

LC3 Microcontroller

Mentor Graphics Design Contest 2014

Team members:

- | | |
|---------------------------------|--|
| 1- Ahmed Magdy Fouad El-Naggar. | amagdy505@ymail.com |
| 2- Abdelrahman Gaber Abu-Bakr. | eng.abdelrahman.gaber@gmail.com |
| 3- Hatem Mohamed El-Kharashy. | h13hatem@yahoo.com |
| 4- Khalid Essam El-Sayed. | k.e.elsayed@ieee.org |
| 5- Mohamed El-Sayed Khalifa. | eeekhalifa@gmail.com |

Contents:

- 1- Acknowledgments.
- 2- Introduction.
- 3- List of achievements.
 - 3.1- RTL Design.
 - 3.1.1- Required design.
 - 3.1.2- Pipelined design.
 - 3.2- Verification.
 - 3.2.1- Required design.
 - 3.2.2- Pipelined design.(proposed).
 - 3.3- Synthesize.
- 4- Implementation problems and solutions.
- 5- References.

1- Acknowledgments:

We thank all the people who spent countless hours helping us learn hardware design. We are specially thankful to Mr. Hesham Ragab from Mentrionix who provided us with all the information we needed.

We must express our gratitude to Mentor Graphics Egypt and Mentrionix for their efforts spent in organizing this contest. We have gained a lot of experience out of it.

2- Introduction:

Little Computer 3 known as LC3, is an educational computer. It's simple Instruction Set Architecture (ISA) makes it a good target practice for computer engineering students.

The LC-3 was developed by Yale N. Patt at the University of Texas at Austin and Sanjay J. Patel at the University of Illinois at Urbana-Champaign.

Courses based on the LC3 Microcontroller are offered in many computer engineering departments in the level of undergraduate students [5].

3- List of Achievements:

3.1 - RTL Design:

We made two different implementations: single instruction per data path controlled by a Finite State Machine (FSM), and a pipelined implementation without a controller module.

3.1.1- Required Design:

We made an RTL implementation of each block in the microcontroller using Verilog on Questa Sim, and Precision synthesis, each block was verified individually [1] [3].

Then we integrated the blocks to form the architecture shown in Figure 1. Every instruction in the microcontroller's Instruction Set Architecture was verified.

We portray our verification method and programs in Section 3.2.

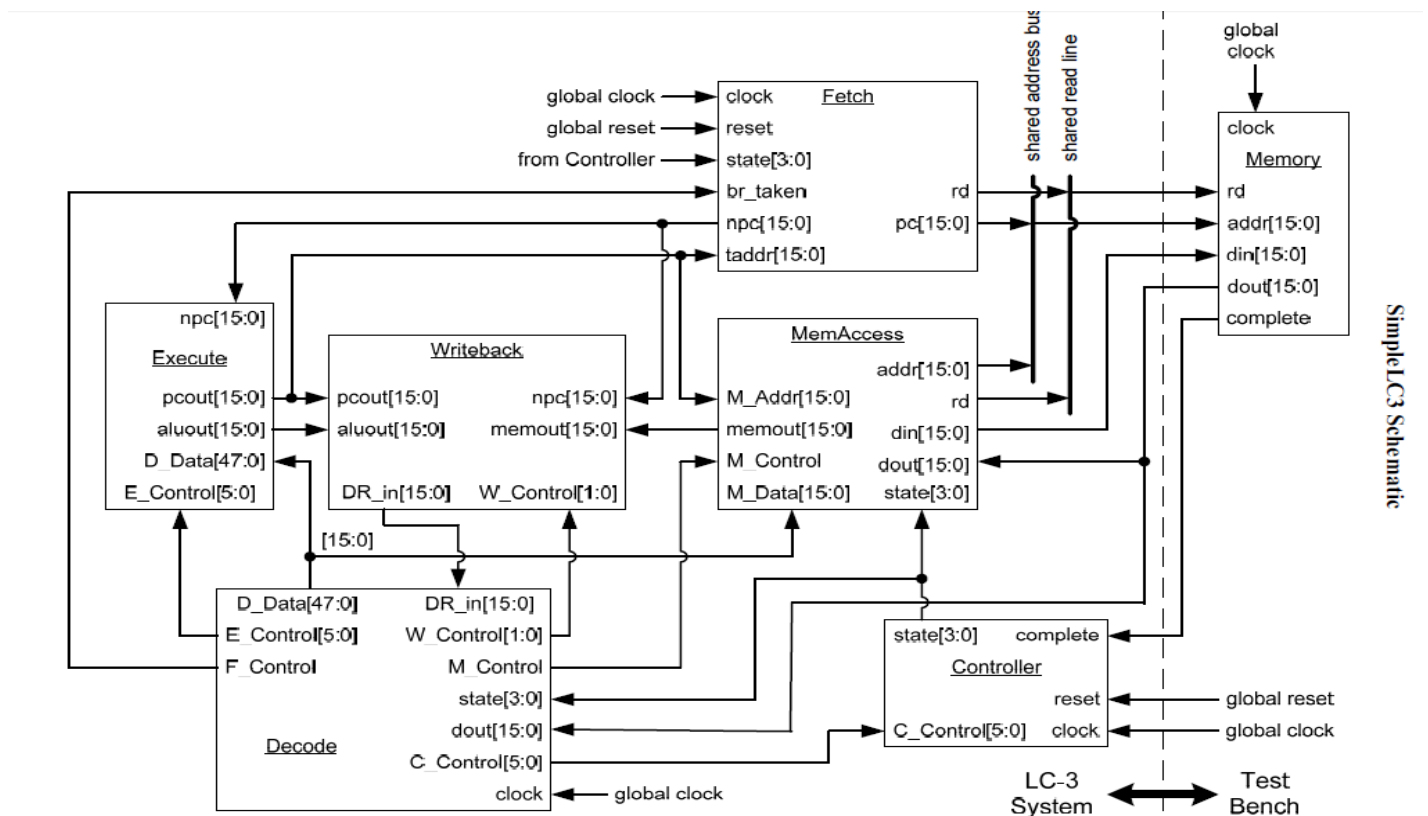


Fig.1: LC3 Microcontroller block diagram

3.1.2 -Pipelined Design:

In order to increase the throughput of our LC3-Microcontroller, we have implemented a pipelined architecture shown in Figure 3. We modified the architecture so it can work with pipeline. We have only tried addition instruction we didn't add hazard control units due to time limits.

