COMPSYS701 GP1 Report

Group 8

Darren Ho (dho524), Thomas Yeh (cyeh243), Kalvin Huynh (khuy281)

Introduction

The ReCOP processor project aims to develop a reactive processor for control-dominated parts in time-critical applications. The processor will be implemented on a DE1-1 SOC using a Cyclone V chip. Our group will focus on building the datapath, control unit, memory interfaces, assembler, and external interfaces for the processor. We will develop an assembly compiler to convert the ReCOP instructions in assembly to MIF format. We tested our design in ModelSim, including the assembly compiler, and tested it on the DE1-SOC using the switches and 7-segment displays to output the values.

Customisations made to the ReCOP Processor

We have decided to exclude some instructions from the ReCOP instruction set to simplify the design and reduce complexity, as they are not a definite requirement for this project. The instructions below were not added to our ReCOP processor, but the option of adding it in the future is open, if required.

- CER
- CEOT
- SEOT
- LER
- SSVOP
- MAX

Control Unit

The control unit operates as an FSM, shown in Figure 1, with states T0, T1, T2, T3. T0 represents the setup state, to reset the mux signals and flags, which then asks the datapath to start the instruction fetch. T1 is the fetching state, where it waits for the instruction to be completely fetched (need more cycles for larger instructions i.e. immediate and direct) and once finished it will then set up the program counter mode based on the instruction fetched. T2 is the decode state where the variables and mux's are set and pointed to their locations and sent to the datapath. Finally, T3 is the execute state where a flag is set to perform the instruction. This modular approach simplifies control flow and allows for easy integration of new instructions.

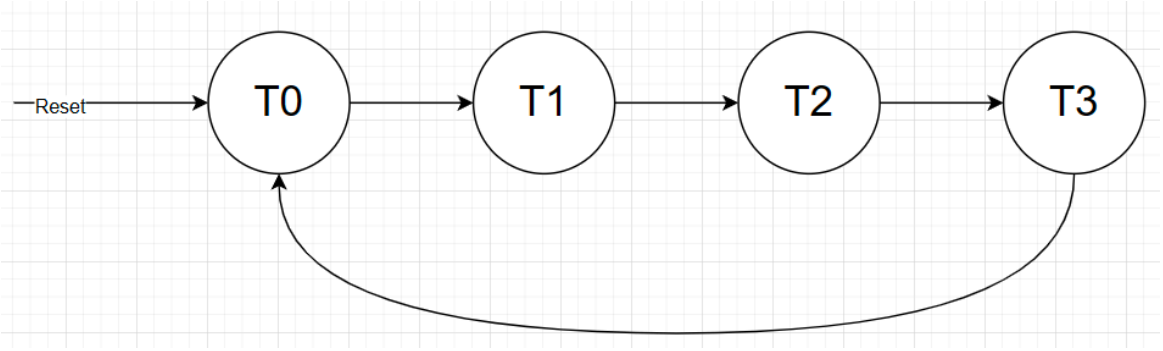


Figure 1. FSM States

Datapath

The figure below shows the datapath implementation of our ReCOP processor.

