EEE 3530
Spring 2023

Project #3: Pipelined RISC-V Implementation

Objectives:

To learn the basic structure and operation of the pipelined RISC-V processor

Due Dates:

Softcopy: June 14th, 23:59 PM (To LearnUS YONSEI)

** Late submission within 24 hours automatically deducts 20 percents of the point.*

Things to Submit:

1. Cover page with your name and Student ID
2. Schematic diagrams and codes printed from Quartus II (your design)
3. Simulation results and analysis (the waveform & answers to the questions)
4. Discussion

**** In the report paper, working out must be shown.***

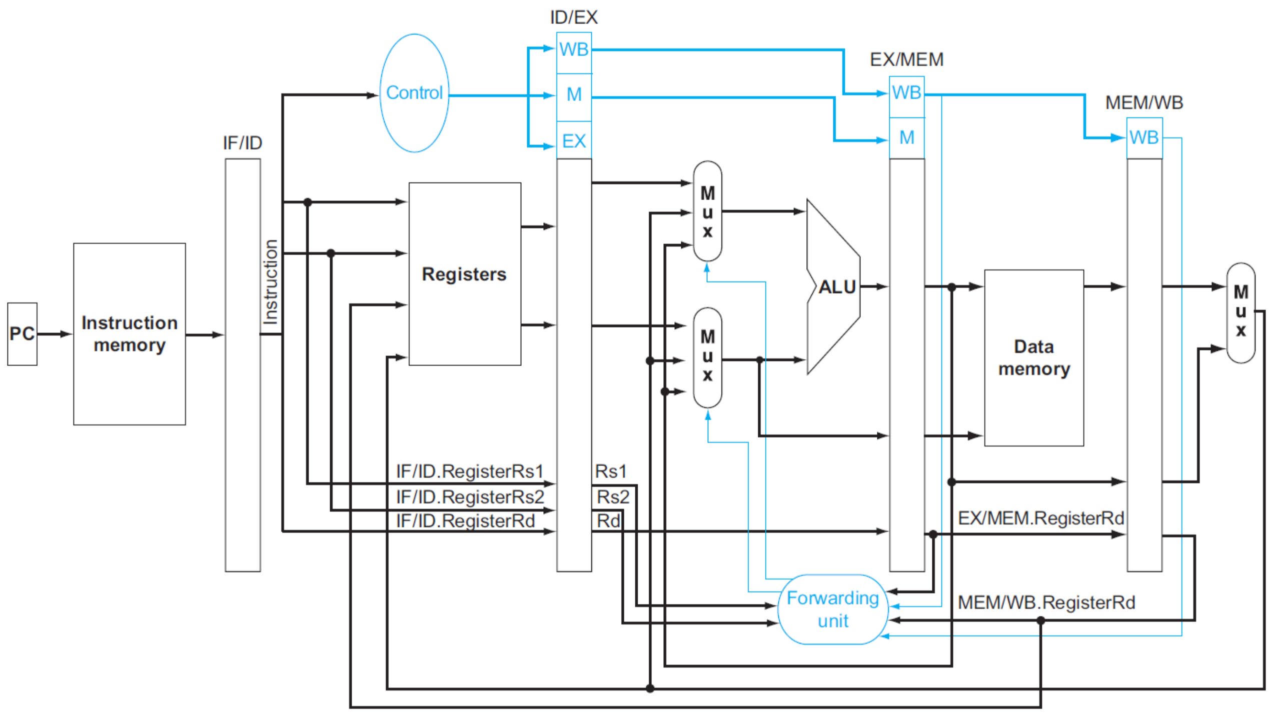
Professor: W. W. Ro

T. A.: Seokjin Go, Dongin Lee, and Kyeonghoon Lim

Feel free to contact us whenever you meet some difficulties or errors in the manual (but, spend some time and efforts to figure them out for yourself first)

64-Bit Pipelined RISC-V Microprocessor Design

In project #2, we implemented the single-cycle RISC-V processor. Now, we will implement the pipelined RISC-V processor with forwarding operation. The processor architecture is shown in the figure below. This figure is identical to Fig. 4.54 in the textbook.

