A Simple 8-bit CPU Implementation on FPGA

Motivation

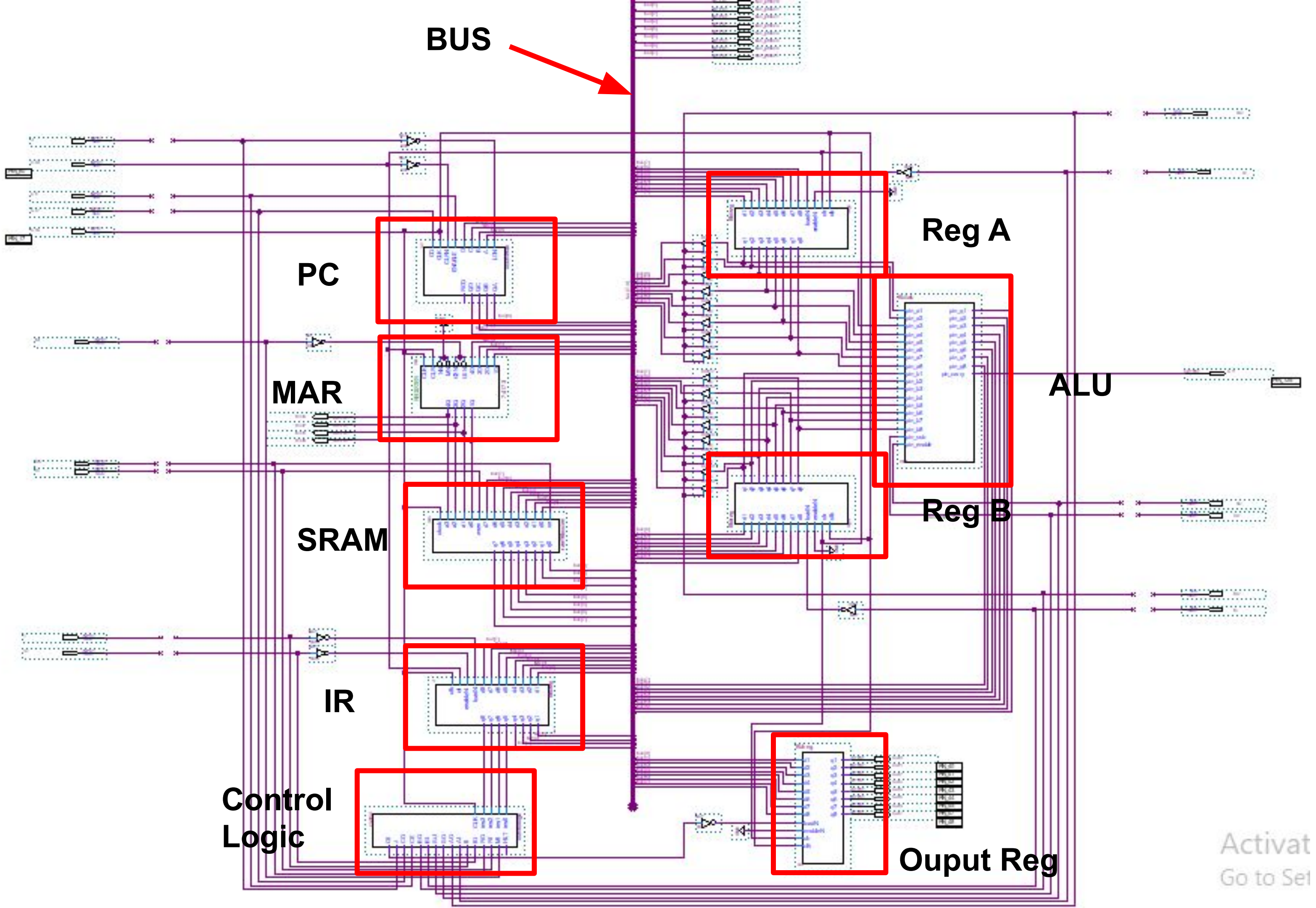
- How CPU execute instruction?
- What is micro-instruction?
- Learn FPGA

References

- https://www.youtube.com/user/eaterbc
- Alto: A personal computer (1979 Xerox)

Overall

- 8-bit instruction length
- 2 general registers
- 4 instructions: LDA, ADD, OUT, HLT
- 16 bytes memory space


BUS
PC
MAR
SRAM
IR
Control Logic
Reg A
ALU
Reg B
Ouput Reg

Bus

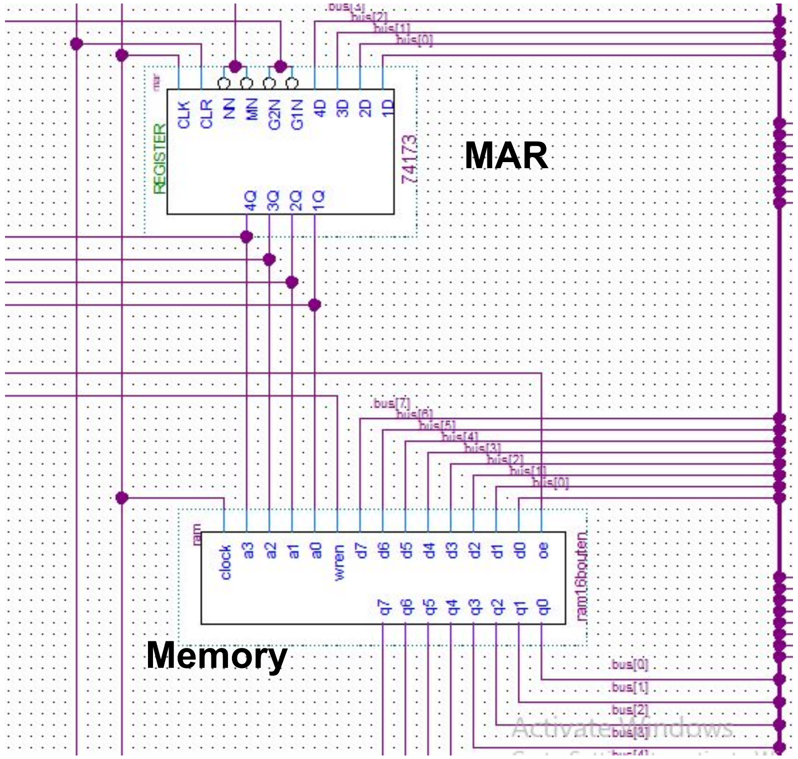
- Instruction length: 8 bits
  - [4 bits opcode][4 bits mem operand]

