Design Contest 2014
Mentor Graphics Egypt
Higher Education Program

Call for Participation

Mentor Graphics Egypt and Mentronix are announcing the 8th annual design contest for students. The focus of this contest is to provide an opportunity for engineering students to showcase their technical talent and competency using Mentor Graphics tools.

Under Graduates students, are invited to participate by submitting project work. This year contest details are as follows:

Proposed Topic: LC3 Microcontroller
Enrollment deadline: 18th June 2014
Submission Date 10th September 2014
Shortlisted Projects Notification: 25th September 2014
Winners Notification: 05th October 2014
Registration: Fill in the registration form and send to HEP@mentronix.net.

Should you need any further information, please do not hesitate contacting us at support@mentronix.net

Our design contest is beginning. Get ready; the next winner may be you. Good Luck to all participants.

Project Submission, Evaluation and Judgment

1. Stage 1: Your final submission should include the following:

   - The software copy of all the projects required deliverables
   - A document describing:
     - Project Implementation method
     - Problems faced during project.
     - Any reference to design principles and theory where appropriate
     - Appropriate imagery to illustrate the design solution
     - Acknowledgement of partial or incomplete solutions

2. Stage 2: The 5 shortlisted Teams will be expected to host their presentation and discuss their approach, design method and solutions with the Technical committee

   - Each presentation should not have more than 15 slides.
   - Presentation must include: The proposed solution's name, team name, University/College affiliation, The perspective taken to address the Project/Design challenge, concise description of the proposed solution and Clear illustrations of key aspects of your proposed solution

3. Stage 3: 3 out of the 5 shortlisted Teams will be selected by the Technical committee

Software

Mentor Graphics EDA Software will be used for the different stages of the design.

1. HDL Designer for Design Entry
2. Questa for simulation and verification
3. Precision Synthesis for RTL synthesis

General and Enrollment Rules for competition

1. Project and thesis work should be done using only Mentor Graphics tools substantially.
2. For enrollment, group must consist of at least two, but no more than five students.
3. There is no limit to the number of teams that may compete from any given College/University.
4. Submissions are invited from all undergraduate students at all stages of their university careers
5. University should be a member in Mentor Graphics Higher Education Program

Awards

- Overall first Prize 4000 EGP
- Overall Second Prize 2000 EGP
- Overall Third Prize 1000 EGP

Project Title: LC3 Microcontroller

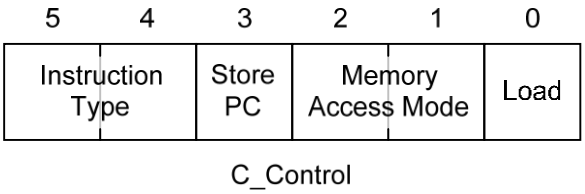
1. Introduction

The main purpose of this project is to let you start dealing with more complex designs, and become familiar with some of the elements used within a CPU.

2. LC3 structure

2.1 Controller

The Controller module is a finite state machine that controls the dataflow and therefore the execution of all the instructions in the microcontroller. The transitions are denoted by the edges. The condition of each transition is determined by the current state and/or input signal C_Control generated by the decoder module. The C_Control can be broken down further into 4 fields as follows.



The self-pointing edges are used for Project 2 to cope with the memory latency. Such looping transitions only occur when the **complete** signal is zero, which never happens in Project 1. State transitions occur and only occur at positive edges of the clock signal. When the **reset** signal is high, the next-state should be the “Fetch Instruction” state.

Notes:
In that project, you will use built in Memories in FPGA for instructions set.

VSR2 field (which in this schematic is the least significant 16 bits of the the **D_Data** signal to the **Execute** block) goes into the **MemAccess** block as the **M_Data** signal. All other signals are simple inputs and outputs.