Verification was done for the entire ISA, depicted in Section 3.2.

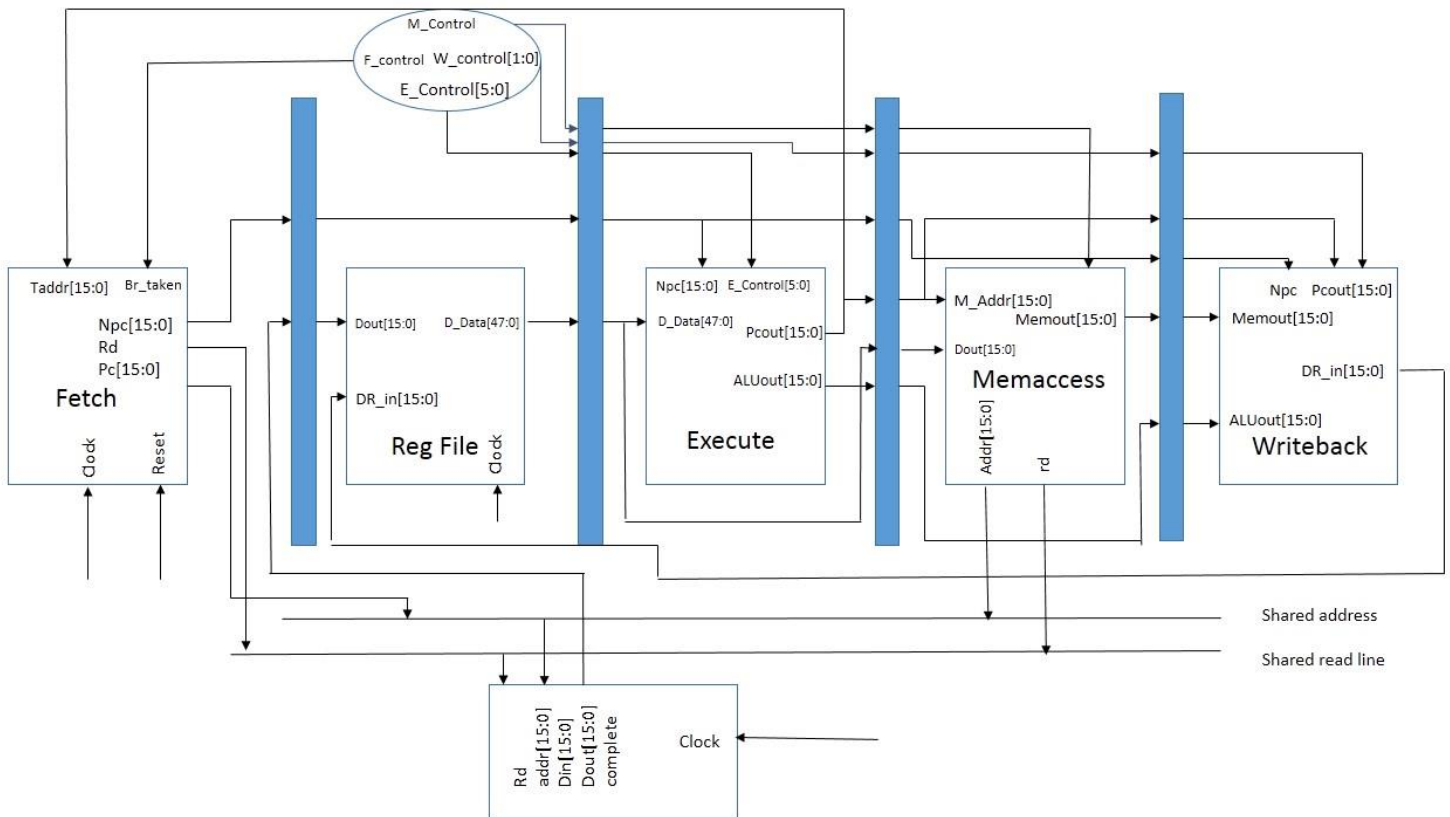


Fig 2: Implemented Pipelined architecture of the LC3-Microcontroller.

3.2- Verification:

3.1.1- Required design:

We used **Questa Advanced Simulator** that combines high performance and capacity simulation with unified advanced debug capabilities for the most complete native support of Verilog, SystemVerilog, VHDL, SystemC, PSL and UPF. The Questa Advanced Simulator is the core simulation and debug engine of the Questa Verification Platform; the comprehensive advanced verification platform capable of reducing the risk of validating complex FPGA and SoC designs.

Our verification work flow:

- 1- We made a testbench for each block to check their functionality.
- 2- We made a testbench for the whole LC3 microcontroller.

The testbench consists of :

- a) Memory that contains instruction and data.
 - b) LC3 module is the design under test.
 - c) Ideal core, a reference core described using behavioral Verilog which mimics the functionality of Microcontroller.
 - d) Check core, which monitors the output of both modules and checks if the instructions are correctly executed.
-
- 3- First we tried the three examples given in the orientation session.
 - 4- We made an additional program was tried to cover more test cases.