- 8-bits bus transfer both address and data
  - 4 bits address
  - 8 bits data

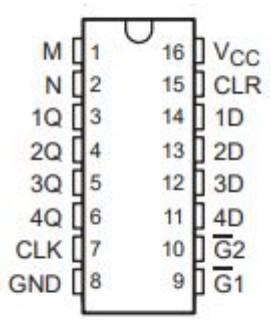
Register

- MAR: Memory Address Register (4 bits)
- IR: Instruction Register (8 bits)
- General Register A (8 bits)
- General Register B (8 bits)

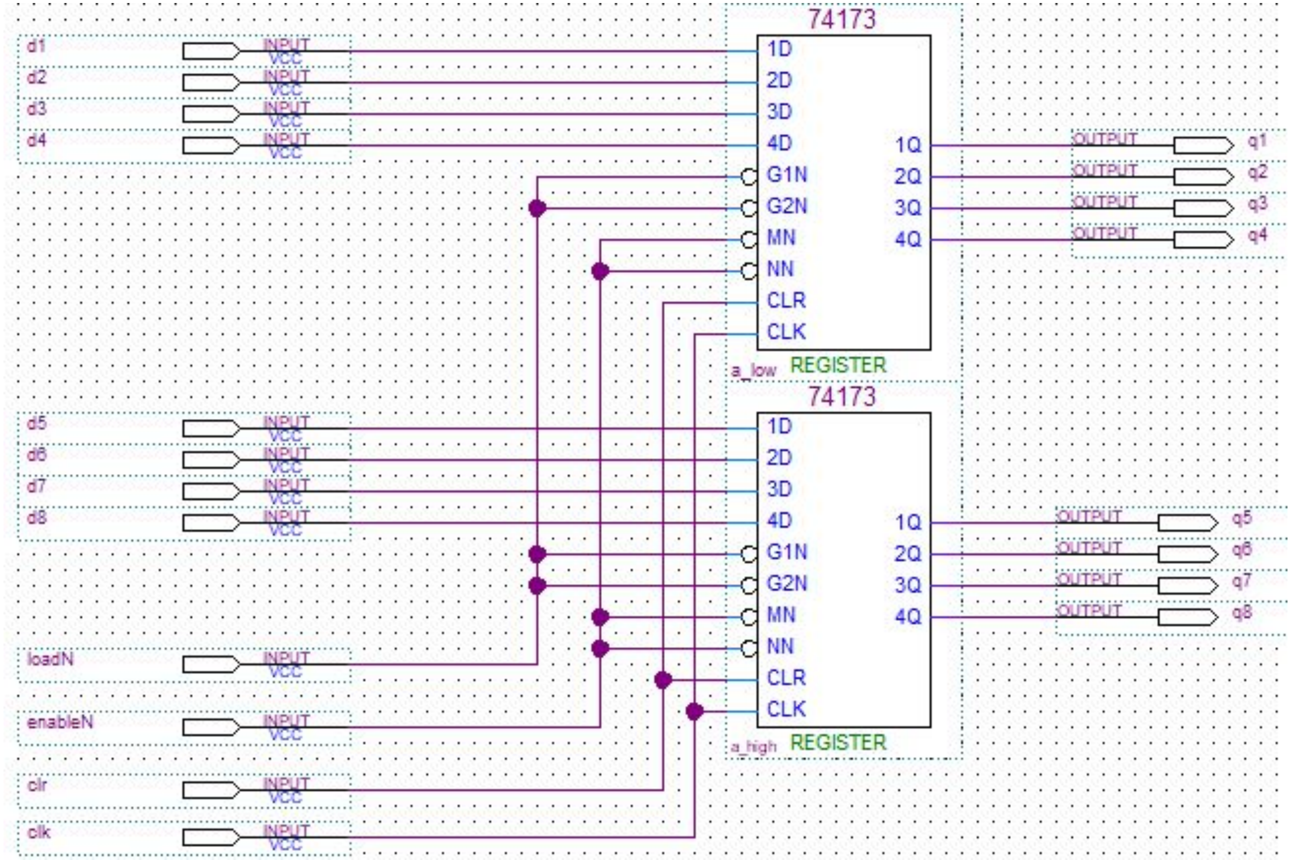
Memory Address Register



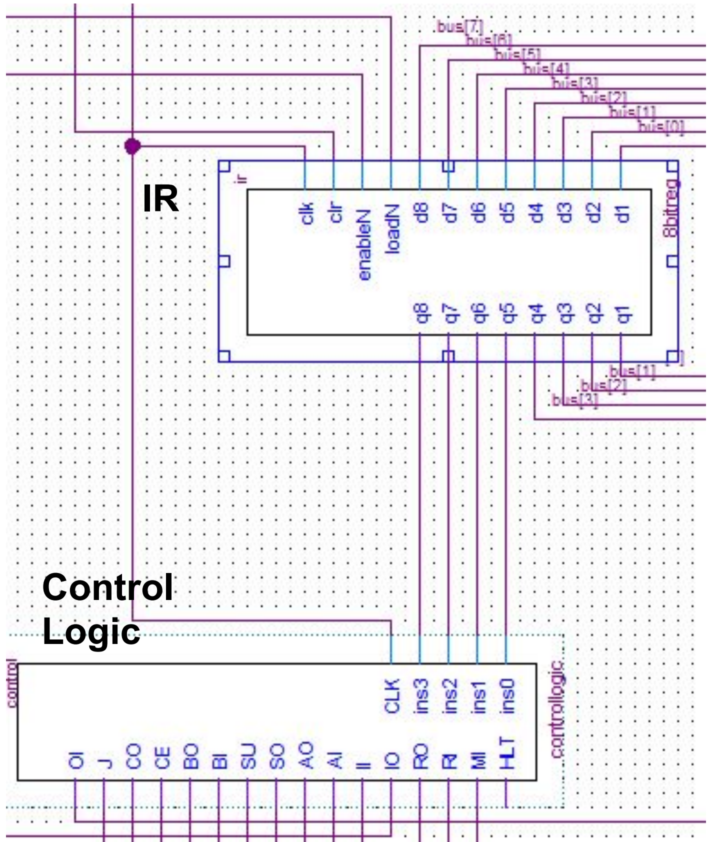
- 74173: 4-bit D-type Register (with 3-state Outputs)
- Control Signals
    - MI (MAR IN)