2.2 Fetch

Fetch module is used to generate the program counter, which contains the address of the instruction to be fetched. The PC should be updated on the rising edge of the clock. Also, the PC should be updated only when the system is in the "Update PC" state, as determined by the Controller block. The signal **rd** should be high-impedance during the "Read Memory", "Write Memory", and "Indirect Address Read" states, because the MemAccess block will drive the shared memory bus during these cycles. In all other states, this signal should be high. **pc** is the memory address and should be high-impedance at the same times that **rd** is high-impedance. The first program instruction is located at the address 16'h3000. Therefore, **pc** should be set to 16'h3000 when **reset** is high. The block diagram of Fetch module is shown below.

2.3 Execute

Execute module performs the arithmetic and logical instructions, target PC computation, and memory address computation. The **E_Control** input is an aggregate of the **ALU Operation Sel**, **OP 2 Sel**, **PC Sel 1**, and **PC Sel 2**. The **D_Data** input is an aggregate of the **IR**, **VSR1** and **VSR2** signals.

2.4 MemAccess

The **MemAccess** block is the master of the shared memory bus during the *Read Memory*, *Write Memory*, and *Read Indirect Address* states. It should setup the memory bus lines as follows:

- Read Memory – **rd** should be 1 and **din** doesn't matter. **addr** should be set to either **M_Addr** or **dout**, depending on **M_Control**. **addr** should be set to **dout** in this state only if the opcode shows an LDI operation.
- Write Memory – **rd** should be 0 and **din** should be **M_Data**. **addr** should be set to either **M_Addr** or **dout**, depending on **M_Control**. **addr** should be set to **dout** in this state only if the opcode shows an STI operation.
- Read Indirect Address - **rd** should be 1 and **din** doesn't matter. **addr** should be set to **M_Addr**.

The **memout** signal should always pass the value of **dout** through to the **Writeback** block.

2.5 Writeback

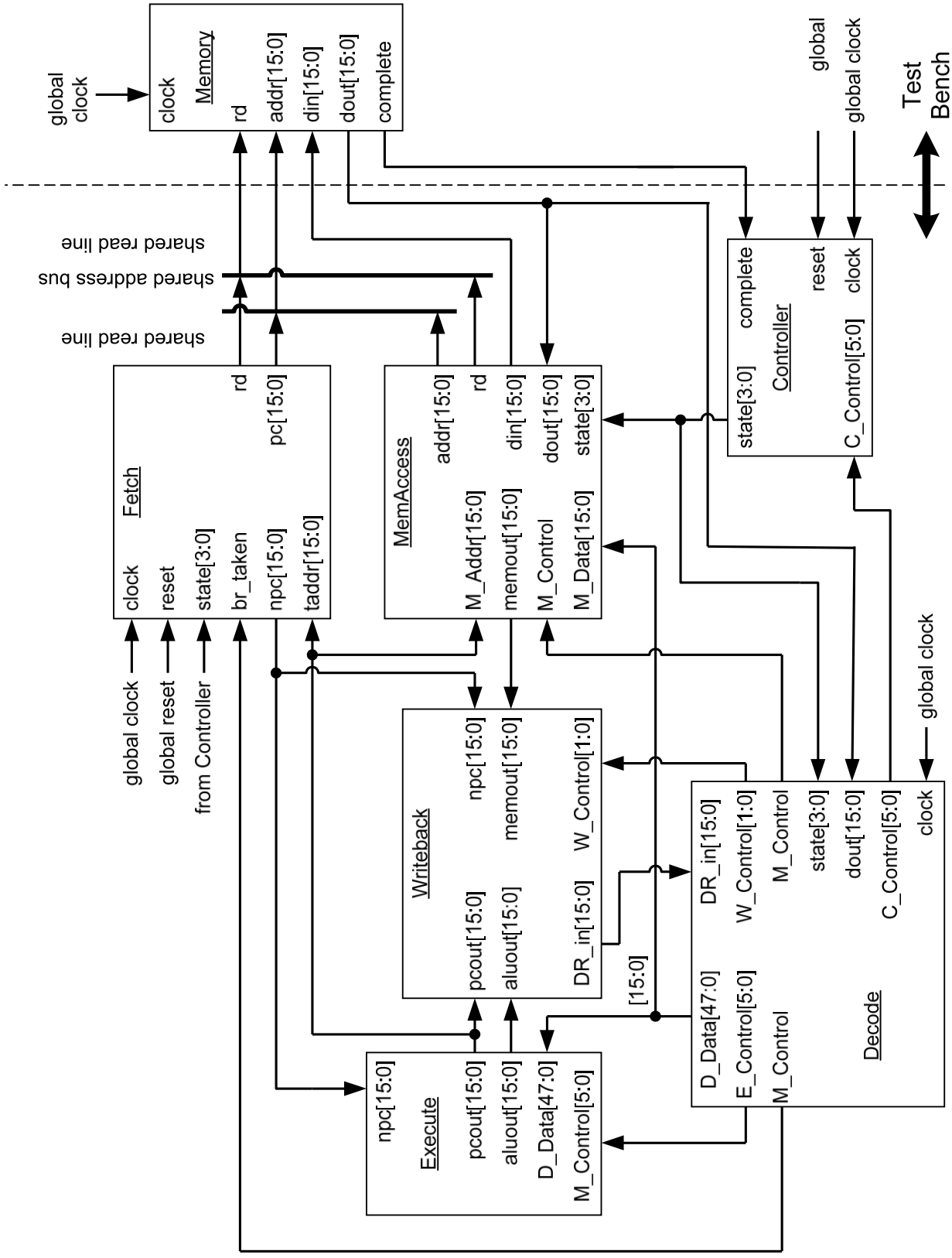
The **Writeback** block should set the **DR_in** lines to the value to be written into the register-file. This value is selected from the following four choices:

- **aluout** – The output of the ALU in the **Execute** block
- **pcout** – The computed memory address output of the **Execute** block
- **npc** – The next value of the program counter from the **Fetch** block
- **memout** – The value read from memory, from the **MemAccess** block

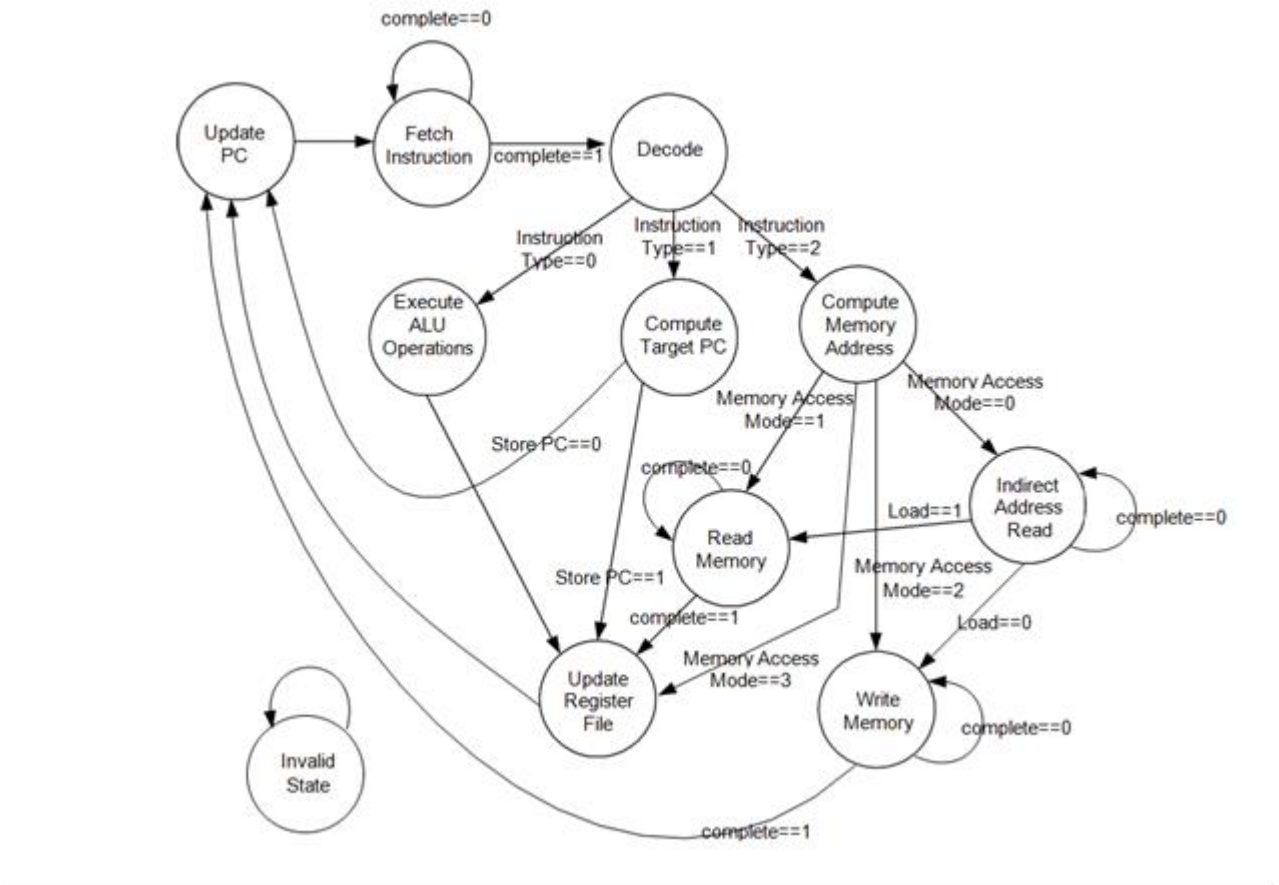
The **W_Control** signal will be used to select between these possibilities.