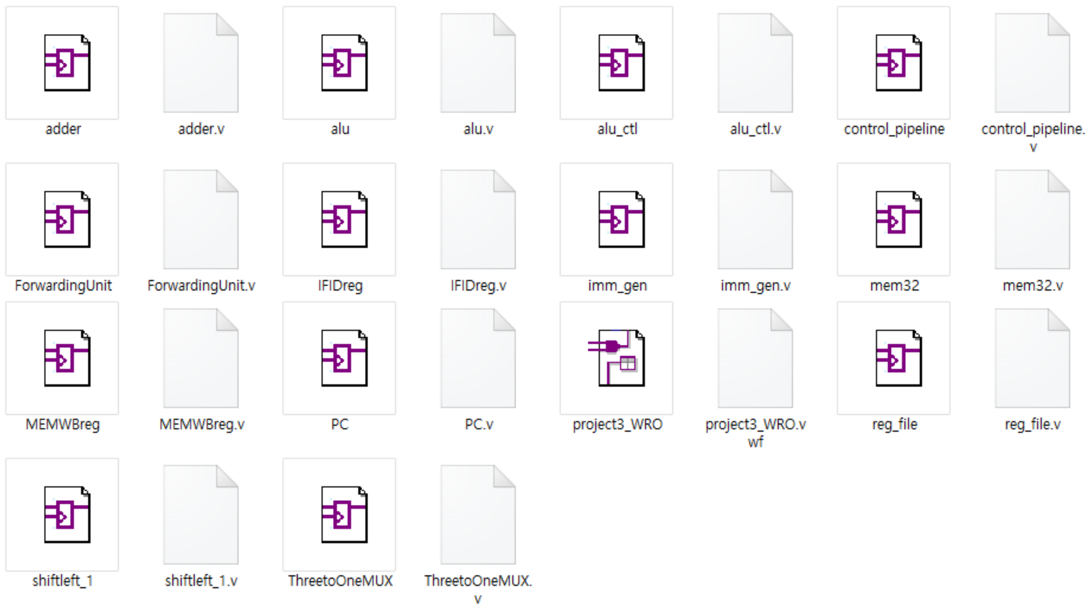


The requirement for the project is to implement and complete the datapath and control logic for pipelined architecture; major tasks include completion of the design and simulation of the 7 instructions: `add`, `sub`, `and`, `or`, `sd`, `ld`, and `beq`. The operation of each instruction follows the descriptions given in the textbook.

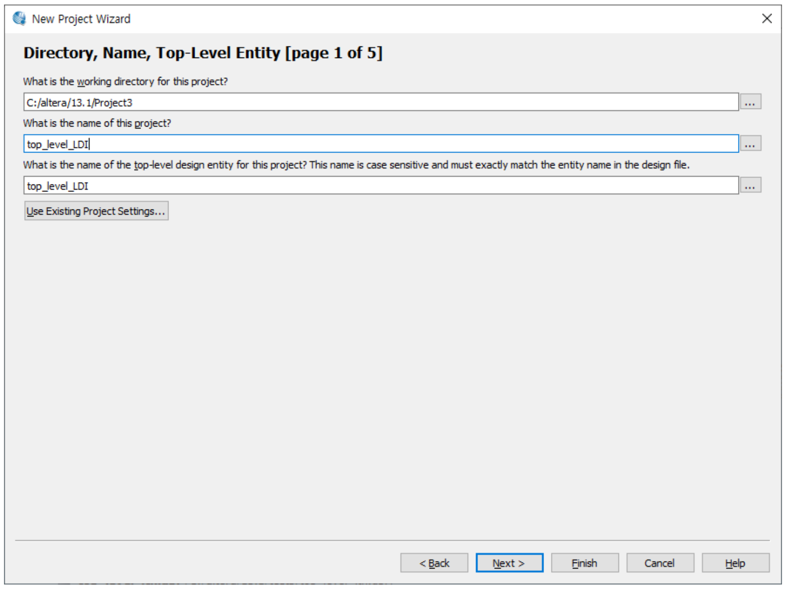
The design is implemented hierarchically with several block structures. Each block's operation and internal structure are identical to the descriptions in the textbook.

File download and setup procedure

1. Copy the following 28 files (You can download the files from the class website) to your working directory for the project.



2. Change the file "project3_WRO.bdf" to "top_level_*your intial*.bdf".
3. Start Quartus II and go to "File" → "New Project Wizard".
4. In the following window, find the working directory that you saved the 28 files in. Then, give the project name and the top-level design name as you named in step 2 (actually, **you can find the file and click that**).



Directory, Name, Top-Level Entity [page 1 of 5]

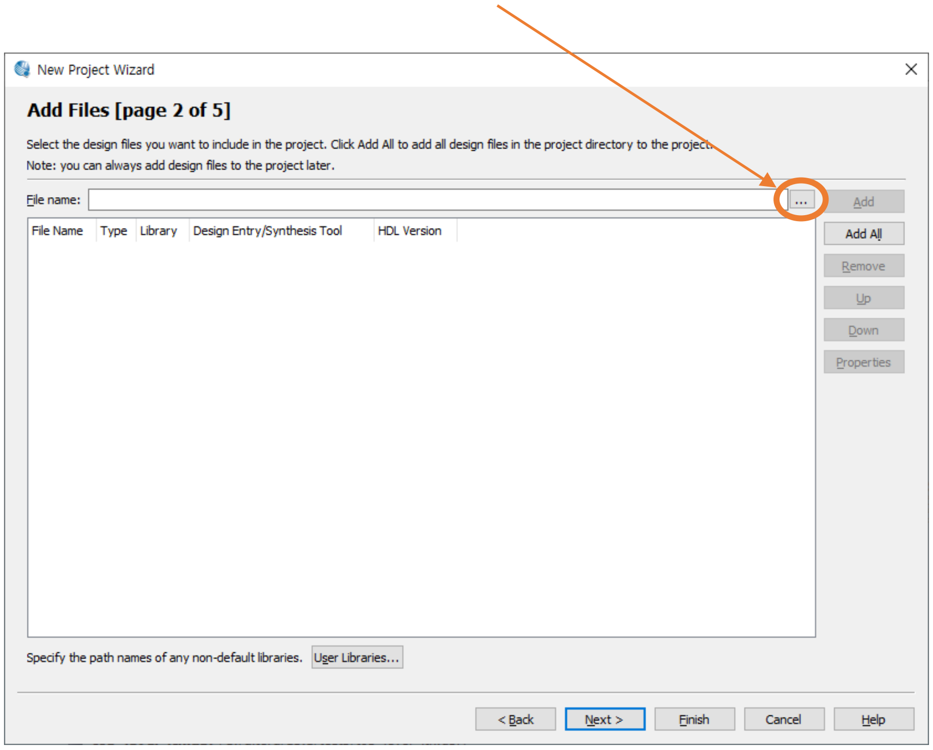
What is the working directory for this project?
C:/altera/13.1/Project3