8-bit Register

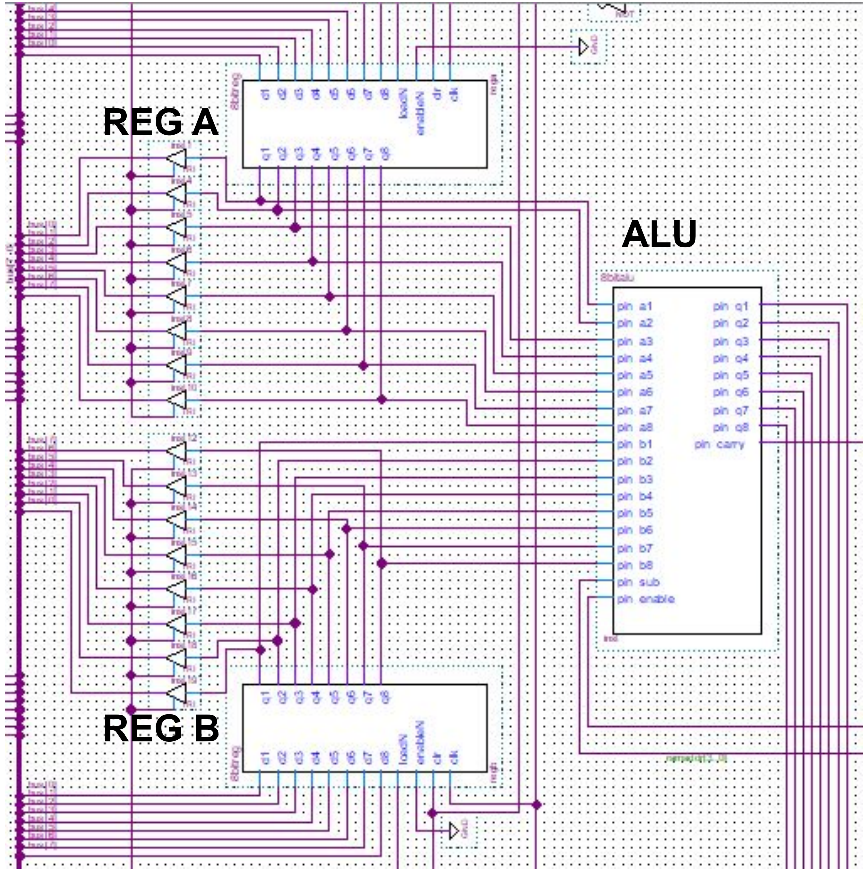


Instruction Register



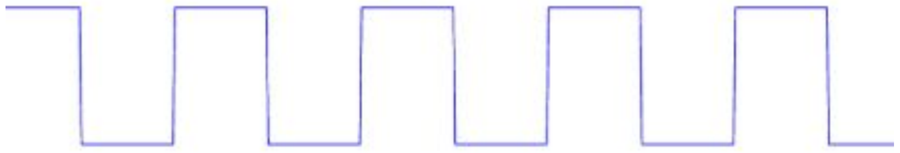
- Input from bus
- Output
  - High 4 bits to control logic
  - Low 4 bits to bus
- Control Signals
  - II (IR IN)
  - IO (IR OUT)

General Register



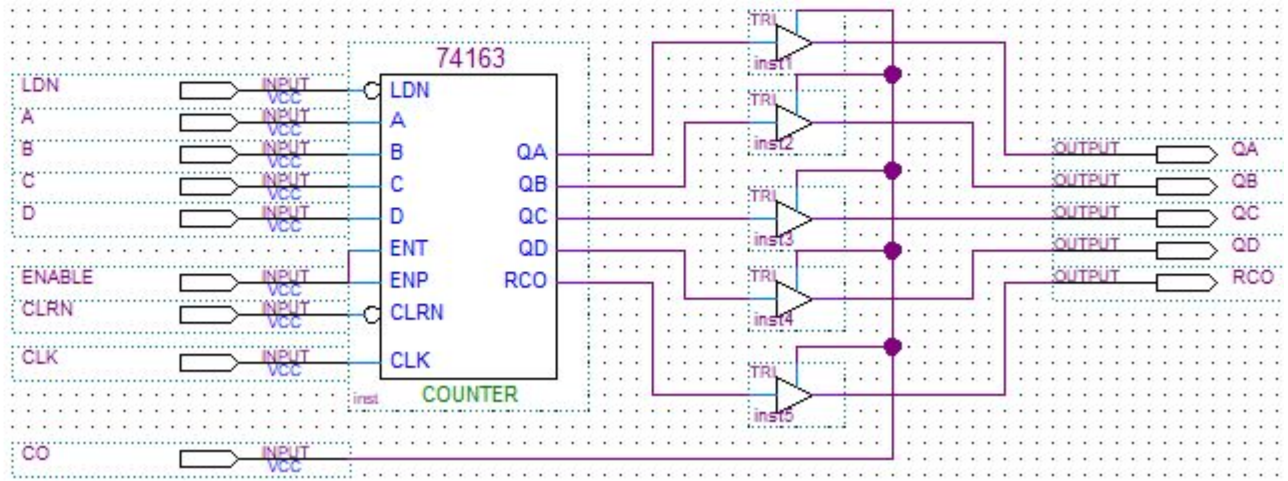
- Input from bus
- Output to both ALU and bus
- Control Signals
    - AI (A IN)
    - AO (A OUT)
    - BI (B IN)
    - BO (B OUT)

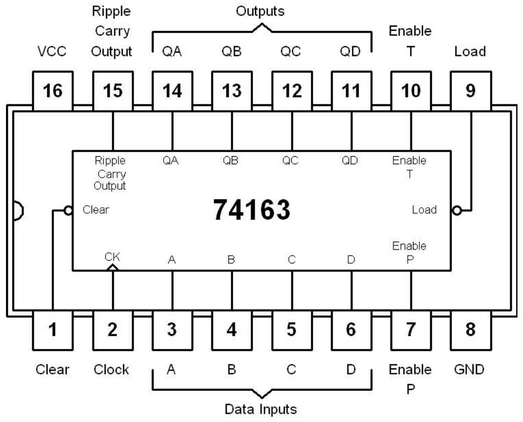
Clock Signal



- Square Wave
- This CPU works on rising edge
- Use Cyclone 2 FPGA’s internal clock signal (pin17)

