Fenerbahce University

Computer Architecture
FB-CPU RTL Design
Project Content

1. Definition:

Within the scope of this project, RTL design of a processor named FB-CPU with Verilog language and various code snippets written in machine language on the designed processor will be written. At the end of the project, it will be observed how RAM, Control Unit and Stores in a simple processor can work together and execute code snippets in machine language.

2. Project team:

The project will consist of teams of 4 people. Each project team will have a supervisor. Students should determine their own project team of 4 people and the project manager.

3. Tools to be used:

3.1. Xilinx Vivado Design Suite

Xilinx Vivado Design Suite is used to create the design required for working on FPGA development boards. It takes the hardware design languages such as Verilog, VHDL etc. and creates the design file that can be configured to the FPGA (files with .bit extension for Xilinx company FPGAs).

Download Address: https://www.xilinx.com/support/download.html

Installation and Licensing Video: https://www.youtube.com/watch?v=yW1bJbXnbRU

4. Design Requirements

Under this section, the requirements of the FB-CPU to be designed are given.

For the processor design, the initial design developed in Verilog language is given.

Initial design: http://www.levent.tc/files/courses/digital_design/project/fbcpu_baslangic.rar

There are 6 files in the initial design rar archive. These;

- fbcpu_core.v: Contains the design of the processor.

- tb_fbcpu.v: It is the testbench design that feeds the commands that will test the processor and checks whether the result is correct.
- memory.v: It is the memory designed as Block RAM where commands and data are kept.
- testCase1.v: Contains the 1st software given in the sample software title.
- testCase2.v: Contains the 2nd software given in the sample software title.
- testCase3.v: Contains the 3rd software given in the sample software title.

The testbench design given in the tb_fbcpu.v file uses fbcpu_core.v, memory and testCase1-2-3 files as submodules. In Figure 1, the connections of modules to each other in testbench are given.

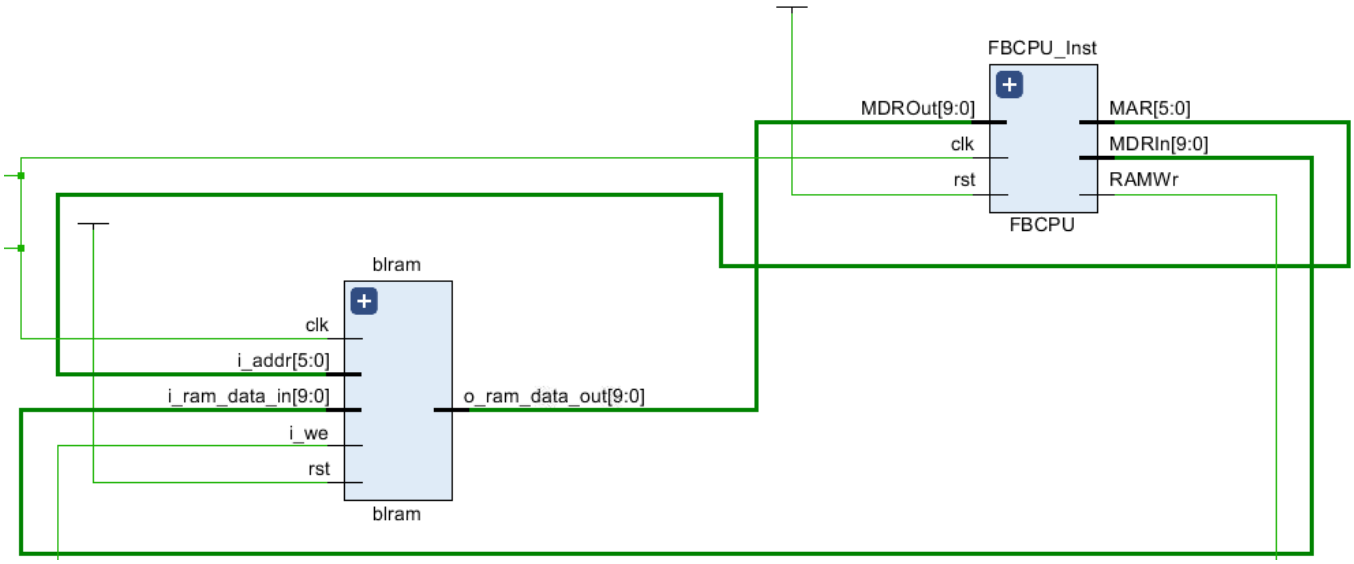

Figure 1. Testbench Links

The desired FBCPU RTL design is in Von Neumann architecture. Figure 2 gives the Von Neumann Architecture.