What is the name of this project?
top_level_LDI

What is the name of the top-level design entity for this project? This name is case sensitive and must exactly match the entity name in the design file.
top_level_LDI

Use Existing Project Settings...

5. Then, in the following window, click here on this point. You need to locate your working directory.



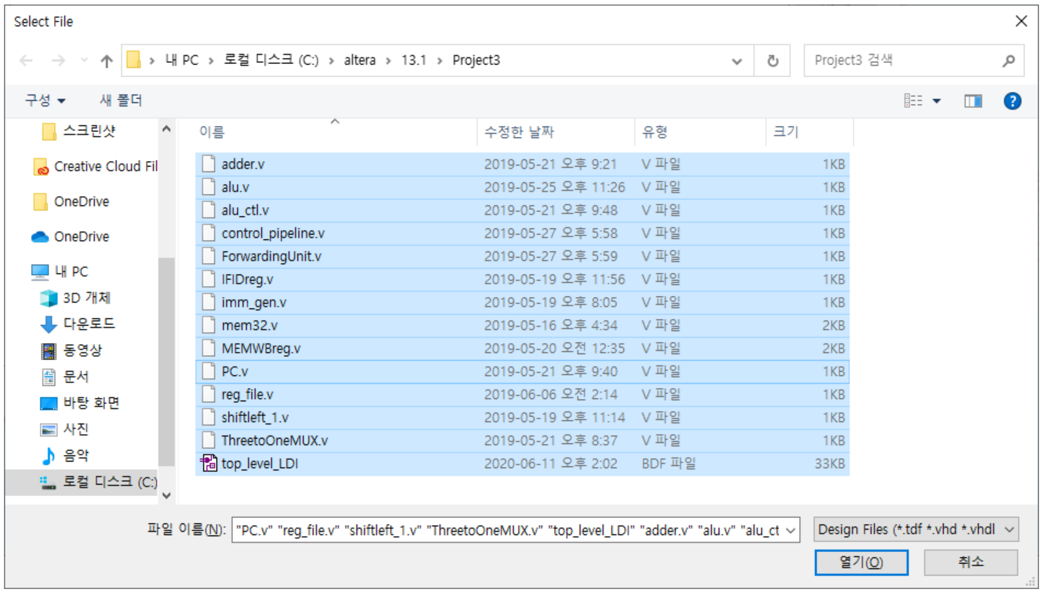
Add Files [page 2 of 5]

Select the design files you want to include in the project. Click Add All to add all design files in the project directory to the project.
Note: you can always add design files to the project later.

File Name	Type	Library	Design Entry/Synthesis Tool	HDL Version

Specify the path names of any non-default libraries.