2.6 Decode

It contains an instruction register (IR) that stores the current instruction during the *Decode* state. It contains a program status register (PSR) that stores the status of the last value written to the register file (positive, negative, or zero) and is update only on the *Update Register File* state. Lastly, it contains a register file that can read two locations on one cycle and write to one location in the same cycle. However, the register file writes only during the *Update Register File* state. Based on the contents of IR and PSR, the decode block generates all of the control signals for the other blocks (C_Control, M_Control, W_Control, E_Control, and F_Control) as well as the source and destination addresses in the register file (sr1, sr2, and dr). Note, however, that the "instruction type" field of the C_Control signal must be valid during the *Decode* state and will therefore not be valid if this field is computed from the contents of IR. Therefore, the "instruction type" field is computed from the memory output, which makes it valid during the *Decode* state (but not necessarily the states after *Decode*).


Test Bench
global clock
global clock
global
global clock
Memory
clock
rd
addr[15:0]
din[15:0]
dout[15:0]
complete
shared read line
shared address bus
shared read line
Controller
state[3:0]
complete
reset
clock
C_Control[5:0]
Fetch
clock
reset
state[3:0]
br_taken
npc[15:0]
taddr[15:0]
rd
pc[15:0]
MemAccess
addr[15:0]
M_Addr[15:0]
memout[15:0]
M_Control
M_Data[15:0]
rd
din[15:0]
dout[15:0]
state[3:0]
global clock
global reset
from Controller
Writeback
npc[15:0]
pcout[15:0]
aluout[15:0]
memout[15:0]
DR_in[15:0]
W_Control[1:0]
Execute
npc[15:0]
pcout[15:0]
aluout[15:0]
D_Data[47:0]
M_Control[5:0]
[15:0]
Decode
D_Data[47:0]
E_Control[5:0]
M_Control
DR_in[15:0]
W_Control[1:0]
M_Control
state[3:0]
dout[15:0]
C_Control[5:0]
clock
global clock

LC3 System



Project Deliverables:

1. Finite-State machine drawing.
2. HDL codes.
3. Testbench and simulation results.
4. Synthesis Results (timing report and area report)