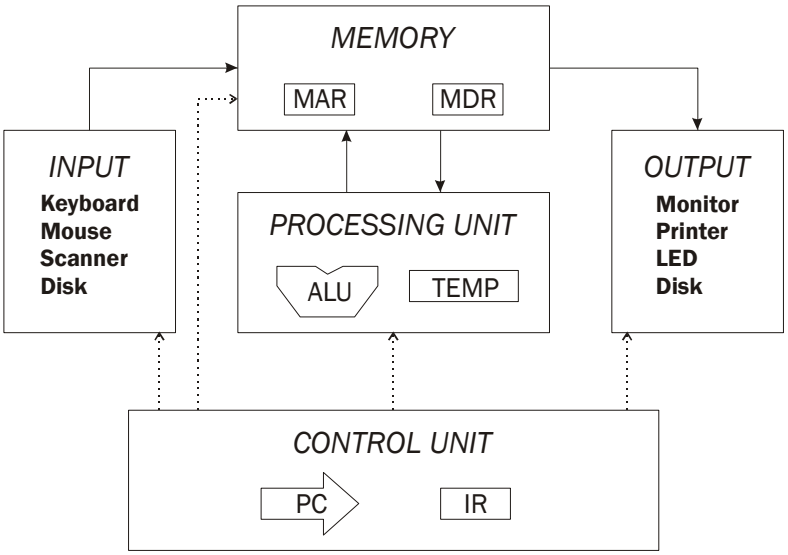

Figure 2. Von Neumann Architecture

It basically has 4 elements.

- Registers (Temp variable under Processing Unit in Figure 2)
- Memory (RAM)
- Processing Unit (ALU)
- Control unit

supported by the FB-CPU are given in Table 1.

Table 1. FB-CPU ISA (Instruction Set Architecture)

Command Name	**Mission**	**Operation Code**
LOD ADDR	Load takes the value from the given address in the Memory and places it in the ACC register. ACC = *(ADDR)	0000
STO ADDR	Store (Store) takes the value inside the ACC and writes it to the address given in the memory. *(ADDR) = ACC	0001
ADD ADDR	It takes the value at the given address in memory, sums it up with ACC and overwrites ACC. ACC = ACC +*(ADDR)	0010
SUB ADDR	It takes the value at the given address in memory, subtracts it with ACC, and overwrites ACC. ACC = ACC - *(ADDR)	0011
MUL ADDR	It takes the value at the given address in memory, multiplies it by ACC, and overwrites ACC. ACC = ACC * (*(ADDR))	0100
JMP NUMBER	PC = Number.	0110
JMZ NUMBER	If the value of ACC is 0, it assigns the given number value to the PC, otherwise it does not operate.	0111
NOP	No Operation, no operation is performed.	one thousand
HLT	The application stops	1001

The processor supports 9 instructions.

Figure 3 shows the separation of bits of the 10-bit instruction of the FB-CPU for operation and address.

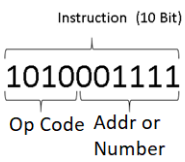

Figure 3. FB-CPU Sample Command Binary Display

The state diagram of the FB-CPU is given in Figure 4. It shows together the tasks that the processor has to do step by step.

