Logic Design & Simulation Practicing

--Quartus II Schematic Tools for Sequential Circuit

Lecturer: TA 蔡承佑 (BL-430)

r10943014@ntu.edu.tw

交換電路與邏輯設計課程

Professors: 吳安宇 盧奕璋 江蕙如

臺灣大學

Switching Circuit & Logic Design

Outline

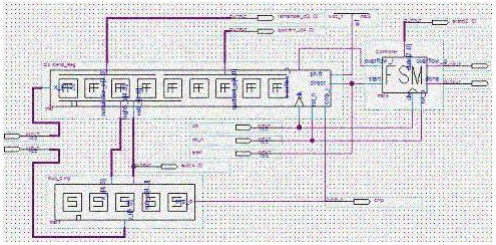
- Introduction
- Review
  - Timing Simulation
- FSM Design
- Topic example: Accumulator
- Advanced exercise
- Debugging
  - Common Problems
- Reference

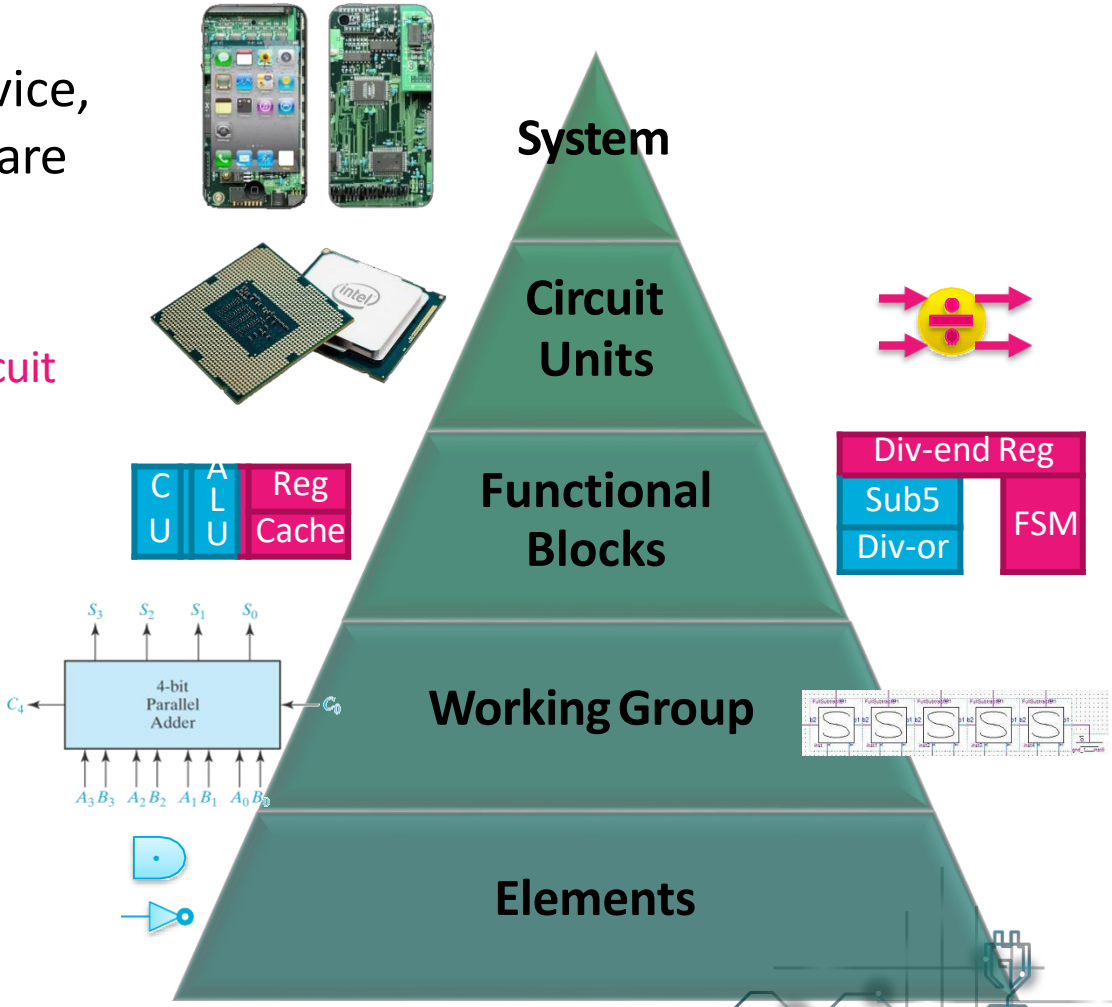
Introduction

Structure of Digital Circuit

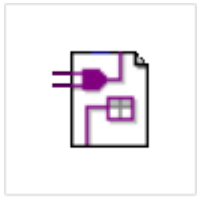
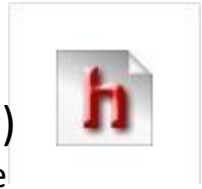
- To accomplish a targeted service, we usually divide our hardware into hierarchical parts
  - Divide and conquer
  - Usually contain sequential circuit

- Today we are going to build a slightly larger example





Review - .bdf .bsf .vwf

	Declaration	Implementation	Instance
Quartus Schematic Module	Module with I/O port info.  FullAdder1.	Ckt Design with ckt design  FullAdder1.bdf	Grab an instance block to another file ex. FullAdder4 to use it