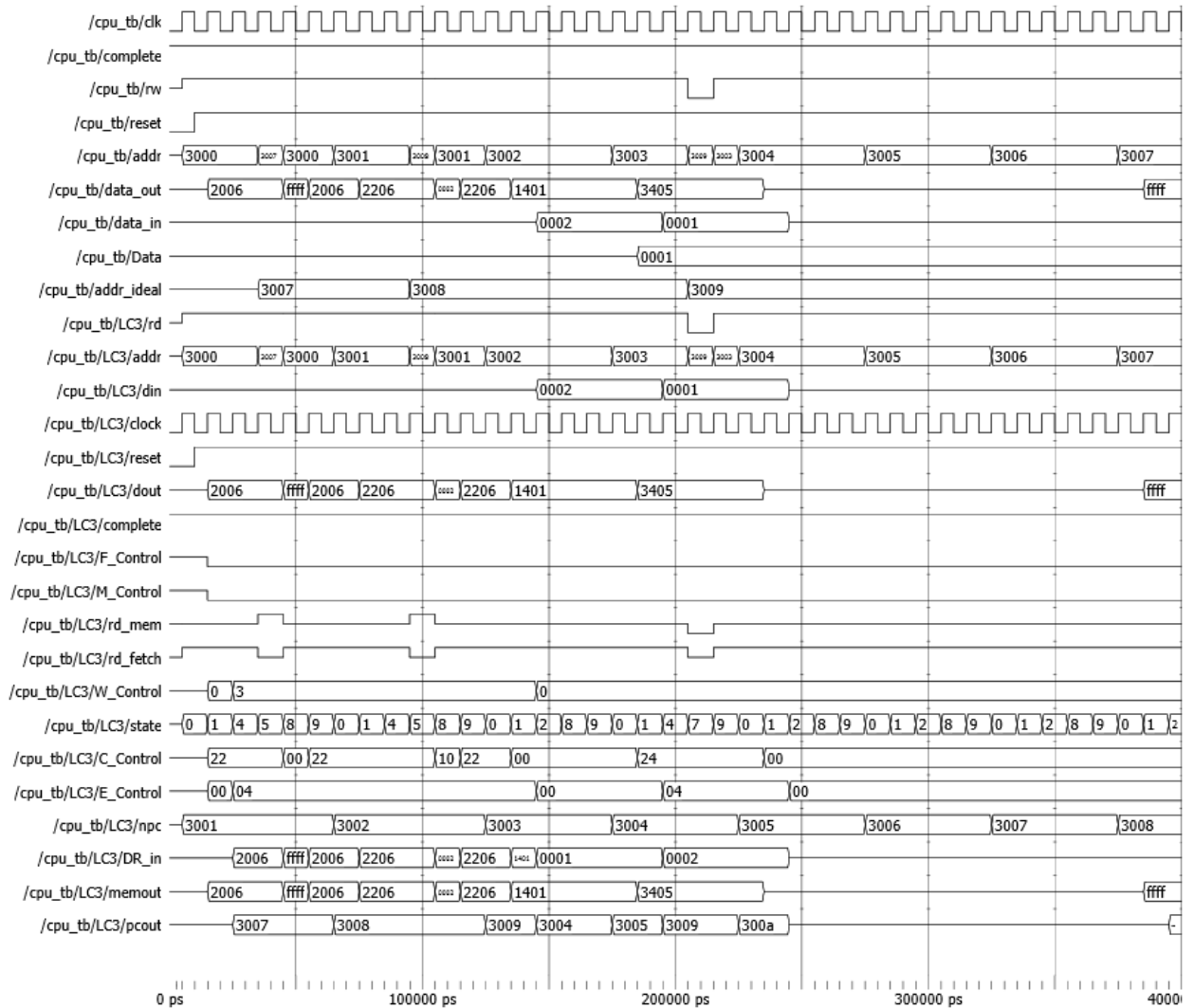
Example 1:

This example load 2 values from memory to registers then add the results to memory.

Binary Code:

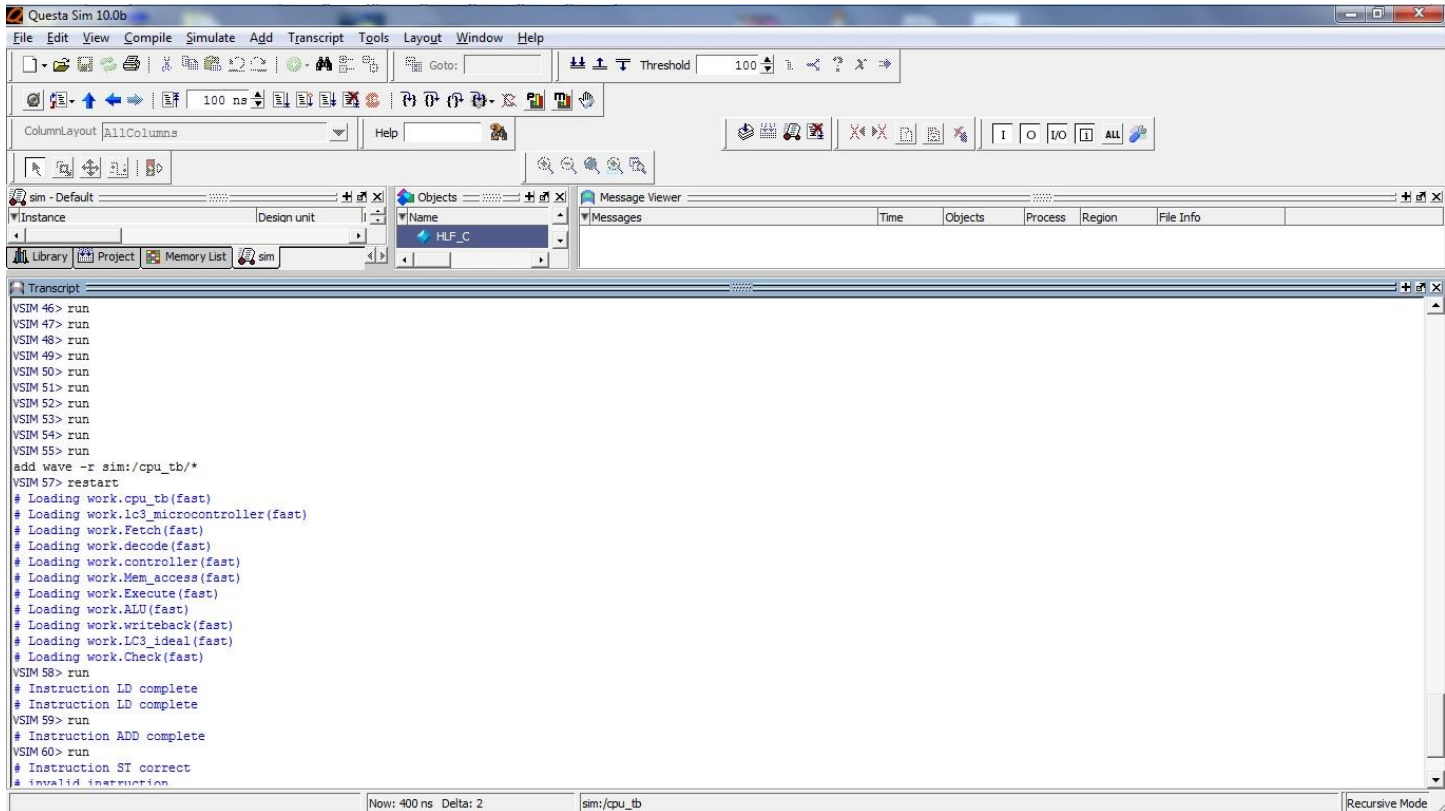
```
@3000 2006
      2206
      1401
      3405
@3007 FFFF
      2
```

Example 1 wave form.



