All our code is included in the attached `.zip` file. The module structure of the design is as follows:

- `cpu` : The top level module (empty, except for creating a `datapath` in Phase 1)
  - `datapath` : Contains all Phase 1 logic
    - `alu` : The ALU, containing all ALU operations required for Phase 1 in various sub-modules.
    - `register_file` : The general purpose register file for registers `r0` - `r15`
    - `register` : A simple register used for `PC`, `IR`, `MD`, `MA`, `HI` and `LO` registers.

2. Testbench Waveforms

We tested all the required datapath instructions in a single test module (the `datapath_test` one), via simulating them sequentially as they pass through the datapath.

Testing ADD Instruction

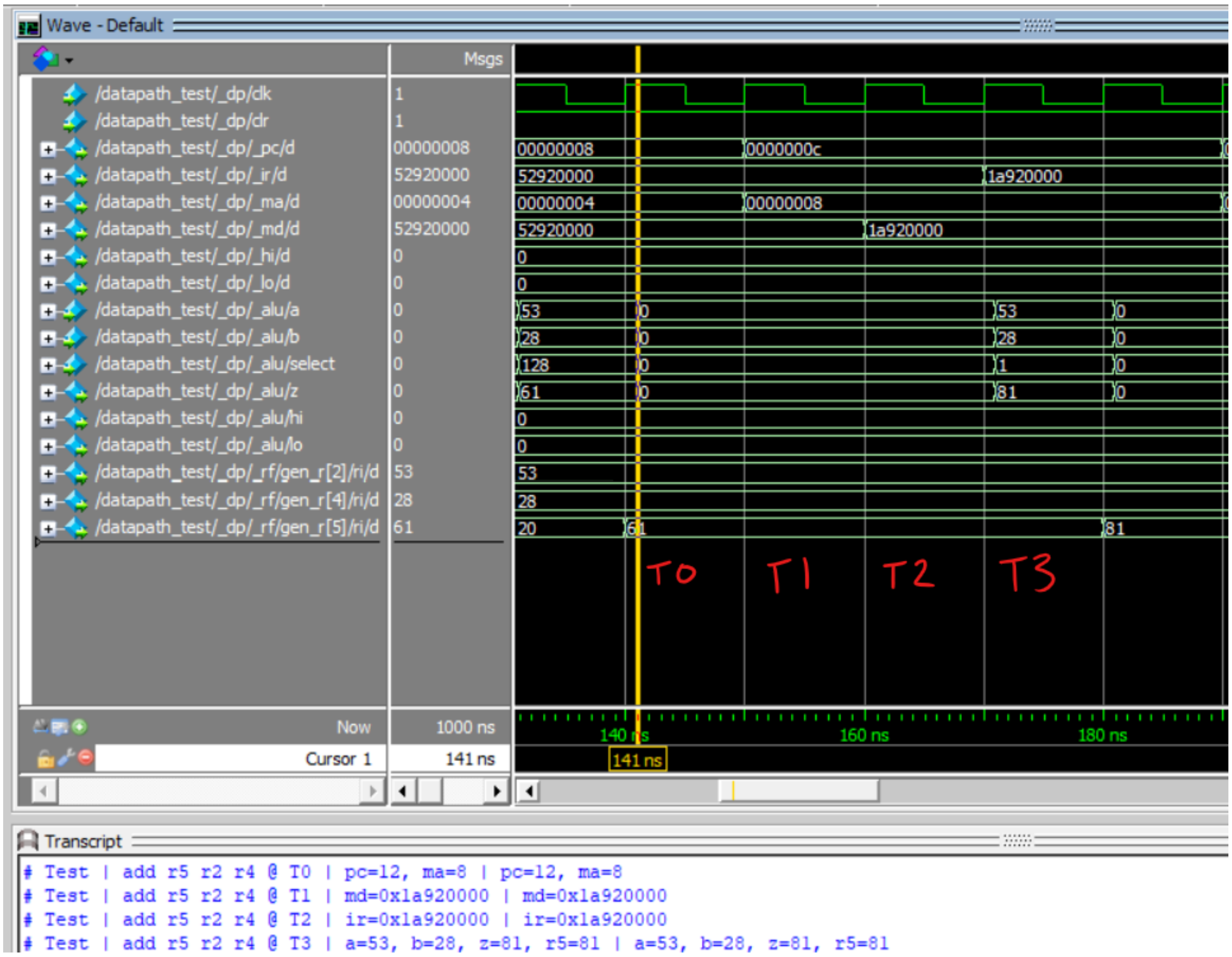
The `add r5, r2, r4` instruction has the following RTN, under our datapath:

- T0: `PC <- PC + 4`, `MA <- PC`
- T1: `MD <- Memory[MA]`
- T2: `IR <- MD`
- T3: `rX <- rY + rZ`

This translates to the following sequence of control signals:

- T0: `pc_increment <= 1'b1; ma_in_pc <= 1'b1;` (Increment `PC`, and load `MA` from `PC`)
- T1: `md_in_memory <= 1'b1;` (Load `MD` from memory)
- T2: `ir_en <= 1'b1;` (Load `IR`)
- T3:
  - `rf_z_addr <= 4'b0101;` Write register rZ = r5
  - `rf_a_addr <= 4'b0010;` Read register rX = r2
  - `rf_b_addr <= 4'b0100;` Read register rY = r4
  - `alu_a_in_rf <= 1'b1;` Select the ALU `A` input to be from rX
  - `alu_b_in_rf <= 1'b1;` Select the ALU `B` input to be from rY
  - `rf_in_alu <= 1'b1;` Write Enable for rZ
  - `alu_add <= 1'b1;` Set the ALU `ADD` flag (visible on the waveform as `alu_select = ????`)

The waveform of this instruction, highlighting all the relavant registers as the data passes through them is pictured below:



```
/datapath_test/_dp/clk                1
/datapath_test/_dp/clr                1
/datapath_test/_dp/_pc/d              00000008
/datapath_test/_dp/_ir/d              52920000
/datapath_test/_dp/_ma/d              00000004
/datapath_test/_dp/_md/d              52920000
/datapath_test/_dp/_hi/d              0
/datapath_test/_dp/_lo/d              0
/datapath_test/_dp/_alu/a             0
/datapath_test/_dp/_alu/b             0
/datapath_test/_dp/_alu/select        0
/datapath_test/_dp/_alu/z             0
/datapath_test/_dp/_alu/hi            0
/datapath_test/_dp/_alu/lo            0
/datapath_test/_dp/_rf/gen_r[2]/ri/d  53
/datapath_test/_dp/_rf/gen_r[4]/ri/d  28
/datapath_test/_dp/_rf/gen_r[5]/ri/d  61
```

```
# Test | add r5 r2 r4 @ T0 | pc=12, ma=8 | pc=12, ma=8
# Test | add r5 r2 r4 @ T1 | md=0x1a920000 | md=0x1a920000
# Test | add r5 r2 r4 @ T2 | ir=0x1a920000 | ir=0x1a920000
# Test | add r5 r2 r4 @ T3 | a=53, b=28, z=81, r5=81 | a=53, b=28, z=81, r5=81
```

The RTN, control signals, and waveform for `add`, `sub`, `and`, `or`, `shl`, `shr`, `rol`, `ror` are all identical, except for the choice of control signal, and so we omitted them. However, the testbench code for each instruction can be found in the `datapath_test` module.

