Logical System Design – BLM 201

Week 10: FB CPU
RTL Design

Fenerbahce University

10 . Week Content

- FB-CPU
  - RTL Design

FB-CPU

- It is an educational processor to explain the basic working principles of FB-CPU processors.

- Von neumann architecture

FB-CPU

Processor;

- Memory (RAM)
- Register
- Control unit
- Arithmetic Logic Unit

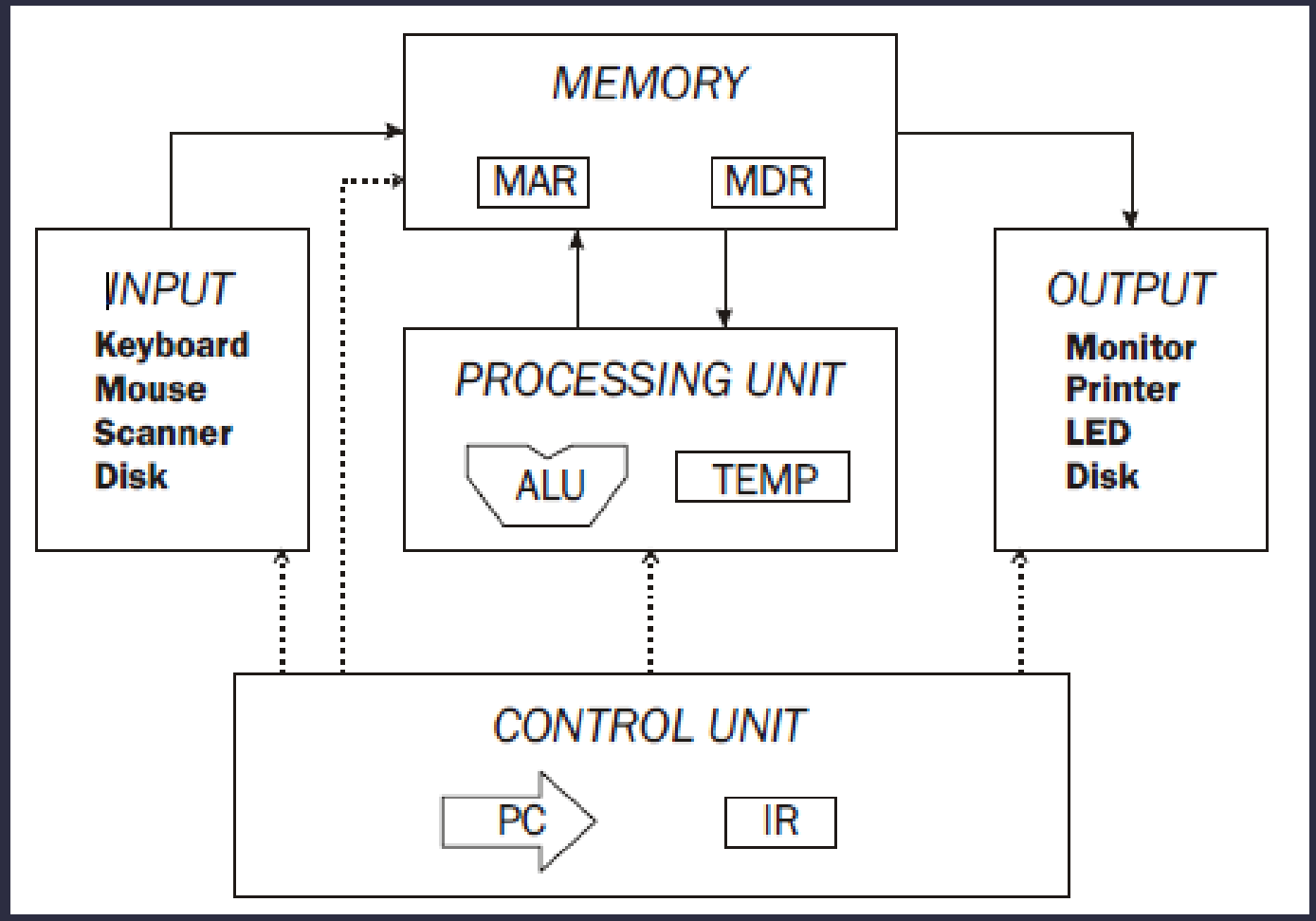
includes structures.