6. In the following window, select the 14 files.

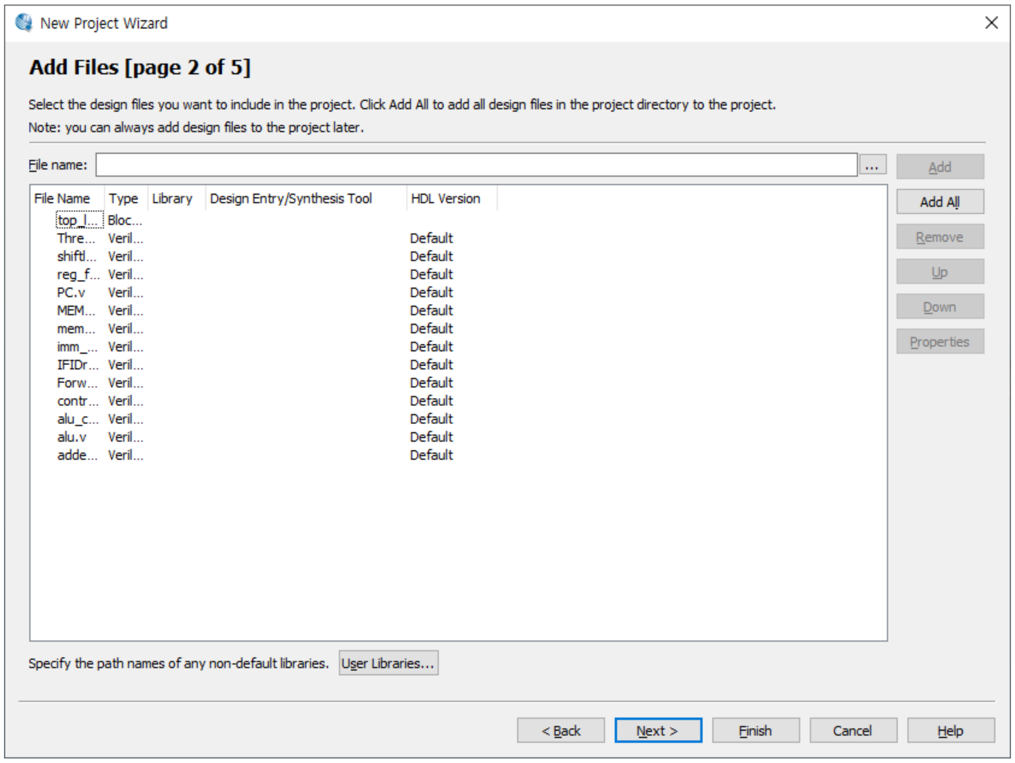


이름	수정한 날짜	유형	크기
adder.v	2019-05-21 오후 9:21	V 파일	1KB
alu.v	2019-05-25 오후 11:26	V 파일	1KB
alu_ctl.v	2019-05-21 오후 9:48	V 파일	1KB
control_pipeline.v	2019-05-27 오후 5:58	V 파일	1KB
ForwardingUnit.v	2019-05-27 오후 5:59	V 파일	1KB
IFIDreg.v	2019-05-19 오후 11:56	V 파일	1KB
imm_gen.v	2019-05-19 오후 8:05	V 파일	1KB
mem32.v	2019-05-16 오후 4:34	V 파일	2KB
MEMWBreg.v	2019-05-20 오전 12:35	V 파일	2KB
PC.v	2019-05-21 오후 9:40	V 파일	1KB
reg_file.v	2019-06-06 오전 2:14	V 파일	1KB
shiftleft_1.v	2019-05-19 오후 11:14	V 파일	1KB
ThreetoOneMUX.v	2019-05-21 오후 8:37	V 파일	1KB
top_level_LDI	2020-06-11 오후 2:02	BDF 파일	33KB

파일 이름(N): "PC.v" "reg_file.v" "shiftleft_1.v" "ThreetoOneMUX.v" "top_level_LDI" "adder.v" "alu.v" "alu_ct

Design Files (*.tdf *.vhd *.vhdl

7. Then, you will see the following box pop up. Just click "Next".



New Project Wizard

Add Files [page 2 of 5]

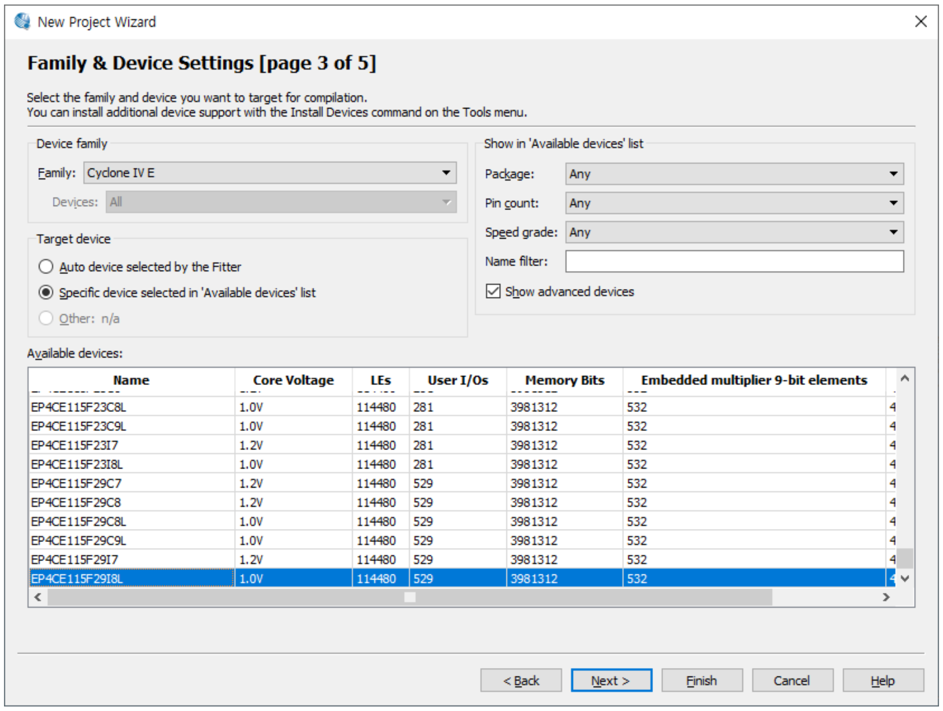
Select the design files you want to include in the project. Click Add All to add all design files in the project directory to the project.
Note: you can always add design files to the project later.