C++ Object	Object Class (header file) with data, interface  EEmitation.h	Object Class (source file)  EEmitation.cpp	Use class included by header in another cpp ex. main.cpp

- .vwf：simulation waveform files

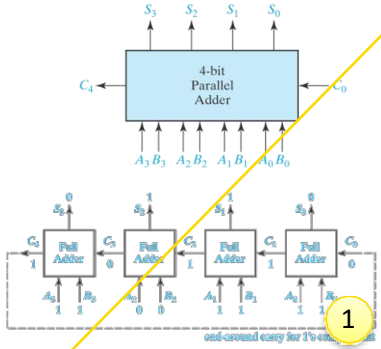
Review - Steps to Design & Verify Circuit

1. Launch Quartus II software
2. Build a Quartus project
3. Build block diagram files (bdf) and realize your circuit
4. Use block symbol files (bsf) to make modules if needed
5. Compile your design
6. Import a wave form file (vwf) (or build it yourself)
7. Run functional simulation
   – check whether your design is correct or not
   – If not, back to step #3

Review - Schematic Comb. Circuit Design

General Design Flow

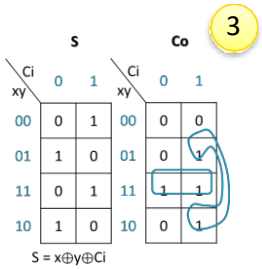
1. Divide system
2. Truth table
3. K-map
4. Boolean expression
5. Draw it first
6. Design with tools
7. Verify & debug



1

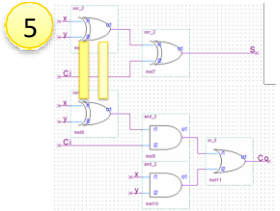
X	Y	C_{in}	C_{out}	Sum
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