4-bit Counter



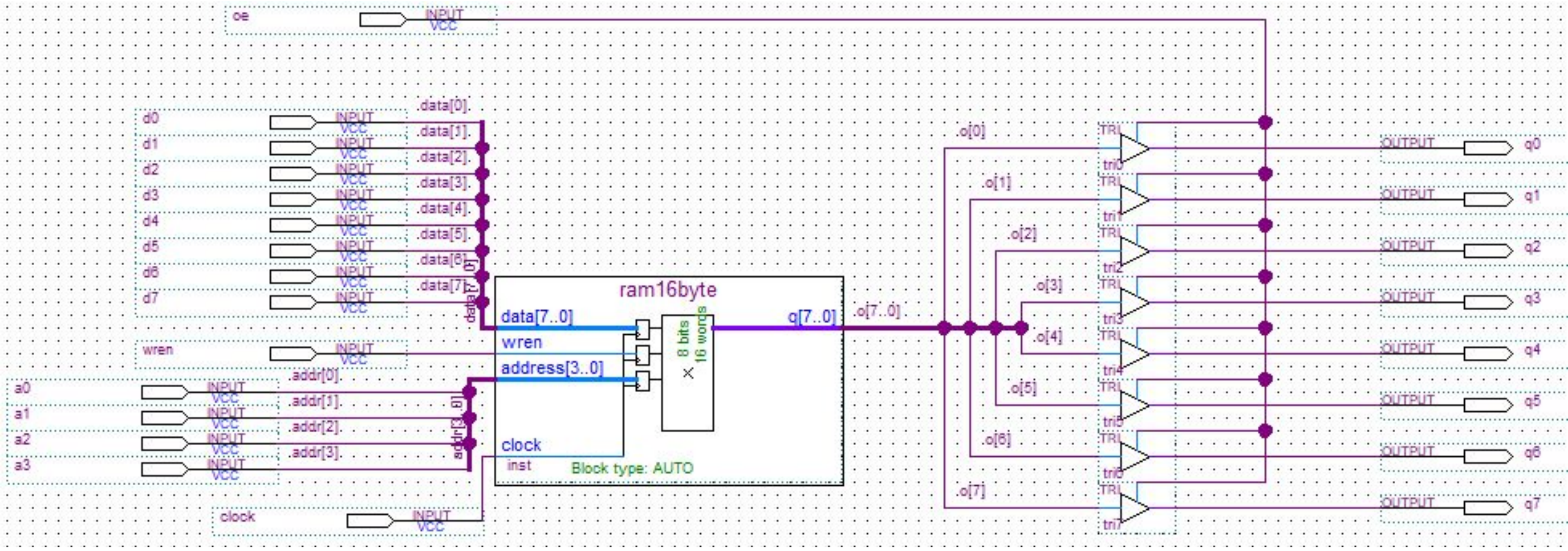


- Control Signals
  - CE (COUNTER ENABLE)
  - CO (COUNTER OUT)

Memory

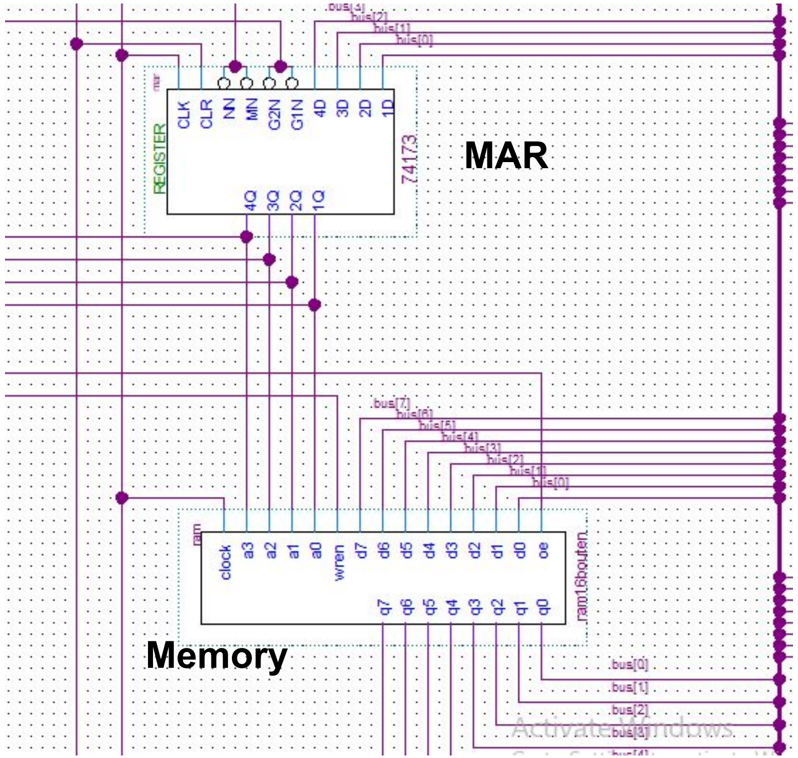
- SRAM
- No cache
- Same speed with CPU (1 clock cycle)
- Memory size: 16 byte (4 bits address)

Memory (cont’d)



- Quartus ram module

Memory (cont'd)



- Input: address from MAR and data from bus
- Output to bus
- Control Signals
  - RI (RAM IN)
  - RO (RAM OUT)

Use mif to set memory data

Control Logic

- Control Signals
  - HLT MI RI RO IO II AI AO SO SUB BI BO CE CO J OI
- How to use control signals to compose an instruction?
- Example:
  - LDA mem: Load memory into register A
  - LDA 0xE
  - Opcode 0x1 Memory Address 0xE
  - 0001 1110 (0x1E)

Instruction LDA

LDA instruction takes 4 clock cycle to finish

1. CO MI
2. RO II CE
3. IO MI
4. RO AI

Instruction Fetch

Decode

Execute & Writeback

Instruction LDA

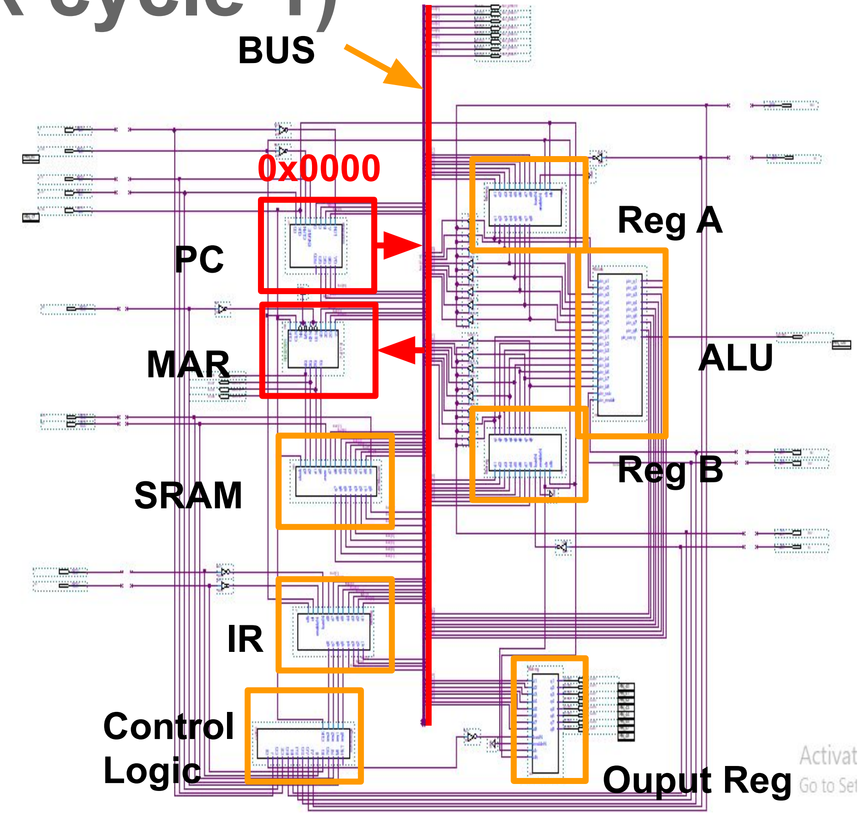
- Initial state:
  a. PC = 0x0000
  b. Memory