Testing MUL Instruction

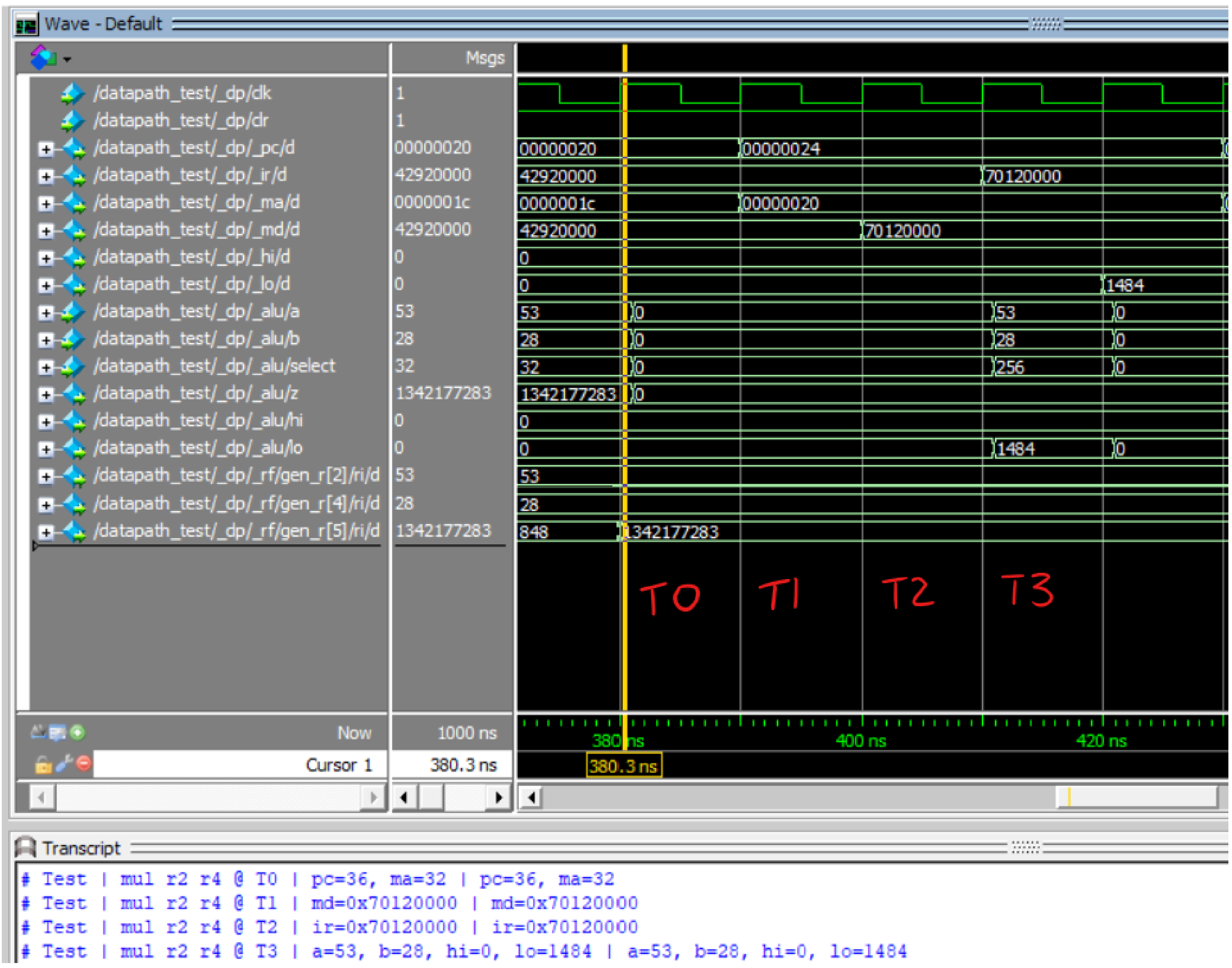
The `mul r2, r4` instruction has the following RTN, under our datapath:

- T0: `PC <- PC + 4`, `MA <- PC`
- T1: `MD <- Memory[MA]`
- T2: `IR <- MD`
- T3: `HI, LO <- rY * rZ`

This translates to the following sequence of control signals:

- T0: `pc_increment <= 1'b1; ma_in_pc <= 1'b1;` (Increment `PC`, and load `MA` from `PC`)
- T1: `md_in_memory <= 1'b1;` (Load `MD` from memory)
- T2: `ir_en <= 1'b1;` (Load `IR`)
- T3:
  - `rf_z_addr <= 4'b0101;` Write register rZ = r5
  - `rf_a_addr <= 4'b0010;` Read register rX = r2
  - `rf_b_addr <= 4'b0100;` Read register rY = r4
  - `alu_a_in_rf <= 1'b1;` Select the ALU `A` input to be from rX
  - `alu_b_in_rf <= 1'b1;` Select the ALU `B` input to be from rY
  - `lo_en <= 1'b1;` Write Enable for `LO`
  - `hi_en <= 1'b1;` Write Enable for `HI`
  - `alu_mul <= 1'b1;` Set the ALU `ADD` flag (visible on the waveform as `alu_select = 256 = 0b000100000000`)