File name:

File Name	Type	Library	Design Entry/Synthesis Tool	HDL Version
top_l...	Bloc...			
Thre...	Veril...			Default
shiftl...	Veril...			Default
reg_f...	Veril...			Default
PC.v	Veril...			Default
MEM...	Veril...			Default
mem...	Veril...			Default
imm_...	Veril...			Default
IFIDr...	Veril...			Default
Forw...	Veril...			Default
contr...	Veril...			Default
alu_c...	Veril...			Default
alu.v	Veril...			Default
adde...	Veril...			Default

Specify the path names of any non-default libraries.

8. In the next pop-up window, choose 'Device Family: Cyclone IV E, Device: EP4CE115F29I8L'.



New Project Wizard

Family & Device Settings [page 3 of 5]

Select the family and device you want to target for compilation.
You can install additional device support with the Install Devices command on the Tools menu.

Device family
Family: Cyclone IV E
Devices: All

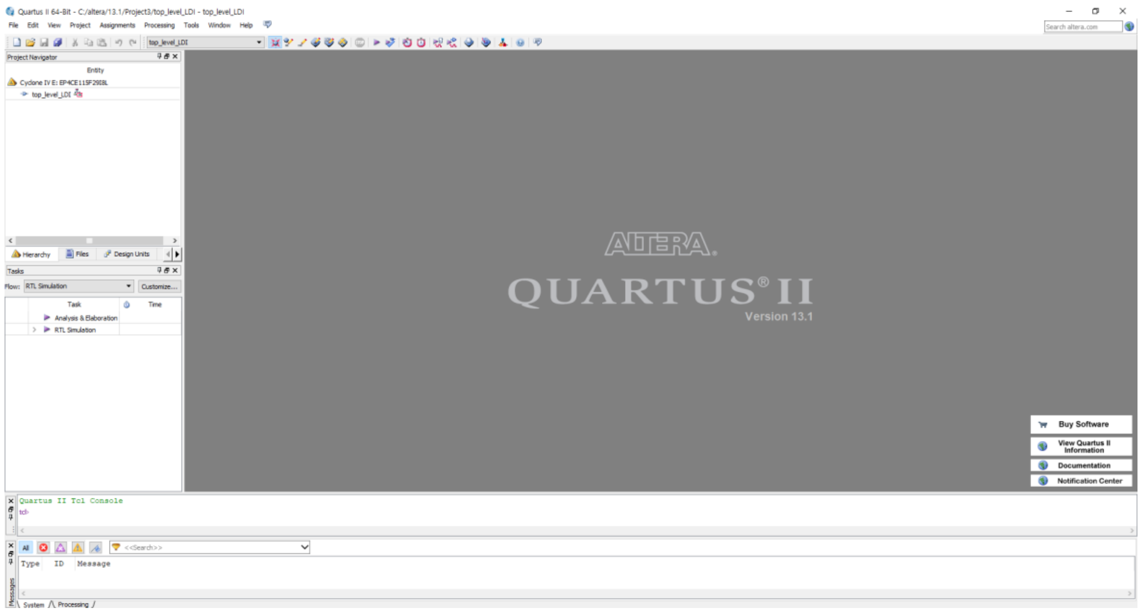
Target device
( ) Auto device selected by the Fitter
(●) Specific device selected in 'Available devices' list
( ) Other: n/a

Show in 'Available devices' list
Package: Any
Pin count: Any
Speed grade: Any
Name filter:
☑ Show advanced devices

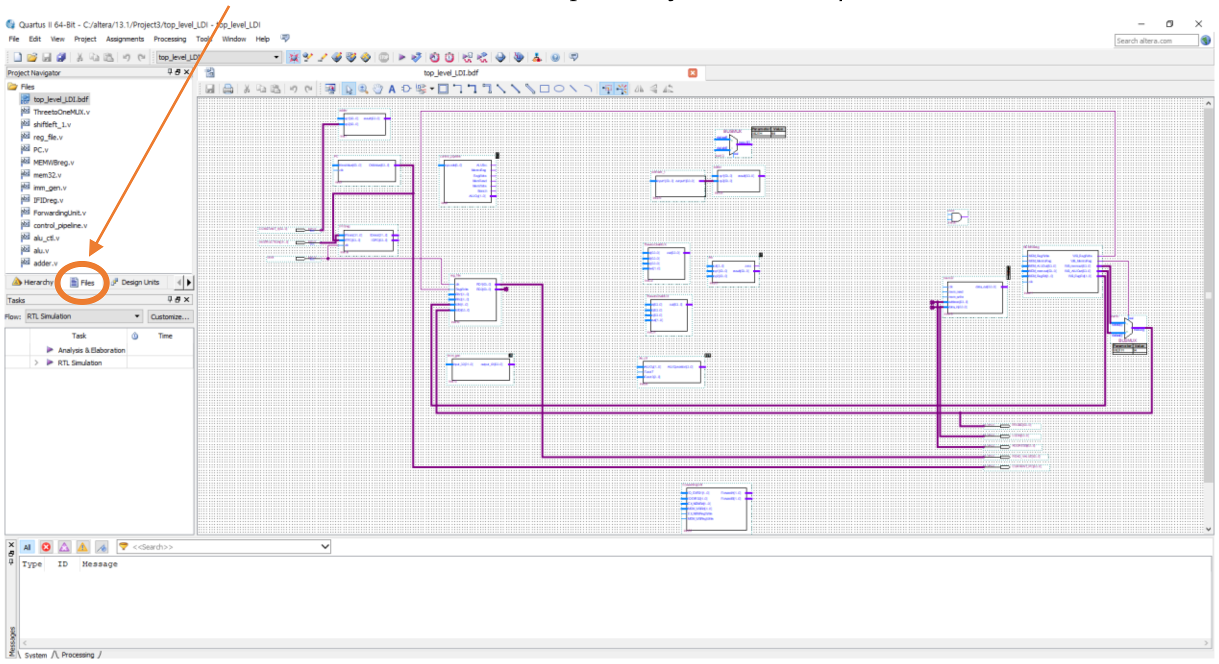
Available devices:

Name	Core Voltage	LEs	User I/Os	Memory Bits	Embedded multiplier 9-bit elements	
EP4CE115F23C8L	1.0V	114480	281	3981312	532	4
EP4CE115F23C9L	1.0V	114480	281	3981312	532	4
EP4CE115F23I7	1.2V	114480	281	3981312	532	4
EP4CE115F23I8L	1.0V	114480	281	3981312	532	4
EP4CE115F29C7	1.2V	114480	529	3981312	532	4
EP4CE115F29C8	1.2V	114480	529	3981312	532	4
EP4CE115F29C8L	1.0V	114480	529	3981312	532	4
EP4CE115F29C9L	1.0V	114480	529	3981312	532	4
EP4CE115F29I7	1.2V	114480	529	3981312	532	4
EP4CE115F29I8L	1.0V	114480	529	3981312	532	4

9. Just select the “Next” buttons until you get to the following window.

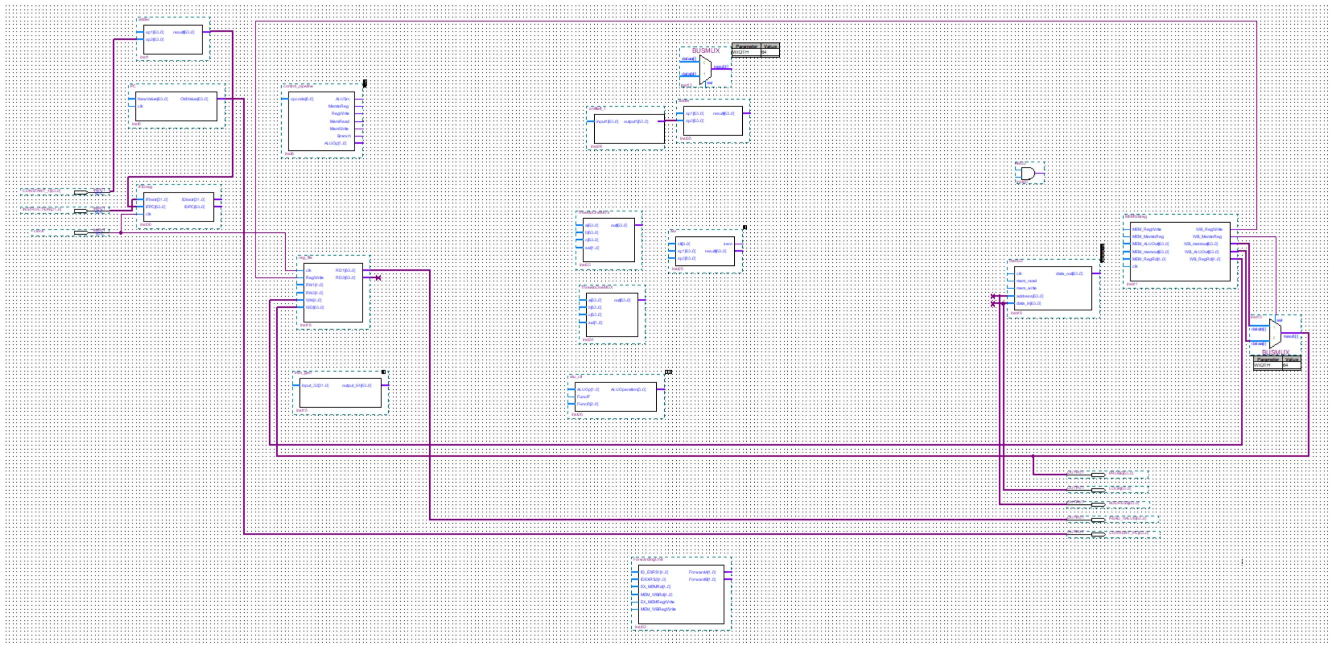


10. Now, click the “Files” and double-click the “*top_level_your initial.bdf*” file.



11. Now the following diagram is an incomplete version of the pipelined RISC-V design.

    (<u>You should connect the disconnected parts</u>)



* The top-level file contains a special part (e.g., 5 output buses) for the simulation purpose, which is not included in the original design of the textbook.