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
1E	0	0	0	0	0	0	0	0	0	0	0	0	0	0A	0

LDA 0xE

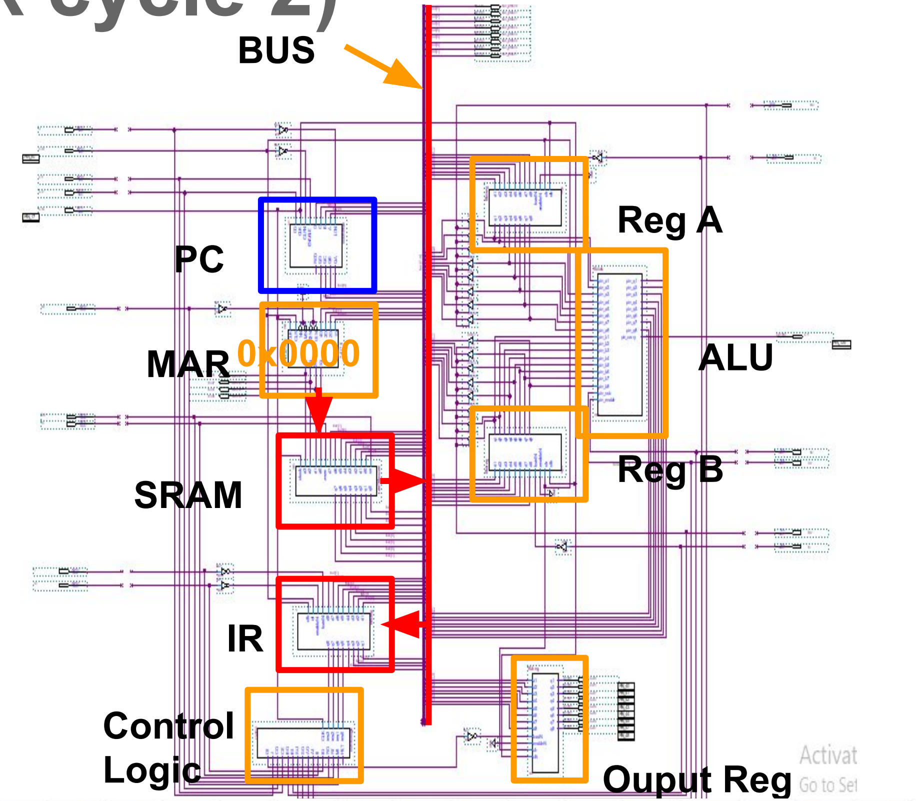
Fetch instruction (Clock cycle 1)

- CO (COUNTER OUT)
  MI (MAR IN)
- PC ---> MAR
- Result:
  - MAR = **0x0000**



Fetch instruction (Clock cycle 2)

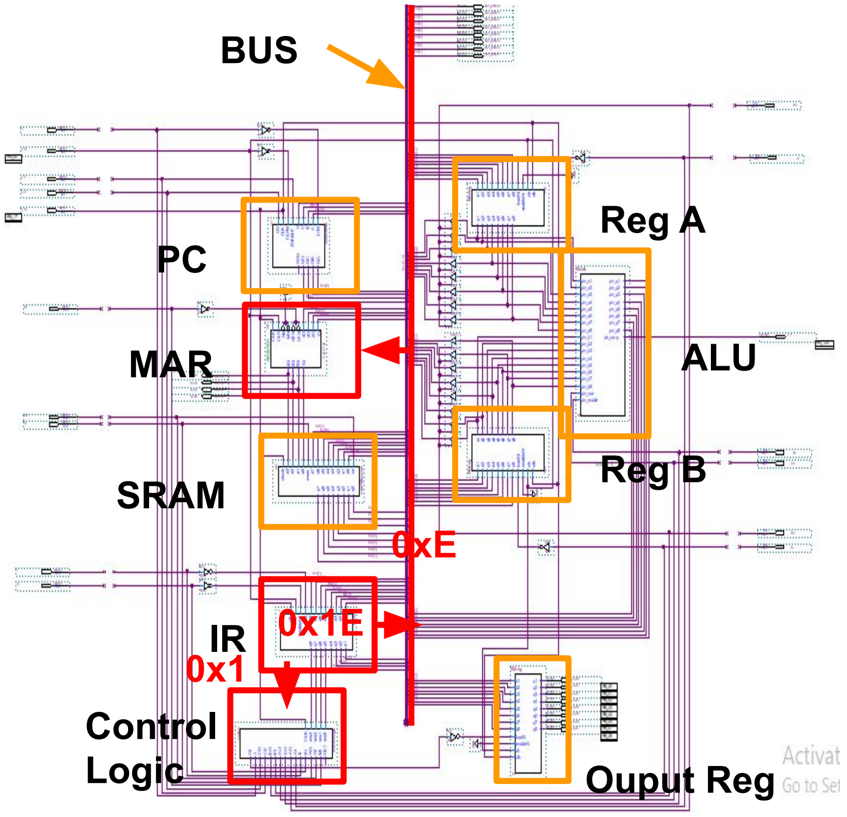
- RO (RAM OUT)
  II (IR IN)
  CE (COUNTER ENABLE)
- [0x0000] ---> IR
  PC = PC + 1
- Result:
  - IR = **0x1E**
  - PC = **0x1**



0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
1E	0	0	0	0	0	0	0	0	0	0	0	0	0	0A	0

Decode (Clock cycle 3)