Transcript of Ex. (1).

It shows that example was executed correctly.



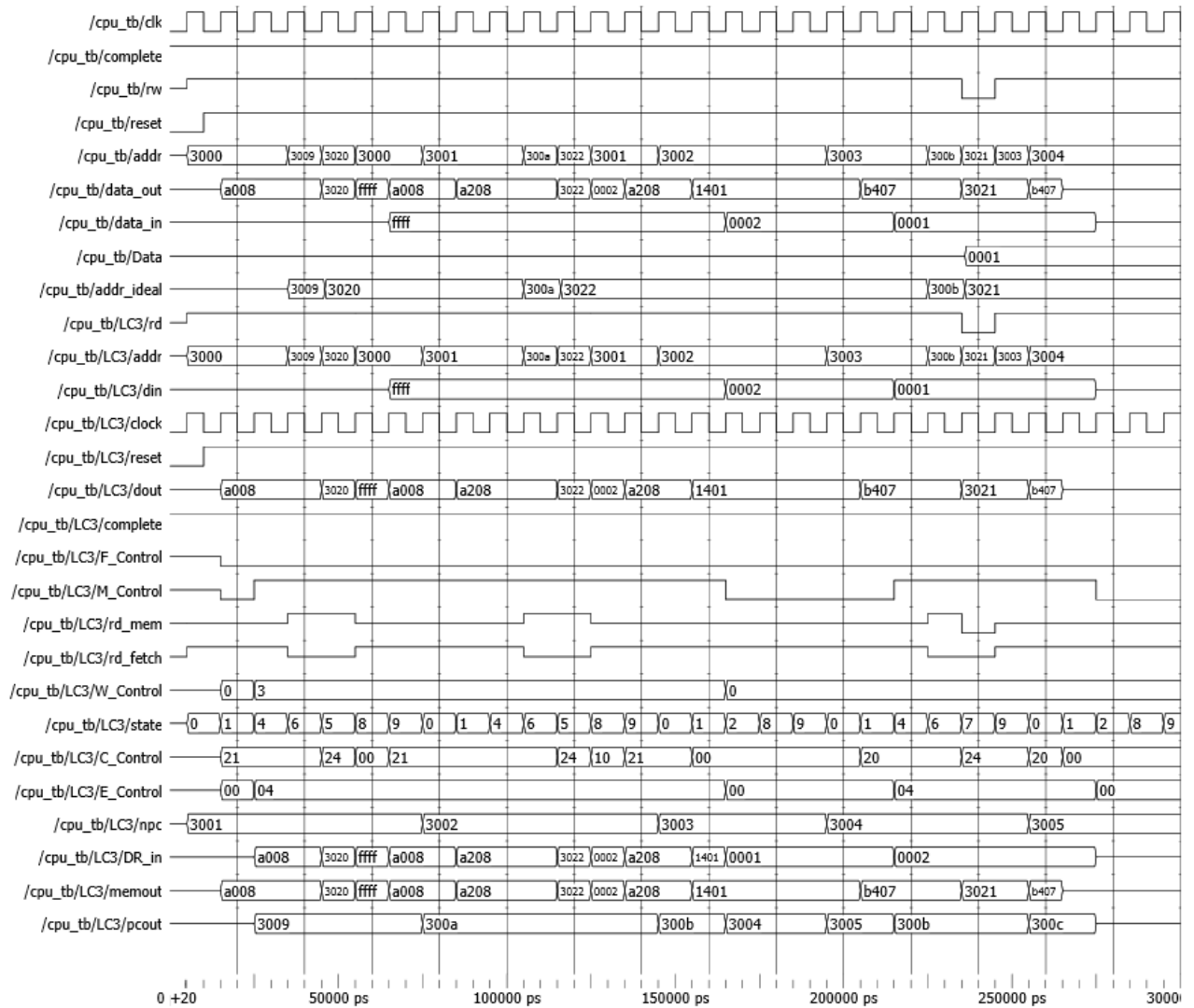
The screenshot displays the Questa Sim 10.0b software interface. The main window shows a simulation transcript for a project named 'sim'. The transcript is as follows:

```
VSIM 46> run
VSIM 47> run
VSIM 48> run
VSIM 49> run
VSIM 50> run
VSIM 51> run
VSIM 52> run
VSIM 53> run
VSIM 54> run
VSIM 55> run
add wave -r sim:/cpu_tb/*
VSIM 57> restart
# Loading work.cpu_tb(fast)
# Loading work.lc3_microcontroller(fast)
# Loading work.Fetch(fast)
# Loading work.decode(fast)
# Loading work.controller(fast)
# Loading work.Mem_access(fast)
# Loading work.Execute(fast)
# Loading work.ALU(fast)
# Loading work.writeback(fast)
# Loading work.LC3_ideal(fast)
# Loading work.Check(fast)
VSIM 58> run
# Instruction LD complete
# Instruction LD complete
VSIM 59> run
# Instruction ADD complete
VSIM 60> run
# Instruction ST correct
# invalid instruction
```

The interface also shows a 'Message Viewer' window with columns for Time, Objects, Process, Region, and File Info. The status bar at the bottom indicates 'Now: 400 ns Delta: 2' and 'sim:/cpu_tb'.