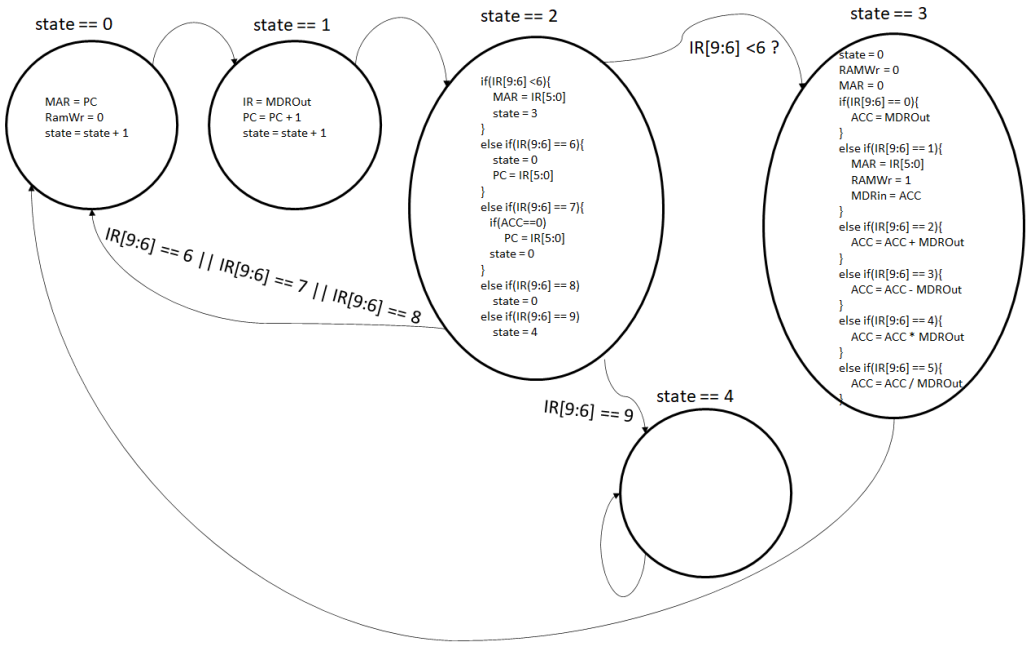

Figure 5. FB-CPU State Machine Display

4.1. Registers

All registers required by the FBCPU are defined in the initial design. There are 4 holders in the design.

These;

- state: In the state machine, the state information is kept.
- PC: Information is kept at which address in RAM the command is running.
- IR: The currently running command itself is kept.
- ACC: Temporary storage area.

Figure 6 shows the registers defined in the fbcpu_core.v file.

```
always@(posedge clk) begin
    durum       <= #1 durumNext;
    PC          <= #1 PCNext;
    IR          <= #1 IRNext;
    ACC         <= #1 ACCNext;
end
```

Figure 6. FBCPU Registers

No new register will be added. All necessary savers are included in the design.

It will be implemented with the FB-CPU state machines method.

The memory signals connected to the I/O ports in the design are given below.

- **MAR (6 Bit):** It is a register called Memory Address Register. This register is connected to the address input of the RAM. Since RAM has 2^6 locations, MAR is 6 bits. The store is in RAM.
- **MDRIn (10 Bit):** Memory Data Register In is the register used when a data is to be written to RAM. Since RAM has a location of 10 bits, the register is 10 bits. The store is in RAM.
- **RAMWr (1 Bit):** It is activated when data will be written to RAM. If it is not 1, no data is written to RAM. The store is in RAM.
- **MDROut (10 Bit):** Memory Data Register is the register used when reading data from RAM. Since RAM has a location of 10 bits, the register is 10 bits. The store is in RAM.

4.2. Memory (RAM, Random Access Memory):

There is a Block RAM mechanism where the FB-CPU reads the commands and writes back the calculated values. This memory is not available in the fbcpu_core.v file. The memory that the testcode instantiates is located in the memory.v file. There are 4 registers, clock and reset signals connected to RAM. The functions of the registers attached to RAM are explained in the registers section.

4.3. Processing Unit (ALU, Arithmetic Logic Unit): It is the section where arithmetic operations are performed. on FB-CPU There are 3 arithmetic operations. These are addition, subtraction and multiplication, they perform operations according to the incoming operation code and write them to the ACC register.

4.4. Control Unit: Registers are responsible for transferring data to Arithmetic Processing Unit and RAM between each other. It manages the intra-processor data flow.

The initial verilog design for the processor design is missing the Process and Control Unit. These missing places are in states 2 and 3 in the state machine.

5. Design of Missing Units

In the given initial design, the design for state 0 and state 1 is given.

Case 2 will be designed in the LAB.

State 3 must be completed for the processor to be operational.

6. Sample Software

In FB-CPU 10-bit instruction, the first 4 bits [9:6] represent the operation code and the last 6 bits [5:0] represent the address. Applications should be developed with the commands given in Table 1. The correct operation of the processor can be tested with the following code snippets. Whichever software you want to test with, the line “TEST_CASE = 1” at the top of the testbench can be replaced with one of the numbers 1-2-3. In other words, if it is desired to test with the second software, it should be arranged as "TEST_CASE = 2". In this case, Testbench will load the 2nd test codes into memory.

6.1. Test Software 1

Develop an application for FB-CPU that records the sum of two numbers at address 50 and 51 at address 52 in memory.