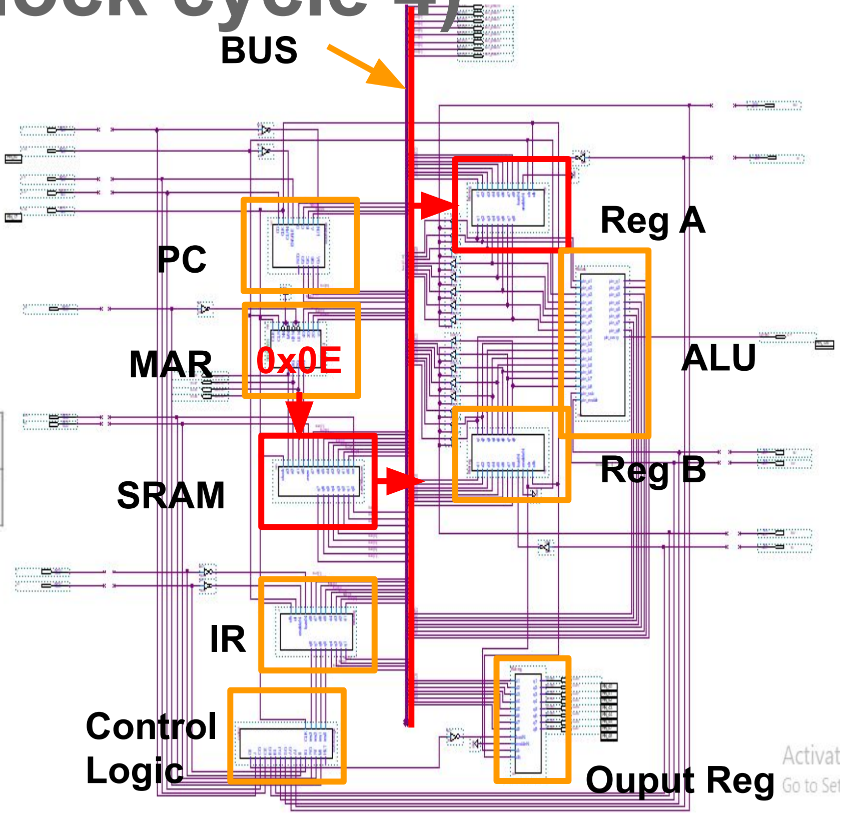
- IO (IR OUT)
  MI (MAR IN)
- IR[0..3] ---> MAR
- Result:
  - MAR = **0xE**



Execute & Writeback (Clock cycle 4)

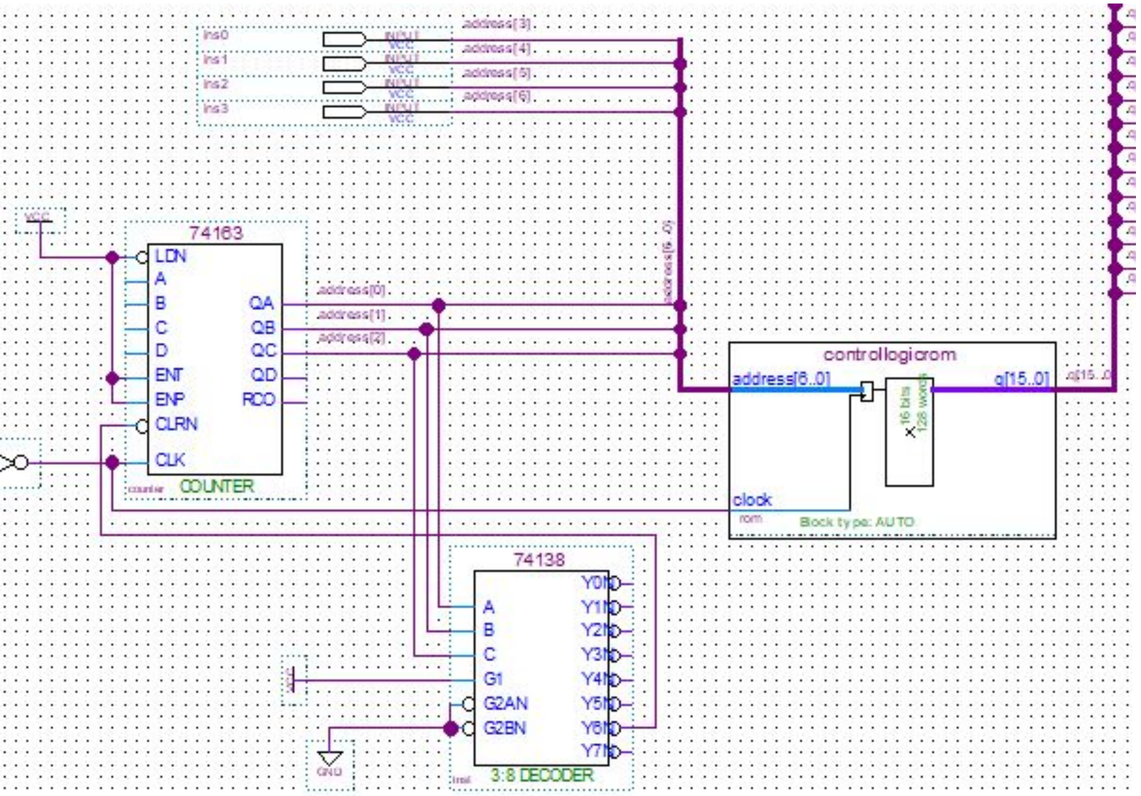
- RO (RAM OUT)
  AI (A IN)
- [0xE] ---> RegA

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
1E	0	0	0	0	0	0	0	0	0	0	0	0	0	0A	0

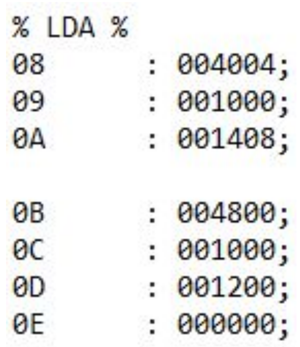


- Result:
  - RegA = **0x0A**

Control Logic (cont'd)