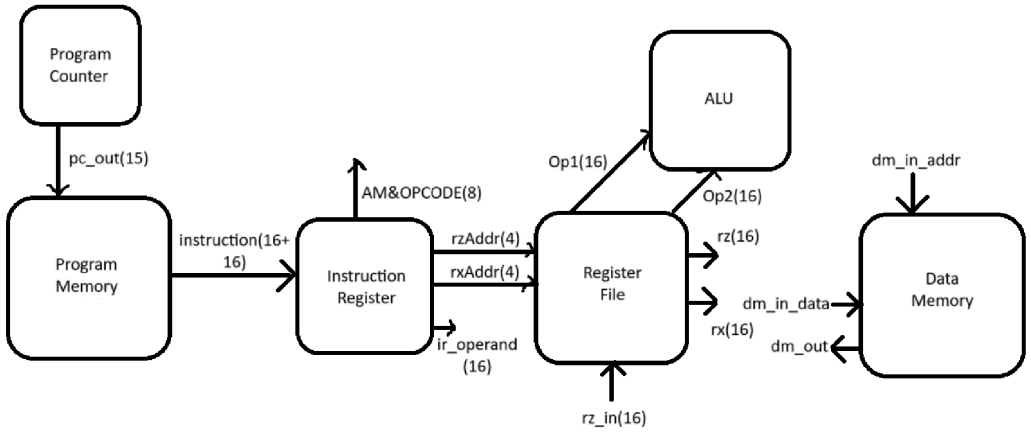


Figure 2. Datapath implementation

Assembly Compiler

```
    LDR R1 #1
    LDR R2 #2
    LDR R3 #0
    ADD R3 R3 R2
    STR R1 R2
    STR R2 R3
    LDR R4 R1
    LDR R5 R2
ENDPROG
END
```

Figure 3. Assembly Code

```
WIDTH = 16;
DEPTH = 4096;

ADDRESS_RADIX = HEX;
DATA_RADIX = BIN;

CONTENT
    BEGIN
    [00..FFF]: 1111111111111111;
    0   :0100000000010000;
    1   :0000000000000001;
    2   :0100000000100000;
    3   :0000000000000010;
    4   :0100000000110000;
    5   :0000000000000000;
    6   :1111100000110010;
    7   :1100001000010010;
    8   :1100001000100011;
    9   :1100000001000001;
    A   :1100000001010010;
    END;
```

Figure 4. Mif File

A custom assembly compiler was written in order to run specific instructions from ReCOP. Instead of using Cygwin terminal to run mrasm.exe, the new compiler is written as a Python script, and when it is run, it will convert the assembly instructions into a .mif format, which can be read by the program memory in the ReCOP processor to execute. The figures above show the conversion between the assembly and mif.

Testing on FPGA and Modelsim

These are the resource usage when compiled on Quartus. Images can be seen in the appendix Fig 6 and Fig 7.

Fmax (clock_50) Raw Input	145.62 MHz
Fmax (clk) Input	115.47 MHz
Total Registers	416
Memory Use	15%

We tested the instructions on modelsim using the generated MIF file, and the output signals were used to verify if the output was intended. Debug signals were added to help with this, as seen in the image below.