0: 0000_110010 // LOD 50, (ACC = *50), Hex = 32
1: 0010_110011 // ADD 51, ACC = ACC + (*51), Hex = B3
2: 0001_110100 // STO 52, (*52) = ACC, Hex = 74
3: 1001_000000 // Halt, Hex = 240
50: 0000000101 // Hex = 5
51:000001010 // Hex = A

6.2. Test Software 2

Develop an application for FB-CPU that records the product of two numbers at address 50 and 51 in memory at address 52.

0: 0000_110010 // LOD 50, (ACC = *50), Hex = 32
1: 0100_110011 // ADD 51, ACC = ACC * (*51), Hex = 133
2: 0001_110100 // STO 52, (*52) = ACC, Hex = 74
3: 1001_000000 // Halt, Hex = 240
50: 0000000101 // Hex = 5
51:000001010 // Hex = A

6.3. Test Software 3

Develop an application for FB-CPU that records the product of two numbers at address 50 and 51 in memory at address 52. However, do not use the multiplication operation. For multiplication, add the number in 50 times the number in 51 and write it to address 52. You can use any addresses you want for the required variables.

0: 0000_110011 // LOD 51, ACC = *51, Hex = 33
1: 0011_110001 // SUB 49, ACC = ACC - *49, Hex = F1
2: 0111_001010 // JMZ 10 will exit the loop if the loop is finished (ACC-49 == 0), Line 10, Hex = 1CA
3: 0000_110000 // LOD 48, load temp, 0 at startup, Hex = 30
4:0010_110010 // ADD 50, add the second number above ACC, Hex = B2
5: 0001_110000 // STO 48 assign value of ACC to temp, Hex = 70
6: 0000_110001 // LOD 49, ACC = i, Hex = 31
7:0010_101110 // ADD 46, ACC = i + 1, Hex = AE
8: 0001_110001 // STO 49, i = i + 1, Hex = 71
9: 0110_000000 // JMP 0, return to the beginning of the loop 0th line, Hex = 180
10: 0000_110000 // LOD 48, ACC = temp, Hex = 30
11: 0001_110100 // STO 52, *52 = ACC, Hex = 74
10: 1001_000000// HLT, finish, Hex = 240

46: 1 // number 1
48: 0 // Hex = 0, temp
49: 0 // Hex = 0 for index i
50: 0000000101 // Hex = 5
51:000001010 // Hex = A

7. Grading and Project Delivery:

This topic contains information about project submission and grading of FB-CPU.

7.1. Grading:

The project has **two** main evaluation criteria. Both criteria are 50 points each.

is the correct execution of the instructions (instructions) of the FB-CPU . FB-CPU has 9 instructions. Evaluation will be made according to whether each command produces correct results.

The second criterion is **Project Delivery Document and Presentation** .

- **Project Delivery Document:**

Students will complete their project reports by filling in the "Project Delivery Document".

Project Delivery Document: http://www.levent.tc/files/courses/digital_design/project/BLM201_proje_delivery_dokumani.docx

In the project delivery document, delete the yellow marked places and write the relevant contents. The project delivery document should be at least 3 and at most 5 pages.

- **Project Presentation:**

They should prepare and record a presentation on Powerpoint that will last an average of 5 minutes (4-6 minutes). Recording can be done with mobile phone or computer screen recording software (Screen-Recorder, Bandicam etc…).

The presentation should be recorded by one of the team members, while explaining how the project is done, how the processor works, etc. on powerpoint slides. Powerpoint slides can be read in the presentation video and the speaker's voice must be understandable. Powerpoint slide view design can be done as desired.

The entire project team receives their grades based on this assessment.

7.2. Delivery:

The following steps are required for the delivery of the project. The requested files should be brought only by the project team leader and uploaded to Teams and Github (A very common open source code sharing platform).

The following files should be uploaded to the "Project Delivery" page of the project team manager, which is opened in Teams.

- Verilog RTL Design (Completed fbcpu_core.v file)
- Prepared powerpoint presentation file (file with .ppt extension)
- Project Delivery Document (must be uploaded in Word format)
    - Subheadings of the document must be filled

- The recorded powerpoint presentation video should be uploaded to youtube, and the address and link of the document should be written in the opened place in the results section (If you want the video not to be visible to everyone, after uploading it to youtube, select the unlisted option, and only those who have the link should have the link. you can see it).
- All files uploaded to Teams (Completed fbcpu_core.v file, ppt extension presentation file and Project Delivery Document (in PDF format)) should be subscribed to github.com site, uploaded and the link of the document should be written to the place in the results section.