FB-CPU

Tools to be used while designing the processor;

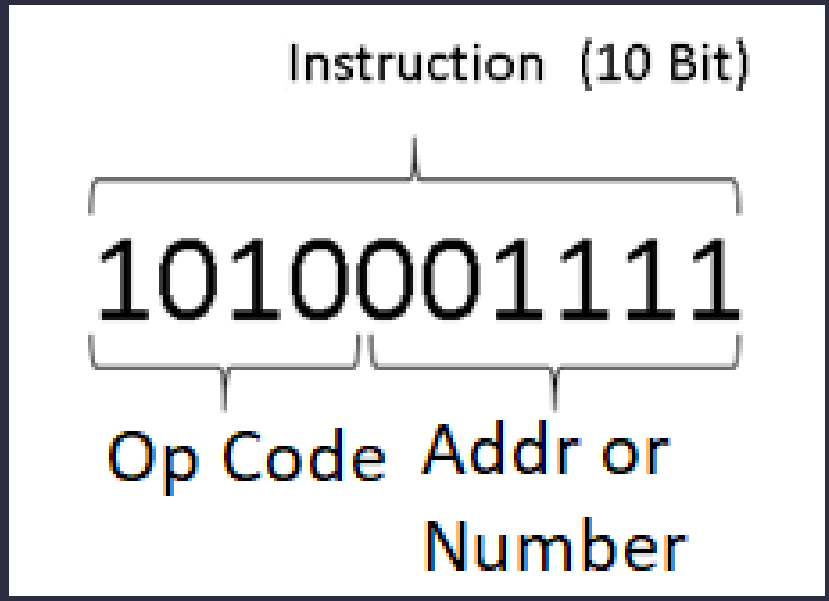
- Xilinx vivado
  - RTL design will be done