Example (2):

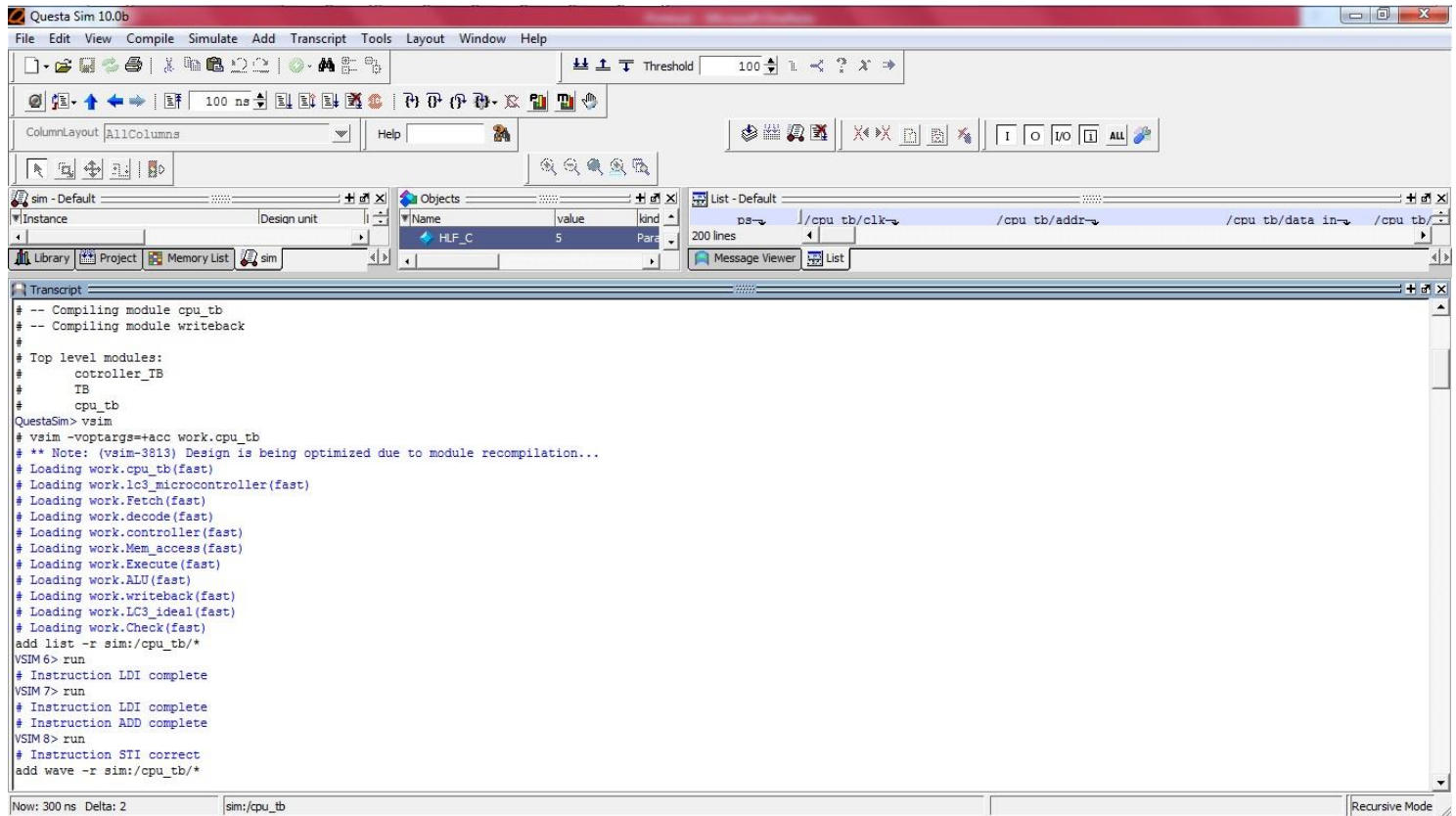
Example (2) waveform:



Entity:cpu_tb Architecture:fast Date: Thu Oct 02 05:05:21 ÎE Egypt Standard Time 2014 Row: 1 Page: 1

Transcript of Ex. (3).

It shows that example was executed correctly.



The screenshot displays the Questa Sim 10.0b software interface. The main window shows the transcript of a simulation. The transcript text is as follows:

```
-- Compiling module cpu_tb
-- Compiling module writeback

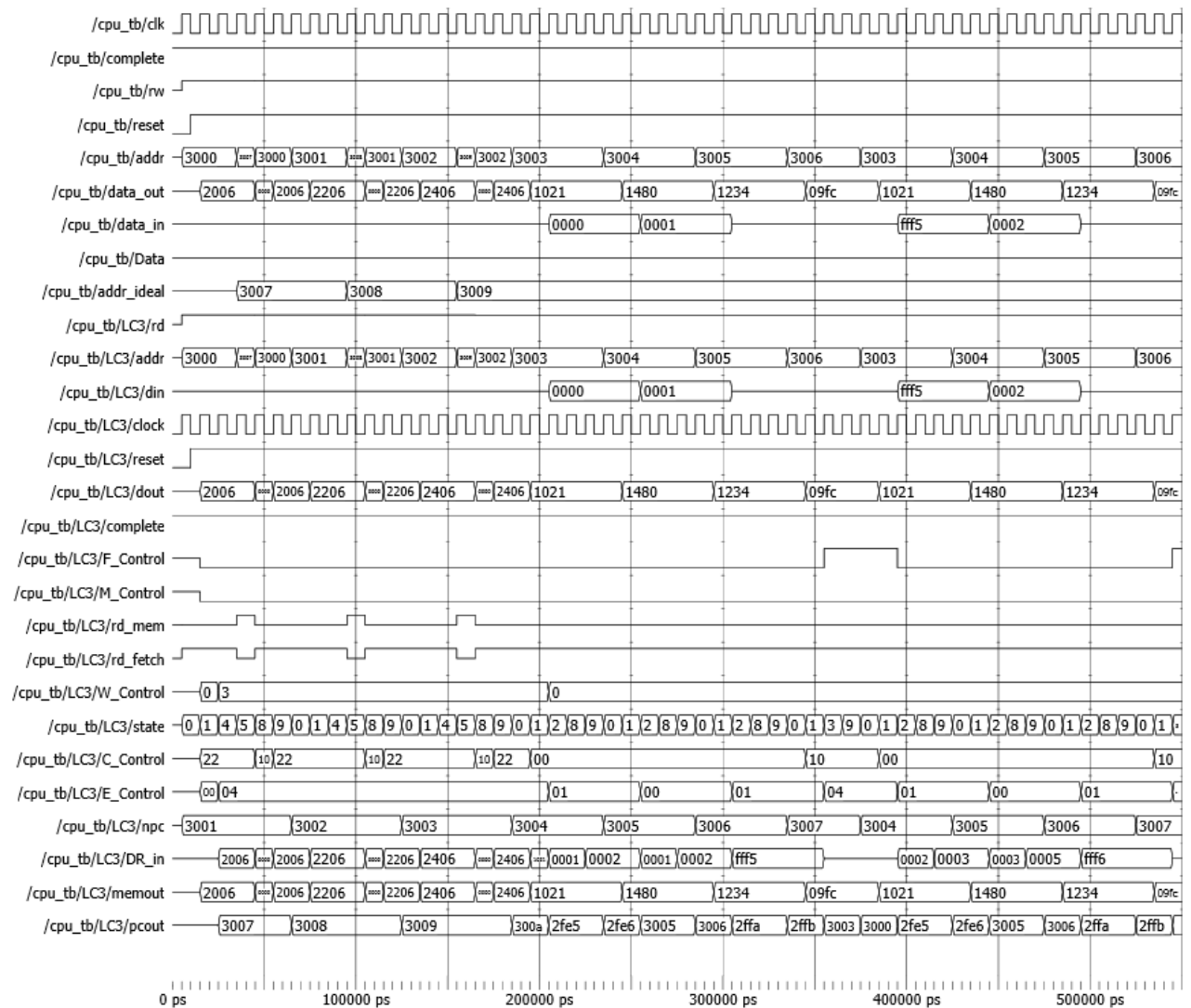
# Top level modules:
# cotroller_TB
# TB
# cpu_tb

QuestaSim> vsim
# vsim -voptargs=-acc work.cpu_tb
# ** Note: (vsim-3813) Design is being optimized due to module recompilation...
# Loading work.cpu_tb(fast)
# Loading work.lc3_microcontroller(fast)
# Loading work.Fetch(fast)
# Loading work.decode(fast)
# Loading work.controller(fast)
# Loading work.Mem_access(fast)
# Loading work.Execute(fast)
# Loading work.ALU(fast)
# Loading work.writeback(fast)
# Loading work.LC3_ideal(fast)
# Loading work.Check(fast)
add list -r sim:/cpu_tb/*
VSIM6> run
# Instruction LDI complete
VSIM7> run
# Instruction LDI complete
# Instruction ADD complete
VSIM8> run
# Instruction STI correct
add wave -r sim:/cpu_tb/*
```