- There are 5 output buses: `“PROBE”`, `“LOOK”`, `“ADDRESS”`, `“CURRENT_PC”` and `“READ_VALUE”`. Those five signals are inserted in the original design to probe each point of the circuit. Our simulation can show run-time signals for those five points.

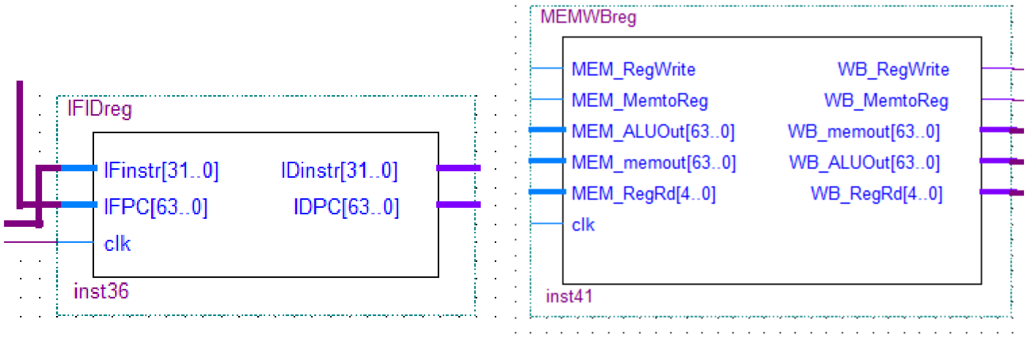
12. Double-click the blocks and fill in the blanks in the source code. You can change the design of the existing modules for the correct operations.

    (1) `control_pipeline.v`

    (2) `ForwardingUnit.v`

13. You should make some pipeline registers to store the necessary values in ID/EX and EX/MEM.

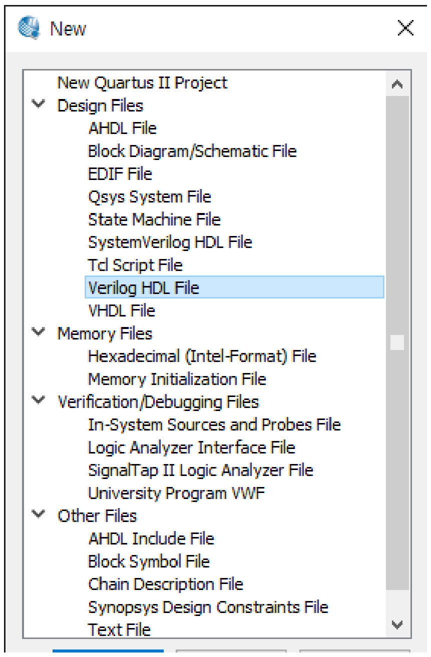
    The pipeline registers for IF/ID and MEM/WB are included in the source codes. You should make the new pipeline registers for the remaining stages based on the given registers.



    You should determine the input and output ports of new registers by yourself. Moreover, the operation should be modified according to the changed input and output.

14. You can include some other functional blocks to complete the design.

- If you verify that the given top module design doesn't have any blocks to implement pipelined RISC-V architecture, you may add some functional blocks to complete the design. You can see "**Create symbol files for this file**" by right-clicking on the file name in your project file list window after you complete and save the source code of a new module. When you save the source code, **be sure that the file name is the same as the module name**.



Simulation:

We provide a sequence of instructions to simulate. Also, we provide the list of initial values of the register file so that the given instruction sequence can be executed properly. The initial values are given in the table below. **Notice that you don't need to initialize the PC and the register file.**

Initial Values

Register No.	Initial Value	Register No.	Initial Value
x0	0	x4	8
x1	20	x5	30
x2	16	x6	2
x3	10	PC	0xA0000000

The instruction sequence is given in the table below. The instructions are given in the assembly codes, and you need to translate the instruction to the corresponding binary code for the simulation. These instructions are executed in the order shown below. You just show the correct output, not the real operation in RISC-V **(i.e., Do not jump to the target address in *beq* instructions, just show the output value)**. You should **refer to the last page of this document for OPCODE and FUNCT code**.

Instruction Sequence

Sequence	Instruction
1	`add  x3, x1, x2`
2	`and  x1, x2, x3`
3	`sub  x5, x3, x1`
4	`sd   x4, 12(x5)`
5	`or   x5, x6, x4`
6	`and  x1, x5, x4`
7	`beq  x1, x3, 0x440`
8	`ld   x6, 38(x5)`
9	`Your own example`
10	`Your own example`

Also, you should answer the following two questions;

(1) ***Compare the implementation without the forwarding unit and the one with the forwarding unit*. Compare and explain the differences between the two results. You can explain the question by pointing out the wrong operations when the forwarding unit is not used.**