FB-CPU

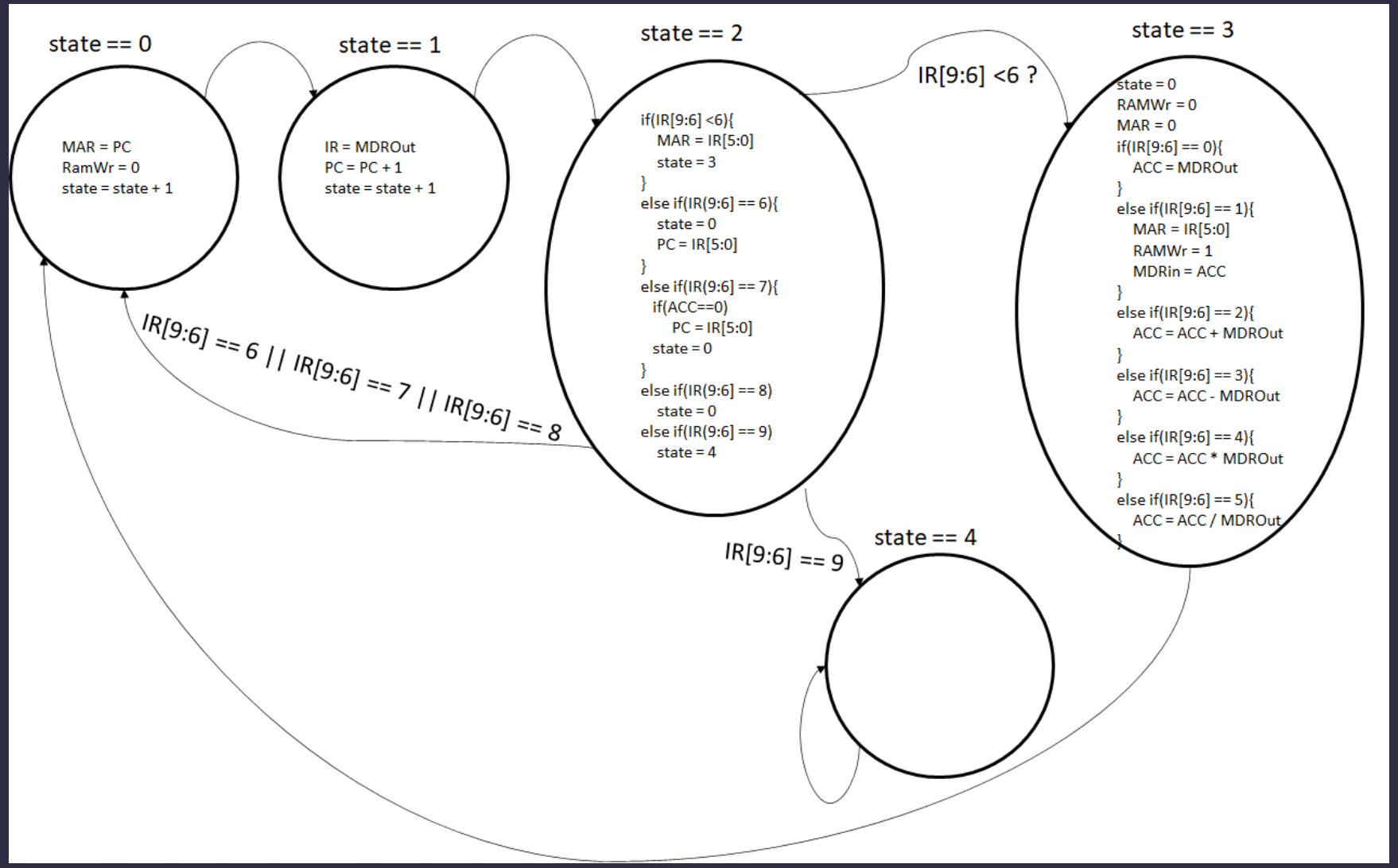

MEMORY
MAR
MDR
INPUT
Keyboard
Mouse
Scanner
Disk
PROCESSING UNIT
ALU
TEMP
OUTPUT
Monitor
Printer
LED
Disk
CONTROL UNIT
PC
IR

FB-CPU

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.	1000
HLT	The application stops	1001