The interface also shows a toolbar with various simulation controls, a list of objects (including HLF_C), and a list of signals (including /cpu_tb/clk, /cpu_tb/addr, /cpu_tb/data in, /cpu_tb/).

Example (3)

Example (3) waveform



Entity:cpu_tb Architecture:fast Date: Thu Oct 02 05:17:52 E Egypt Standard Time 2014 Row: 1 Page: 1

Transcript of Ex. (3).

It shows that example was executed correctly.

The screenshot displays the Questa Sim 10.0b software interface. The main window shows a transcript of simulation events, indicating that the execution was successful. The transcript includes commands like 'run' and status messages such as '# Instruction ADD complete' and '# Instruction BR complete'. The bottom status bar shows the current time as 2,700 ns and the simulation target as 'sim:/cpu_tb/CH0'.

Transcript:

```
# Instruction ADD complete
VSIM 33> run
# Instruction ADD complete
VSIM 34> run
# Instruction BR complete
# Instruction ADD complete
# Instruction ADD complete
VSIM 35> run
# Instruction ADD complete
# Instruction BR complete
# Instruction ADD complete
VSIM 36> run
# Instruction ADD complete
# Instruction ADD complete
VSIM 37> run
# Instruction BR complete
# Instruction ADD complete
VSIM 38> run
# Instruction ADD complete
# Instruction ADD complete
VSIM 39> run
# Instruction BR complete
VSIM 40> run
# Instruction BR complete
# invalid instruction
VSIM 41> run
VSIM 42> ]
```

Memory List:

Instance	Range	Depth	Width
/cpu_tb/mem	[65535:0]	65536	16
/cpu_tb/LC3/decod...	[0:7]	8	16

Objects:

Name	value	kind
clk	St0	Net
rw	St1	Net
write	001100000000	Net

Memory Data - /cpu_tb/LC3/decode_unit/reg_file - Default:

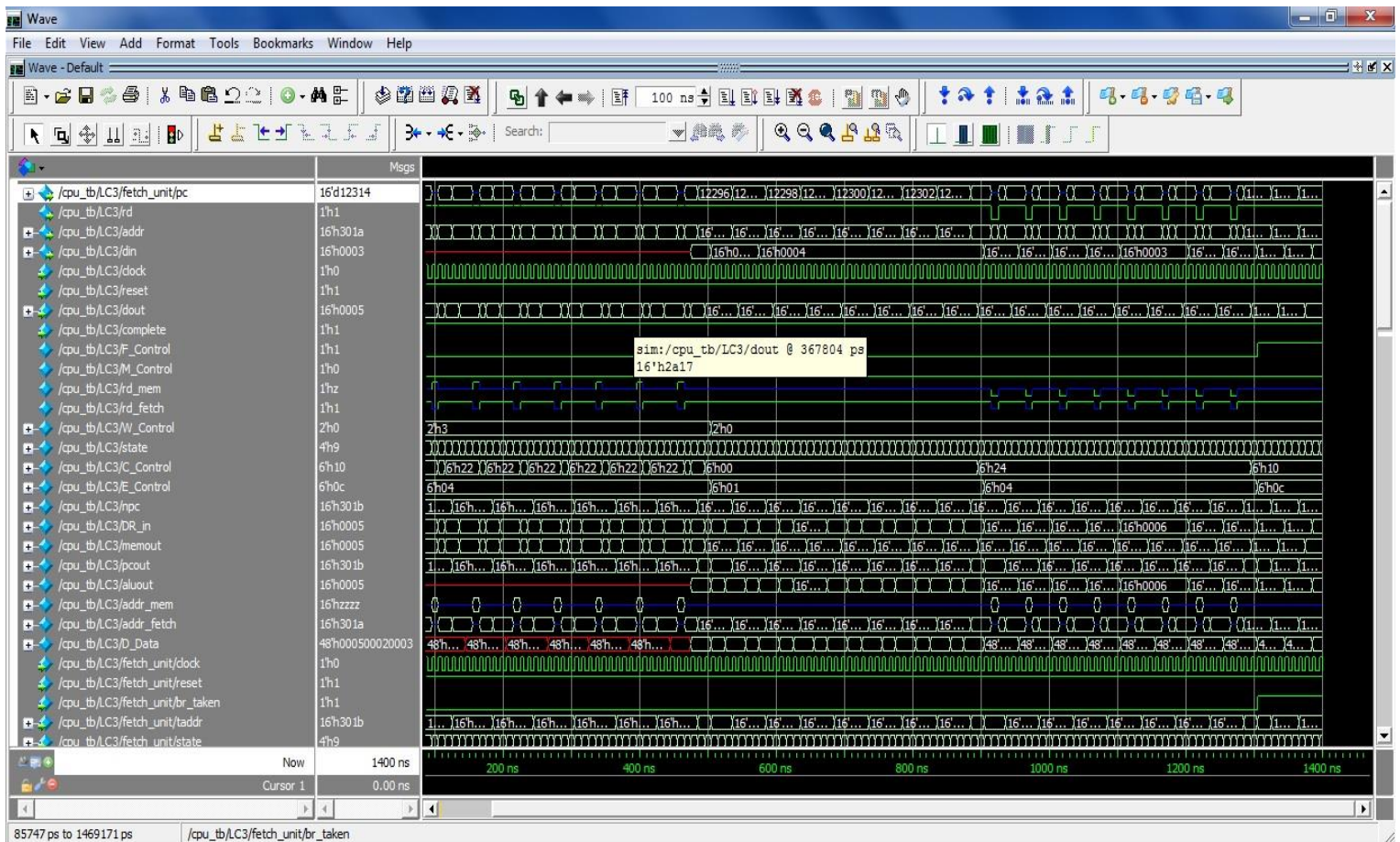
Address	Data
00000000	000c 0000 004e xxxx xxxx xxxx xxxx