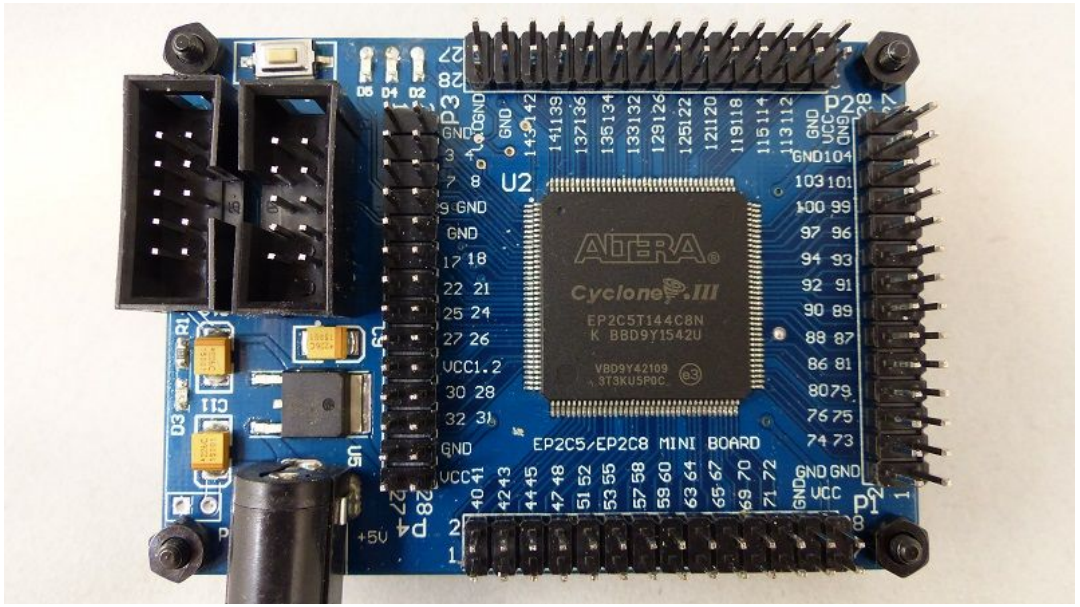


- 4 bits opcode as input
- 6 steps for every instruction
- ROM stores signals for each stage



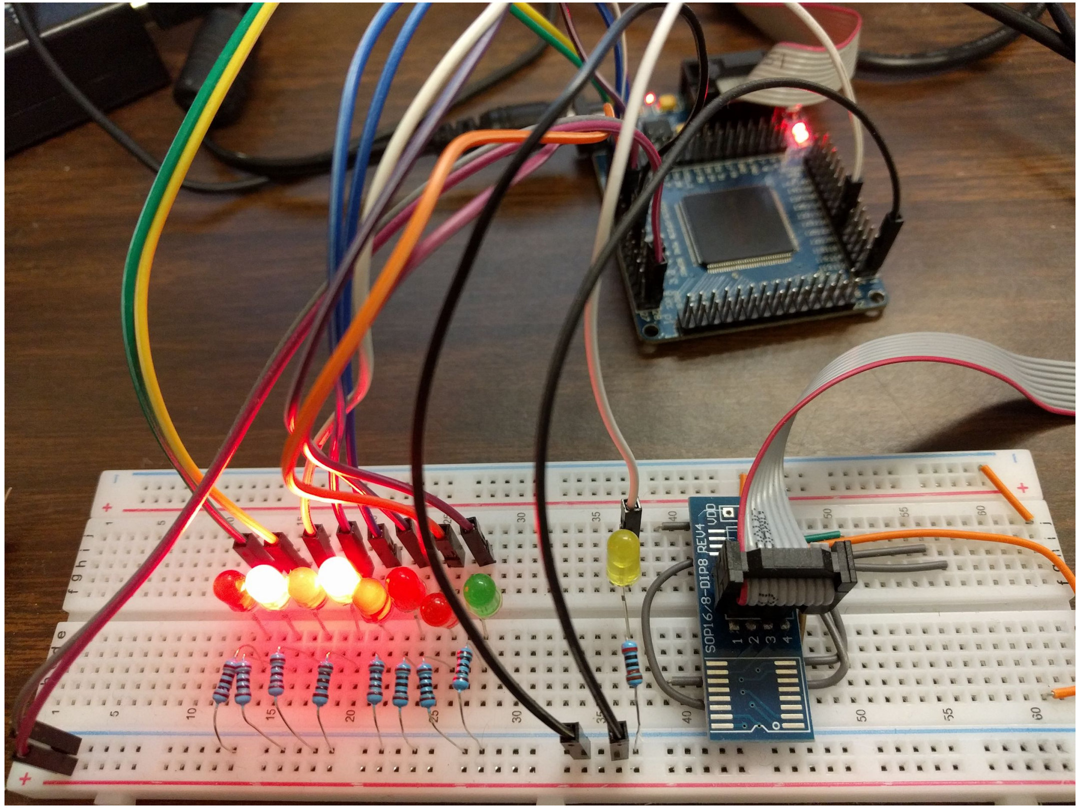
```
% LDA %
08      : 004004;
09      : 001000;
0A      : 001408;

0B      : 004800;
0C      : 001000;
0D      : 001200;
0E      : 000000;
```