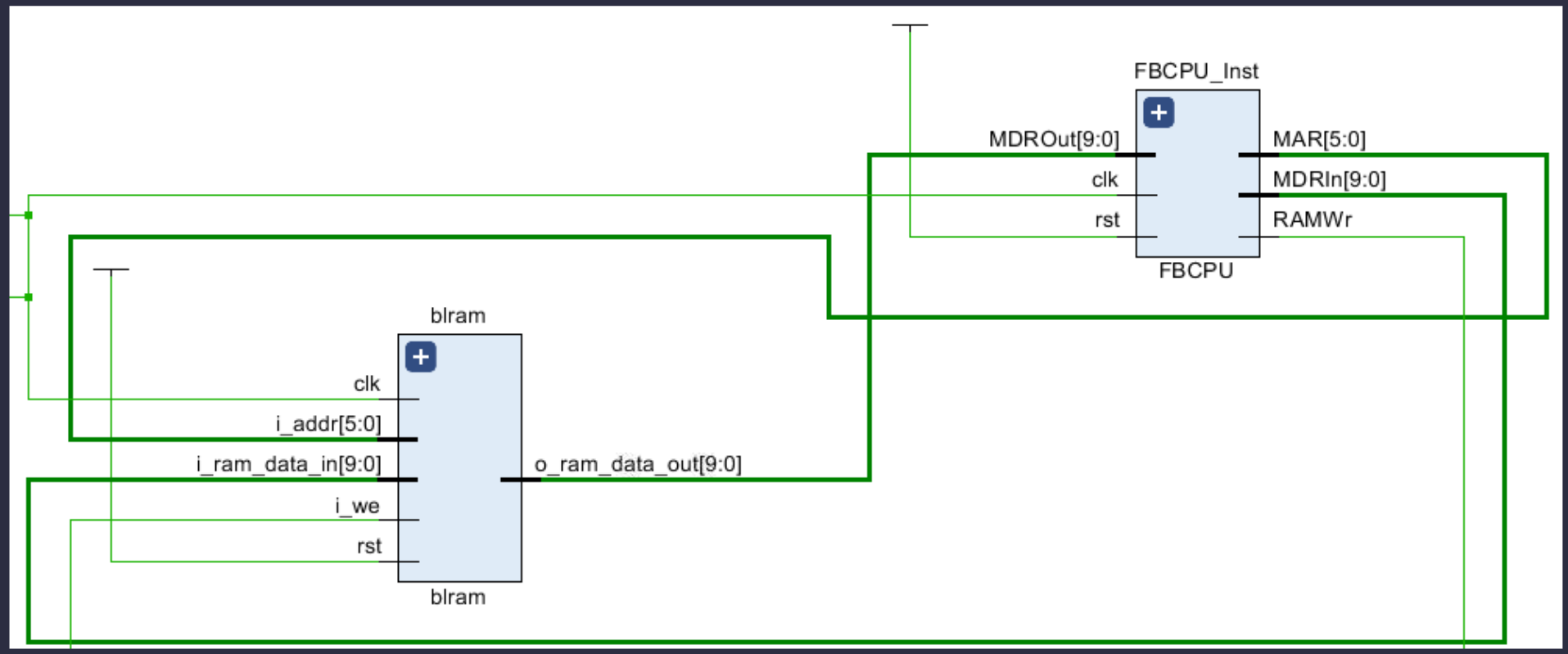


FB-CPU


state == 0
MAR = PC
RamWr = 0
state = state + 1
state == 1
IR = MDROut
PC = PC + 1
state = state + 1
state == 2
if(IR[9:6] <6){
MAR = IR[5:0]
state = 3
}
else if(IR[9:6] == 6){
state = 0
PC = IR[5:0]
}
else if(IR[9:6] == 7){
if(ACC==0)
PC = IR[5:0]
state = 0
}
else if(IR[9:6] == 8)
state = 0
else if(IR[9:6] == 9)
state = 4
IR[9:6] <6 ?
state == 3
state = 0
RAMWr = 0
MAR = 0
if(IR[9:6] == 0){
ACC = MDROut
}
else if(IR[9:6] == 1){
MAR = IR[5:0]
RAMWr = 1
MDRin = ACC
}
else if(IR[9:6] == 2){
ACC = ACC + MDROut
}
else if(IR[9:6] == 3){
ACC = ACC - MDROut
}
else if(IR[9:6] == 4){
ACC = ACC * MDROut
}
else if(IR[9:6] == 5){
ACC = ACC / MDROut
}
IR[9:6] == 6 || IR[9:6] == 7 || IR[9:6] == 8
IR[9:6] == 9
state == 4

FBCPU

Tools to be used while designing the processor;



FB-CPU

There are 4 register in the design.

- 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

FB-CPU

There are 4 register in the design.

- **Register declarations**

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

FB-CPU

- All other registers would work according to the change of state register. That is, the input signals of all registers change according to the value of the state.

- In other words, other signals are assigned to the registers according to the value of the state register, and the progress of the system depends on the status signal.

FB-CPU

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

- **MAR (6 Bits):** Memory Address It is a register named 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 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 .

Design of Missing Units

- In the given initial design, the design for state 0 and state 1 is given.
- State 2 will be designed in the LAB .
- State 3 must be completed for the processor to be operational.

FB-CPU

Sample Software 1

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

FB-CPU

Sample Software 2

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

FB-CPU

Sample Software 3

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

FB-CPU

Sample Software 3

- 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, finishing , 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

FB-CPU Delivery

Project Delivery Document and Presentation

FB-CPU

FB-CPU Delivery

- Project Delivery Document.
- In addition, the following files should be uploaded to the "Project Submission" page opened in LMS .
- Processor design made on Vivado (file with .v extension)
- Prepared powerpoint presentation file (file with .ppt extension)
- Project Delivery Document (must be uploaded in Word format) o Sub-headings of the document should be filled
- Recorded powerpoint presentation video It should be uploaded to youtube , the address and the link to the opened place in the results section of the document should be written (If you want it not to be visible to everyone , you can choose the unlisted option after uploading to youtube so that only people who have the link can see it).
- to Teams (Processor design file (.v), two software in machine language, ppt extension presentation file and Project Delivery Document (in PDF format)) are registered to github.com site, uploaded and link to the place in the results section of the document. should be written.