Status Bar: Now: 2,700 ns Delta: 2 sim:/cpu_tb/CH0 Recursive Mode

Additional Example

This example loads 8 values to register then add one to each register then store the values again in memory

Waveform of this example:

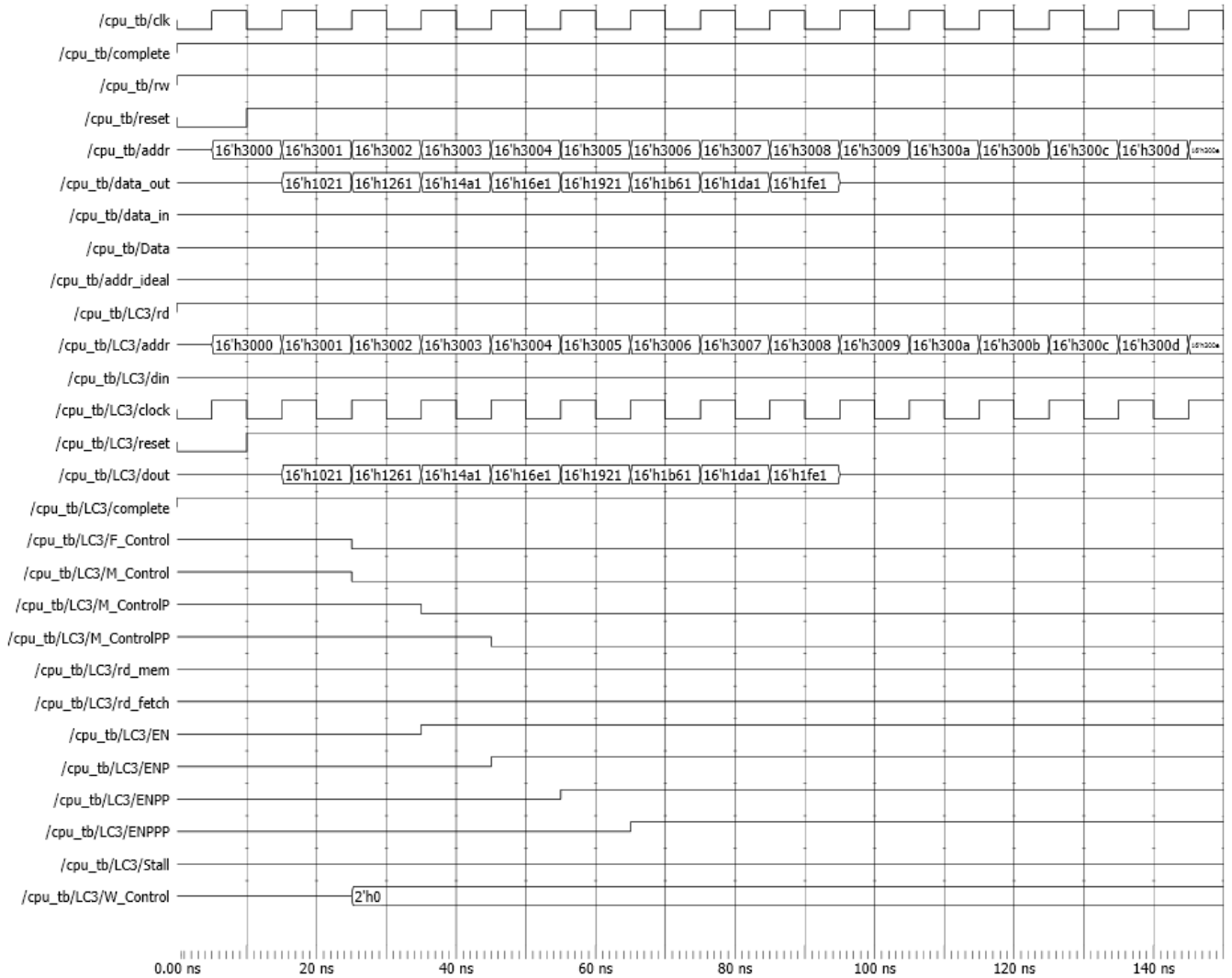


Transcript of the additional example:

```
add wave -r sim:/cpu_tb/LC3/*
VSI14> run
# Instruction LD complete
# Instruction LD complete
run
# Instruction LD complete
run
# Instruction LD complete
# Instruction LD complete
run
# Instruction LD complete
# Instruction LD complete
run
# Instruction LD complete
# Instruction ADD complete
run
# Instruction ADD complete
# Instruction ADD complete
run
# Instruction ADD complete
# Instruction ADD complete
run
# Instruction ADD complete
# Instruction ADD complete
run
# Instruction ADD complete
run
# Instruction ST correct
# Instruction ST correct
run
# Instruction ST correct
# Instruction ST correct
run
# Instruction ST correct
# Instruction ST correct
run
# Instruction ST correct
# Instruction ST correct
run
```


3.1.2- Pipelined design (Proposed):

It performs 5 additions in 13 clock cycles instead of 40 clock cycles as shown in the wave form.