FPGA

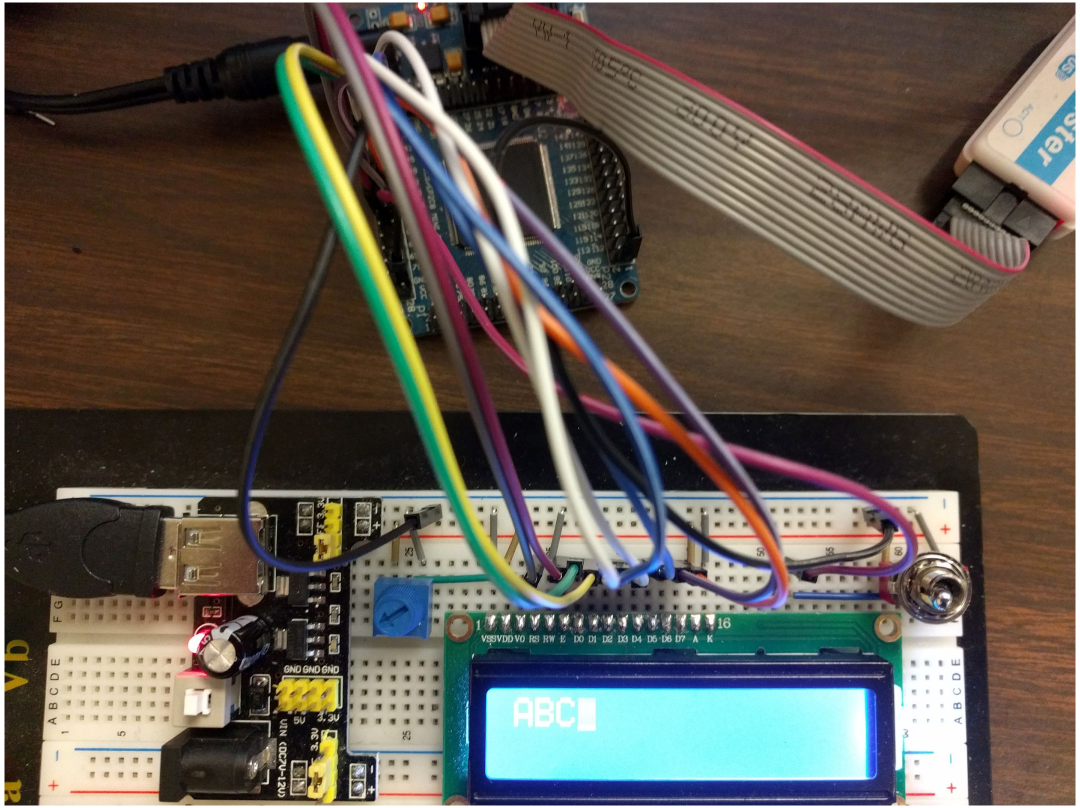


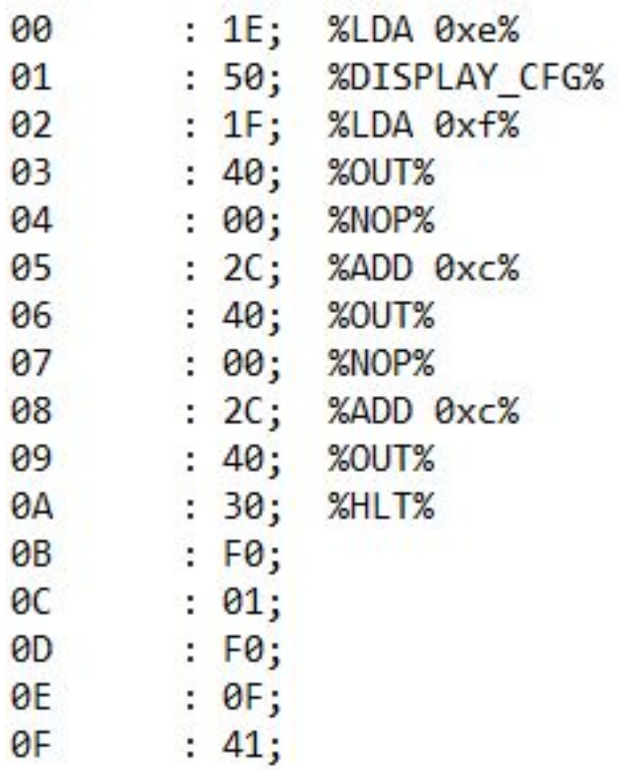
- Cyclone II EP2C5 Mini Dev Board

Display (8 LED lights)

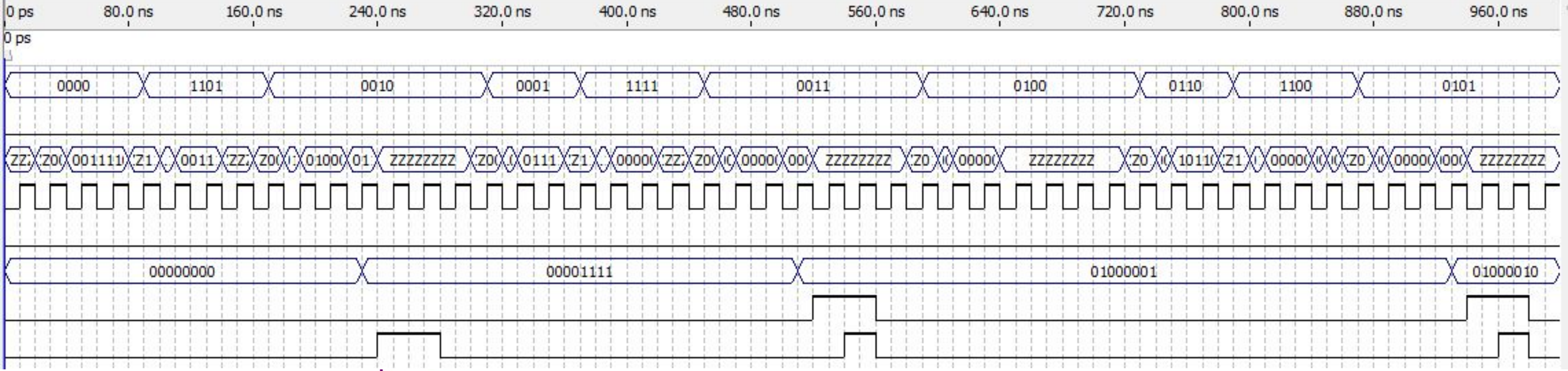


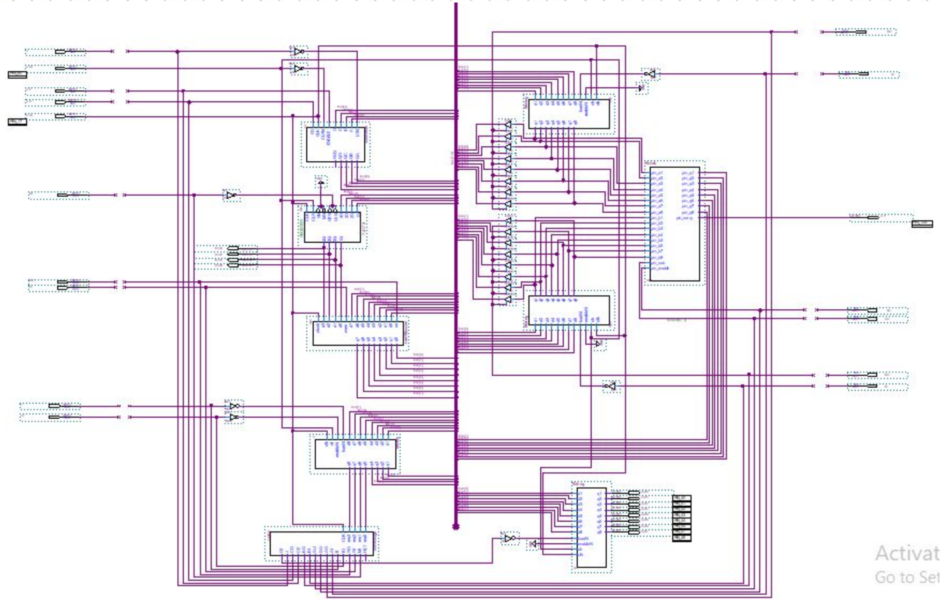
Display (1602A LCD)





```
00          : 1E;   %LDA 0xe%
01          : 50;   %DISPLAY_CFG%
02          : 1F;   %LDA 0xf%
03          : 40;   %OUT%
04          : 00;   %NOP%
05          : 2C;   %ADD 0xc%
06          : 40;   %OUT%
07          : 00;   %NOP%
08          : 2C;   %ADD 0xc%
09          : 40;   %OUT%
0A          : 30;   %HLT%
0B          : F0;
0C          : 01;
0D          : F0;
0E          : 0F;
0F          : 41;
```


0 ps
80.0 ns
160.0 ns
240.0 ns
320.0 ns
400.0 ns
480.0 ns
560.0 ns
640.0 ns
720.0 ns
800.0 ns
880.0 ns
960.0 ns
0000
1101
0010
0001
1111
0011
0100
0110
1100
0101
ZZZZZZZZ
00000000
00001111
01000001
01000010



Thank you!

Q>