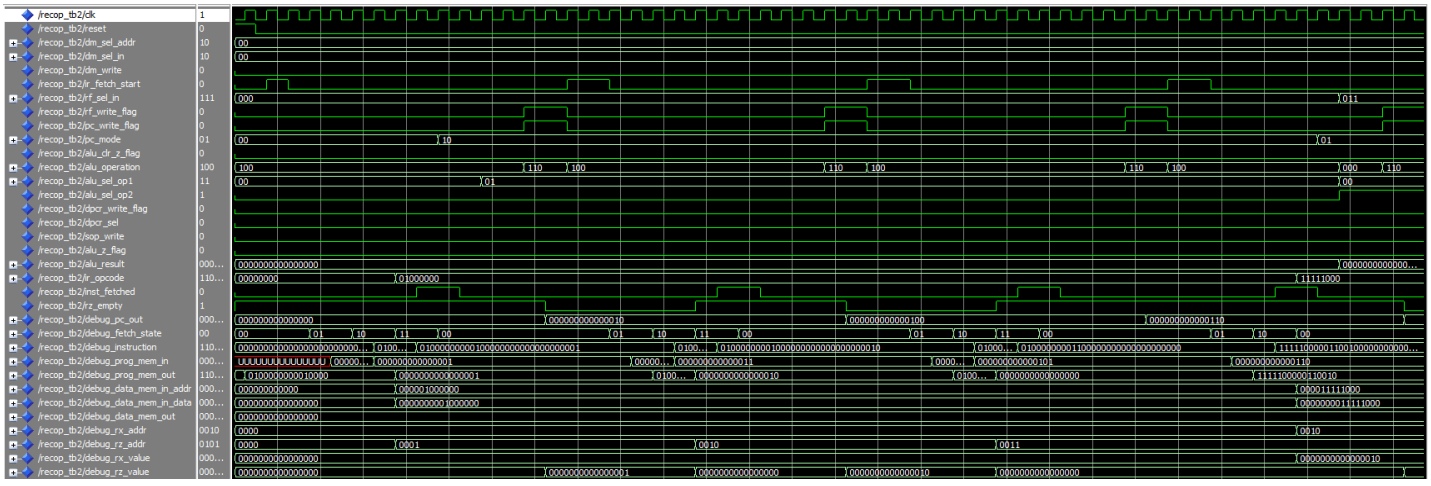
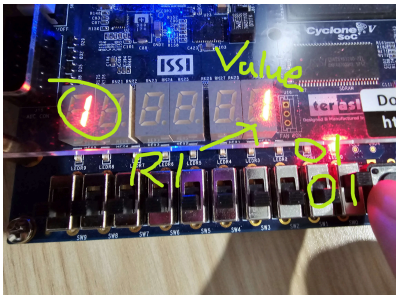
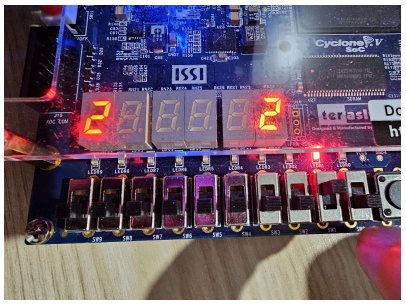
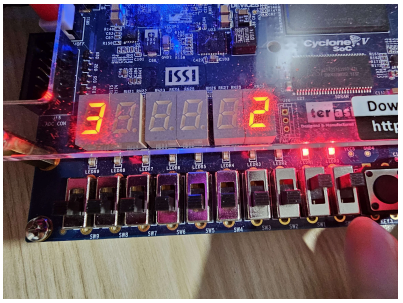
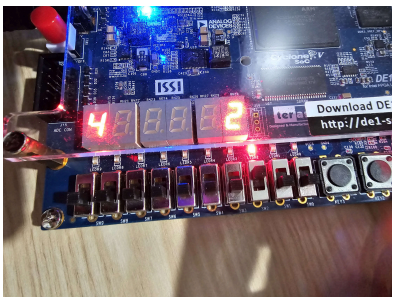
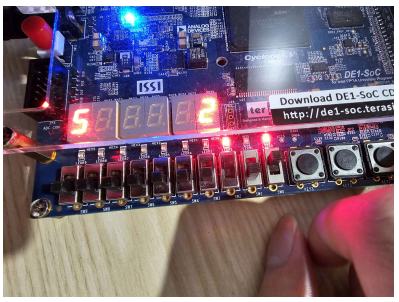


Figure 5. Screenshot of testing on Modelsim

Testing on the DE1 required us to program the DE1 with our code and upload the mif instructions onto the DE1, the process of this is stated in the ReadMe. The table with images below shows the expected output for the assembly instructions shown above in the assembly compiler images. Where the switches control the register to be displayed on the 7-segment display, the left-most display shows the register number and the right-most display shows the value stored in the register.

Reg	Expected Value	Image
R1	1	
R2	2	
R3	2	
R4	2	

R5	2	

Appendix

Flow Status	Successful - Fri May 02 23:11:29 2025
Quartus Prime Version	18.1.0 Build 625 09/12/2018 SJ Lite Edition
Revision Name	recop
Top-level Entity Name	recop
Family	Cyclone V
Device	5CSEMA5F31C6
Timing Models	Final
Logic utilization (in ALMs)	428 / 32,070 (1 %)
Total registers	416
Total pins	67 / 457 (15 %)
Total virtual pins	0
Total block memory bits	589,824 / 4,065,280 (15 %)
Total DSP Blocks	0 / 87 (0 %)
Total HSSI RX PCSs	0
Total HSSI PMA RX Deserializers	0
Total HSSI TX PCSs	0
Total HSSI PMA TX Serializers	0
Total PLLs	0 / 6 (0 %)
Total DLLs	0 / 4 (0 %)

Figure 6. Resource Utilization

	Fmax	Restricted Fmax	Clock Name	Note
1	115.47 MHz	115.47 MHz	control_unit:impl_control_unit|alu_operation_signal[0]	
2	145.62 MHz	145.62 MHz	clock_50	

Figure 7. Fmax