2



3

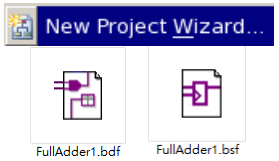
4
- S = (x⊕y)⊕Ci
- Co = (x⊕y)·Ci + x·y

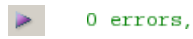


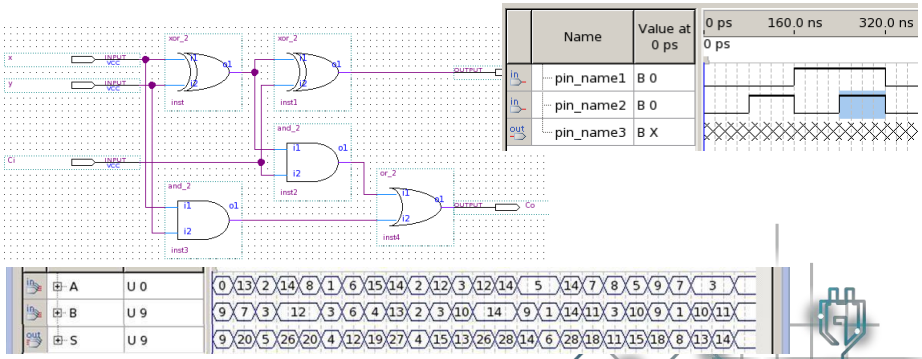
5

Quartus II Project Flow

1. Launch Quartus II software
2. Build a Quartus project
3. Realize your circuit in .bdf
   - .bsf as modules declaration
4. Compile your design
5. Import / build a .vwf file as testbench
6. Run functional simulation







Sequ. Ckt Design flow

- Gate-level schematic design flow:
  1. Divide system - Divide big design to a system of small modules
     - For each small module, (especially for FSM) do:
  2. Draw state graph - Think cautiously about your topic
  3. Make state table - (State assignment), truth table for Q^+ & output
  4. K-map - Simplify the logic (multi-output simplification)
  5. Boolean expression - The exact form you're going to make
  6. Draw it first - Be sure meet the requirements (ex. gate count)
  7. Design with tools - Implement it (.bdf or .v)
     - You can pack partial circuit as sub-module by .bsf + .bdf
  8. Verify & debug - Test some typical patterns, simulate it by .vwf

Lab2_1: Sequence detector

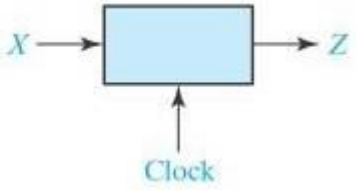
FSM: Mealy Machine and Moore Machine

Finite State Machine Design

- Sequential circuits usually built as a FSM
  - Mealy machine
  - Moore machine

- Let's take 101 sequence detector as an example
  - Textbook 14.1

FIGURE 14-1
Sequence Detector to Be Designed



$X =$	0	0	1	1	0	1	1	0	0	1	0	1	0	1	0	0
$Z =$	0	0	0	0	0	1	0	0	0	0	0	1	0	1	0	0
(time:	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15)

Draw State Graph, Build State Table

- Mealy machine

FIGURE 14-4
Mealy State Graph for Sequence Detector

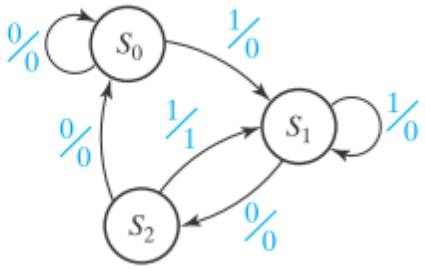


TABLE 14-1

Present State	Next State $X=0$	Next State $X=1$	Present Output $X=0$	Present Output $X=1$
S_0	S_0	S_1	0	0
S_1	S_2	S_1	0	0
S_2	S_0	S_1	0	1

TABLE 14-2

AB	A^+B^+ $X=0$	A^+B^+ $X=1$	Z $X=0$	Z $X=1$
00	00	01	0	0
01	10	01	0	0
10	00	01	0	1

- Moore machine

FIGURE 14-6
Moore State Graph for Sequence Detector

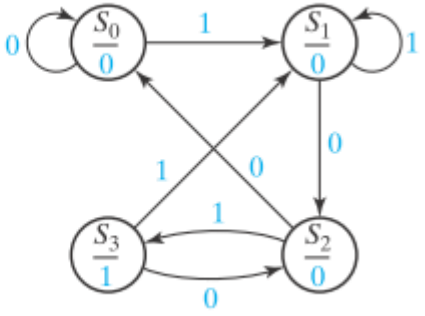


TABLE 14-3

Present State	Next State $X=0$	Next State $X=1$	Present Output(Z)
S_0	S_0	S_1	0
S_1	S_2	S_1	0
S_2	S_0	S_3	0
S_3	S_2	S_1	1