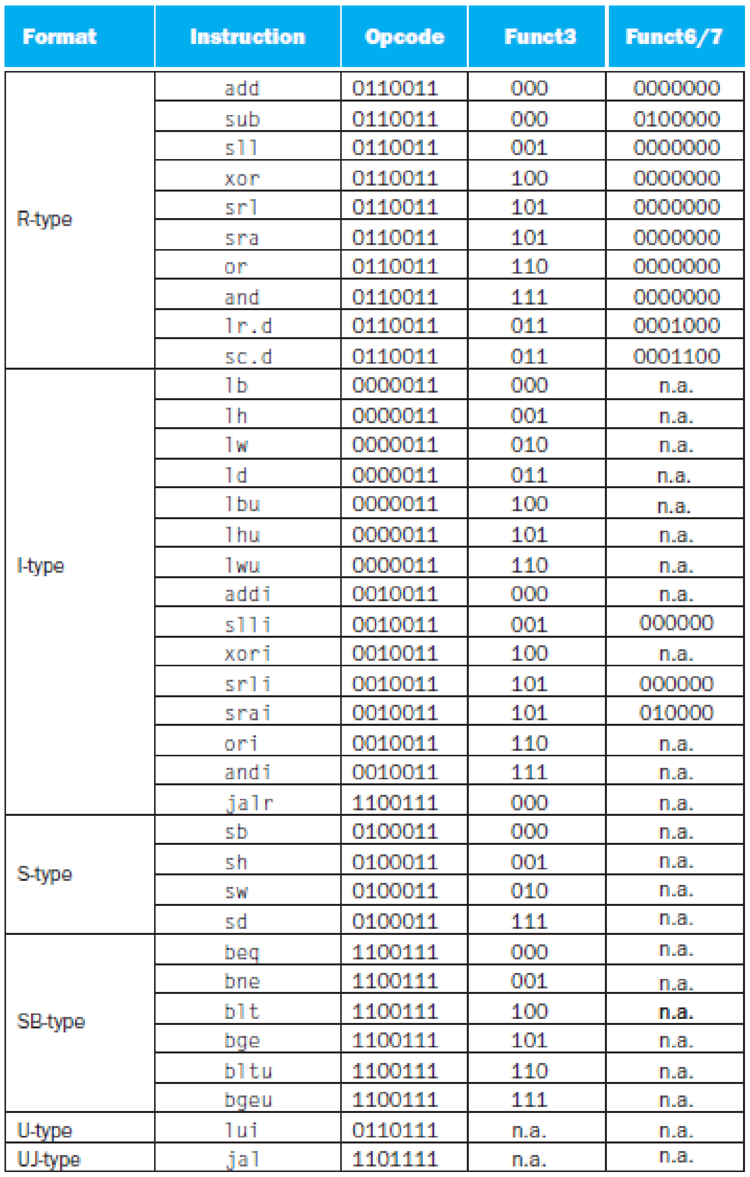
(2) ***Is the branch instruction (Instruction 7, beq) taken or not? If taken, how many instructions will be nop? Also, explain the appropriate method to resolve this problem(briefly)*. You can use any simulation results to support your answer. Note that you don't have to implement 'nop(flush)' operation.**

Things to turn in:

(1) Submit a soft copy of your report. The report should include the schematic design of your top module, source codes, a handful of discussions, and **simulation results (including the waveform).**

(2) Also, upload a compressed file of all the files given (whether modified or not, all of them) and added to the class web page. Please match the format ***Name(Korean)_StudentID.zip***.

(3) Analysis of the simulation results must be included to the report. You should completely describe and analyze the results and it should be written **in English.**



Format	Instruction	Opcode	Funct3	Funct6/7
R-type	`add`	0110011	000	0000000
	`sub`	0110011	000	0100000
	`sll`	0110011	001	0000000
	`xor`	0110011	100	0000000
	`srl`	0110011	101	0000000
	`sra`	0110011	101	0000000
	`or`	0110011	110	0000000
	`and`	0110011	111	0000000
	`lr.d`	0110011	011	0001000
	`sc.d`	0110011	011	0001100
I-type	`lb`	0000011	000	n.a.
	`lh`	0000011	001	n.a.
	`lw`	0000011	010	n.a.
	`ld`	0000011	011	n.a.
	`lbu`	0000011	100	n.a.
	`lhu`	0000011	101	n.a.
	`lwu`	0000011	110	n.a.
	`addi`	0010011	000	n.a.
	`slli`	0010011	001	000000
	`xori`	0010011	100	n.a.
	`srli`	0010011	101	000000
	`srai`	0010011	101	010000
	`ori`	0010011	110	n.a.
	`andi`	0010011	111	n.a.
	`jalr`	1100111	000	n.a.
S-type	`sb`	0100011	000	n.a.
	`sh`	0100011	001	n.a.
	`sw`	0100011	010	n.a.
	`sd`	0100011	111	n.a.
SB-type	`beq`	1100111	000	n.a.
	`bne`	1100111	001	n.a.
	`blt`	1100111	100	n.a.
	`bge`	1100111	101	n.a.
	`bltu`	1100111	110	n.a.
	`bgeu`	1100111	111	n.a.
U-type	`lui`	0110111	n.a.	n.a.
UJ-type	`jal`	1101111	n.a.	n.a.