Entity:cpu_tb Architecture: Date: Thu Oct 02 21:35:49 CAT 2014 Row: 1 Page: 1

3.3- Synthesize.

We used Mentor Graphics **Precision Synthesis** tool that offers high quality of results, industry-unique features, and integration across Mentor Graphics' FPGA Flow– the industry's most comprehensive FPGA vendor independent solution to synthesize our LC3-Microcontroller [4].

The following synthesis results are for the non-pipelined design. The pipelined design was not synthesizable. The block diagram in Figure 3 shows the synthesized design using the tool. We achieved a bandwidth of 28.6 Mbps.

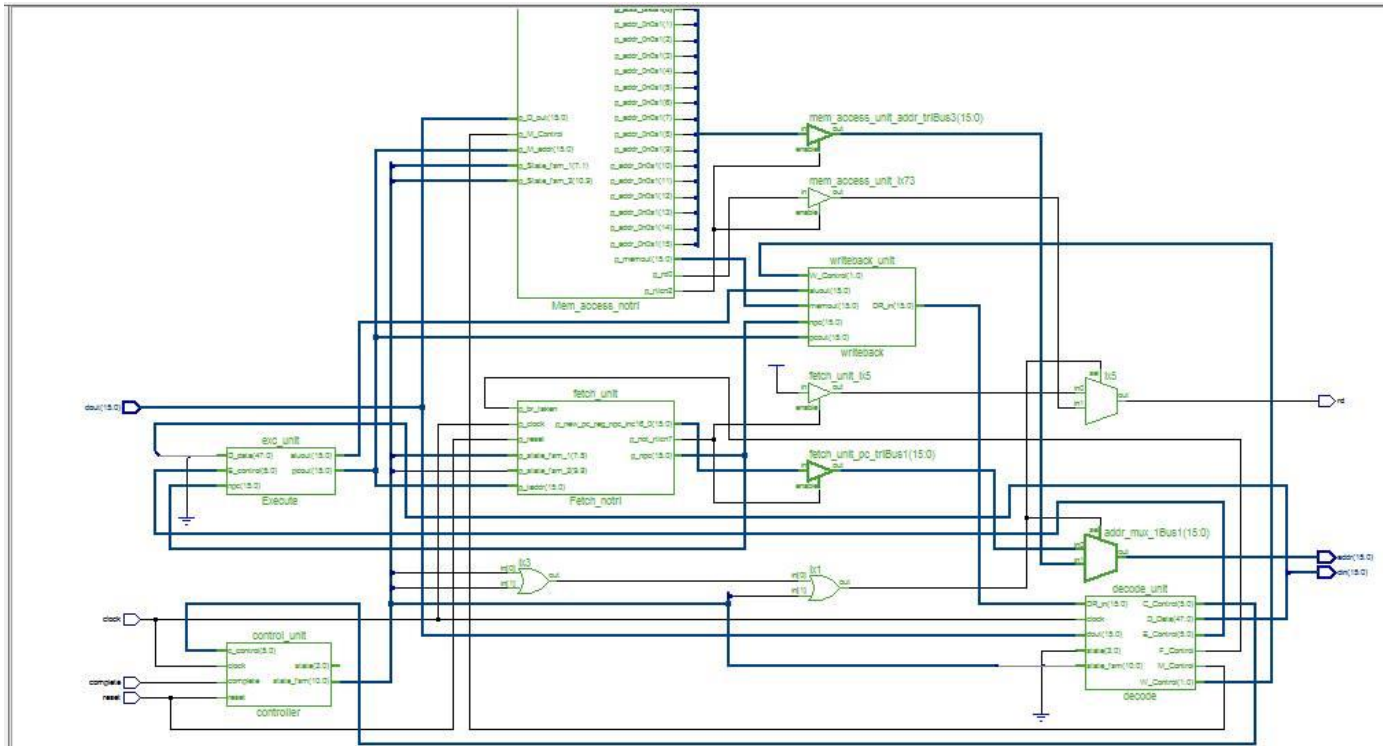


Fig. 3: Synthesis schematic of our LC3-Microcontroller.

We target Xilinx Spartan3A FPGA. The design passed timing closure. Timing and area summery are provided in Table 1, and 2 respectively. Detailed reports are attached with this document as PDF files in the deliverables folder.

a) Area Report:

Resource	Used	Available	Utilization
IOs	52	108	48.15%
Global Buffers	1	24	4.17%
LUTs	338	1408	24.01%
CLB Slices	169	704	24.01%
Dffs or Latches	42	1624	2.59%
Block RAMs	0	3	0.00%
Block Multipliers	1	1	33.33%

Table 1: Area utilization on Spartan3A.

b) Timing Report:

Initial edge separation	14.286
Source clock delay	-2.785
Dest clock delay	2.785
Edge separation	14.286
Setup constraint	0.301
Data required time	14.587
Data arrival time	- 14.262 (65.61% cell delay, 34.39% net delay)
Slack	0.325
Achieved period	13.961 (71.628MHz)

Table 2: Timing report summery in ns.

4- Implementation problems and solutions:

- a) It was the first time for us to deal with ISA [2].
- b) Integration.
 - Each block works properly, but when we integrated the whole architecture, we faced many functional errors. However we succeeded to debug all the errors.
 - We faced many synthesis errors and warnings; we traced them and succeeded to make the design synthesizable.
- c) We used to use another commercial synthesis tool in our previous projects, so we faced problems to learn the new tools.
- d) We faced many problems during synthesizing required design, like multiple drivers, unused latches but we succeeded to debug these errors.
 but in the pipelined core we faced many problems of multiple drivers, and succeeded to debug them except this problem that we can't find a solution till now:
 # Error: [9046]: Input of inverter or buffer is directly connected to its output --
 instance:ix5
 This problem appears in the pipelined core compilation using Precision Synthesis tool, but Questa Sim compiles the files correctly.

5- References:

- [1] Yale Patt, Sanjay Patel Introduction to Computing Systems- From bits and gates to C and beyond 2005.
- [2] Computer organization and design: the hardware/software interface/David A. Patterson, John L. Hennessy. — 5th ed.
- [3] J. Bhasker Verilog HDL Synthesis A Practical Primer 1998
- [4] Precision Synthesis Reference Manual, by Mentor Graphics.
- [5] ECE 406 Design of Complex Digital Systems, by NC State University.