TABLE 14-4

AB	A^+B^+ $X=0$	A^+B^+ $X=1$	Z
00	00	01	0
01	11	01	0
11	00	10	0
10	11	01	1

Pre-draw & Boolean Function

- Mealy machine
  - $A^+ = X'B$
  - $B^+ = X$
  - $Z = XA$

AB	A^+B^+ $X=0$	A^+B^+ $X=1$	Z $X=0$	Z $X=1$
00	00	01	0	0
01	10	01	0	0
10	00	01	0	1

A^+

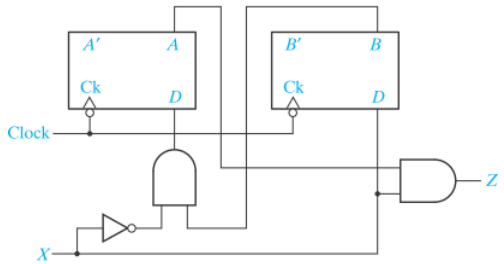
AB \ X	0	1
00	0	0
01	1	0
11	X	X
10	0	0

B^+

AB \ X	0	1
00	0	1
01	0	1
11	X	X
10	0	1

Z

AB \ X	0	1
00	0	0
01	0	0
11	X	X
10	0	1



- Moore machine
  - $A^+ = (A \oplus B \oplus X)\cdot(A+X')$
  - $B^+ = (A \oplus B)+A'X$
  - $Z = AB'$