The waveform of this instruction, highlighting all the relavant registers as the data passes through them is pictured below:



```
/datapath_test/_dp/clk                  1
/datapath_test/_dp/clr                  1
/datapath_test/_dp/_pc/d                00000020
/datapath_test/_dp/_ir/d                42920000
/datapath_test/_dp/_ma/d                0000001c
/datapath_test/_dp/_md/d                42920000
/datapath_test/_dp/_hi/d                0
/datapath_test/_dp/_lo/d                0
/datapath_test/_dp/_alu/a               53
/datapath_test/_dp/_alu/b               28
/datapath_test/_dp/_alu/select          32
/datapath_test/_dp/_alu/z               1342177283
/datapath_test/_dp/_alu/hi              0
/datapath_test/_dp/_alu/lo              0
/datapath_test/_dp/_rf/gen_r[2]/ri/d    53
/datapath_test/_dp/_rf/gen_r[4]/ri/d    28
/datapath_test/_dp/_rf/gen_r[5]/ri/d    1342177283
```

```
# Test | mul r2 r4 @ T0 | pc=36, ma=32 | pc=36, ma=32
# Test | mul r2 r4 @ T1 | md=0x70120000 | md=0x70120000
# Test | mul r2 r4 @ T2 | ir=0x70120000 | ir=0x70120000
# Test | mul r2 r4 @ T3 | a=53, b=28, hi=0, lo=1484 | a=53, b=28, hi=0, lo=1484
```

The RTN, control signals, and waveform for `mul` and `div` are all identical, except for the choice of control signal, and so we omitted the `div` instruction.

Testing NEG Instruction

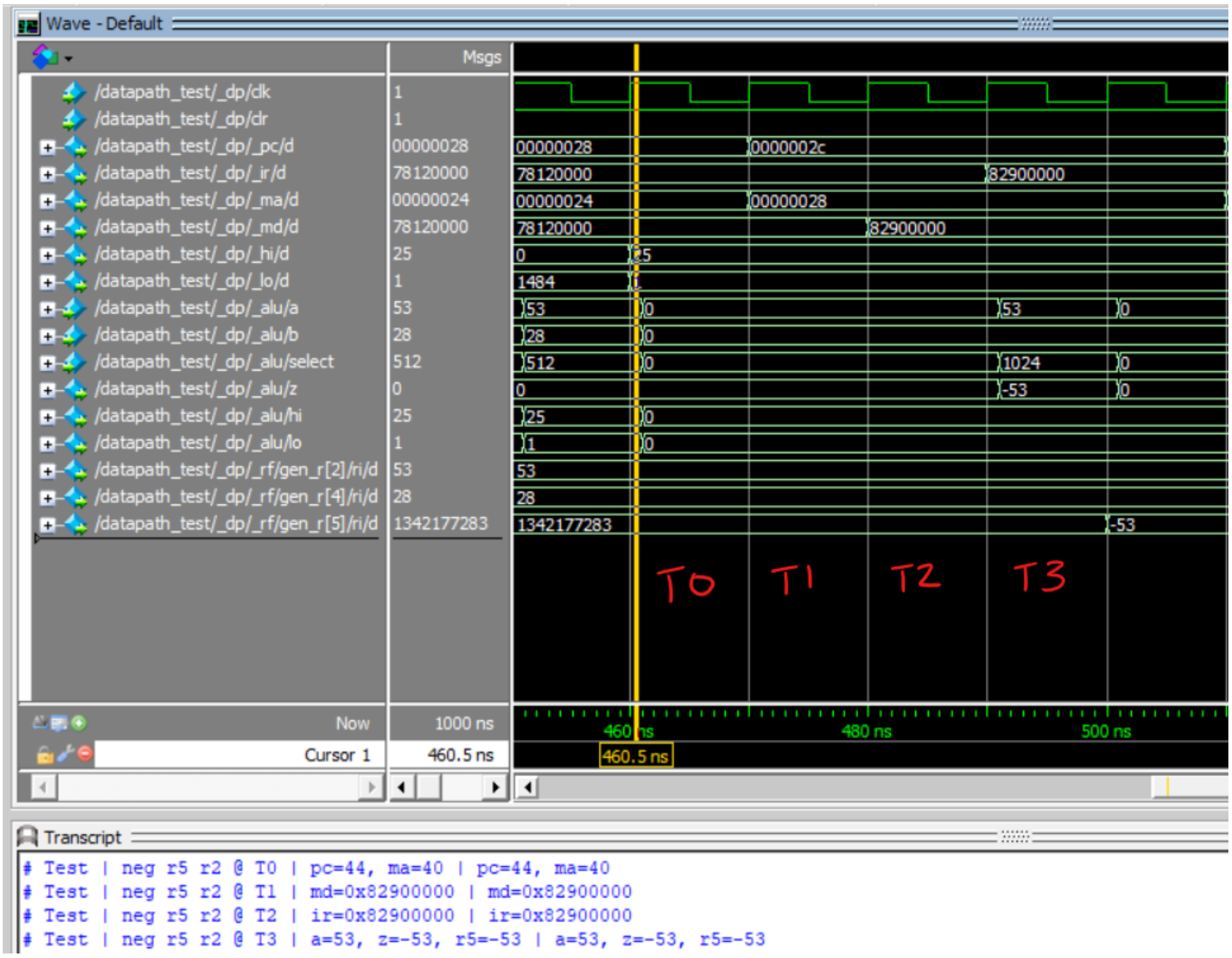
The `neg r5, r2` instruction has the following RTN, under our datapath:

- T0: `PC <- PC + 4`, `MA <- PC`
- T1: `MD <- Memory[MA]`
- T2: `IR <- MD`
- T3: `rX <- -(rY)`

This translates to the following sequence of control signals:

- T0: `pc_increment <= 1'b1; ma_in_pc <= 1'b1;` (Increment `PC`, and load `MA` from `PC`)
- T1: `md_in_memory <= 1'b1;` (Load `MD` from memory)
- T2: `ir_en <= 1'b1;` (Load `IR`)
- T3:
  - `rf_z_addr <= 4'b0101;` Write register rZ = r5
  - `rf_a_addr <= 4'b0010;` Read register rX = r2
  - `alu_a_in_rf <= 1'b1;` Select the ALU `A` input to be from rX
  - `rf_in_alu <= 1'b1;` Write Enable for rZ
  - `alu_neg <= 1'b1;` Set the ALU `NEG` flag (visible on the waveform as `alu_select = ????`)

The waveform of this instruction, highlighting all the relavant registers as the data passes through them is pictured below:


Wave - Default
Msgs
/datapath_test/_dp/clk 1
/datapath_test/_dp/clr 1
/datapath_test/_dp/_pc/d 00000028
/datapath_test/_dp/_ir/d 78120000
/datapath_test/_dp/_ma/d 00000024
/datapath_test/_dp/_md/d 78120000
/datapath_test/_dp/_hi/d 25
/datapath_test/_dp/_lo/d 1
/datapath_test/_dp/_alu/a 53
/datapath_test/_dp/_alu/b 28
/datapath_test/_dp/_alu/select 512
/datapath_test/_dp/_alu/z 0
/datapath_test/_dp/_alu/hi 25
/datapath_test/_dp/_alu/lo 1
/datapath_test/_dp/_rf/gen_r[2]/ri/d 53
/datapath_test/_dp/_rf/gen_r[4]/ri/d 28
/datapath_test/_dp/_rf/gen_r[5]/ri/d 1342177283
00000028
0000002c
78120000
82900000
00000024
00000028
78120000
82900000
0
25
1484
1
53
0
53
0
28
0
512
0
1024
0
0
-53
0
25
0
1
0
53
28
1342177283
-53
T0
T1
T2
T3
Now 1000 ns
Cursor 1 460.5 ns
460 ns
480 ns
500 ns
460.5 ns
Transcript

```
# Test | neg r5 r2 @ T0 | pc=44, ma=40 | pc=44, ma=40
# Test | neg r5 r2 @ T1 | md=0x82900000 | md=0x82900000
# Test | neg r5 r2 @ T2 | ir=0x82900000 | ir=0x82900000
# Test | neg r5 r2 @ T3 | a=53, z=-53, r5=-53 | a=53, z=-53, r5=-53
```

The RTN, control signals, and waveform for `neg` and `not` are identical, except for the choice of control signal, and so we omitted the `not` instruction.