AB	A^+B^+ $X=0$	A^+B^+ $X=1$	Z
00	00	01	0
01	11	01	0
11	00	10	0
10	11	01	1

A^+

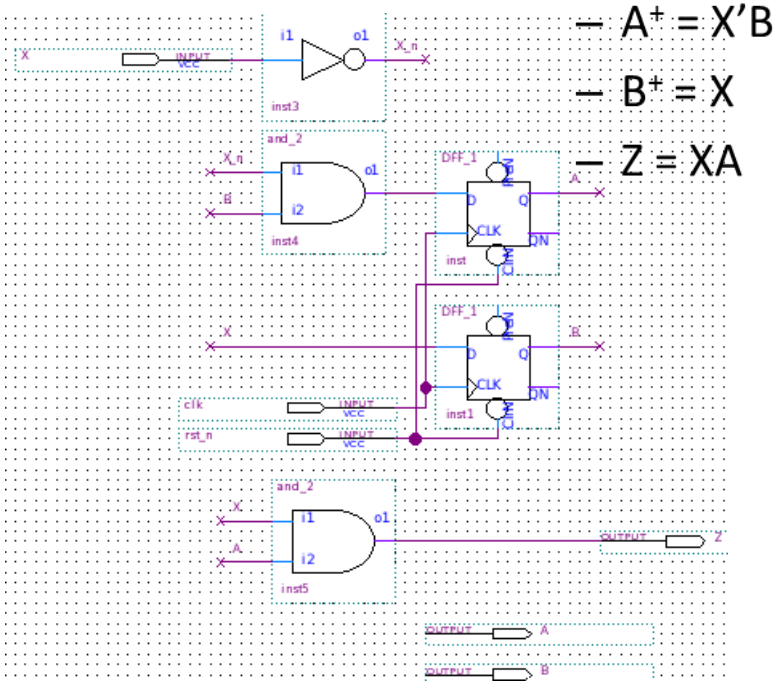
AB \ X	0	1
00	0	0
01	1	0
11	0	1
10	1	0

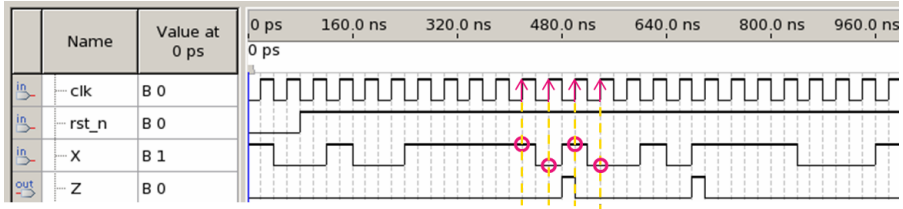
B^+

AB \ X	0	1
00	0	1
01	1	1
11	0	0
10	1	1

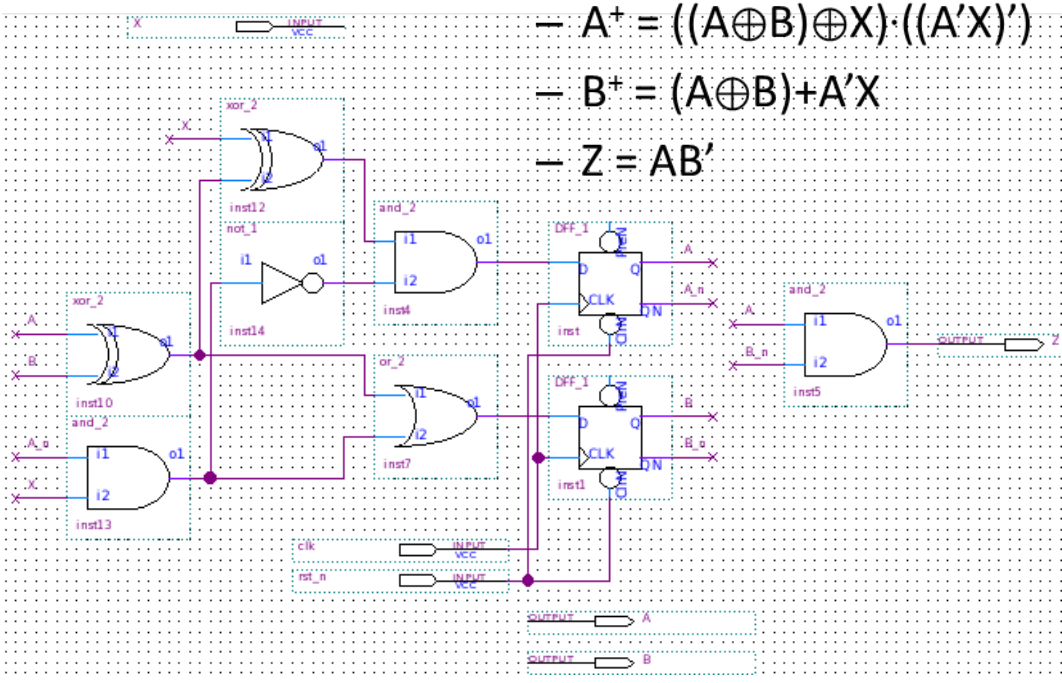
Quartus Design

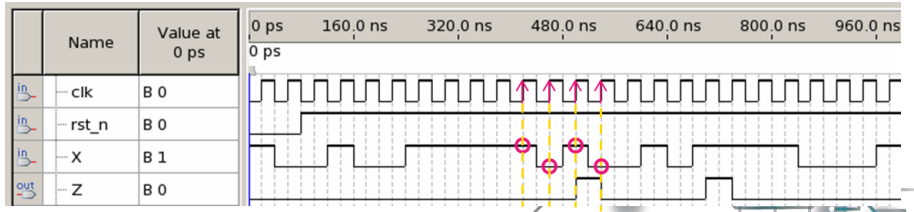
- Mealy machine
  - $A^+ = X'B$
  - $B^+ = X$
  - $Z = XA$





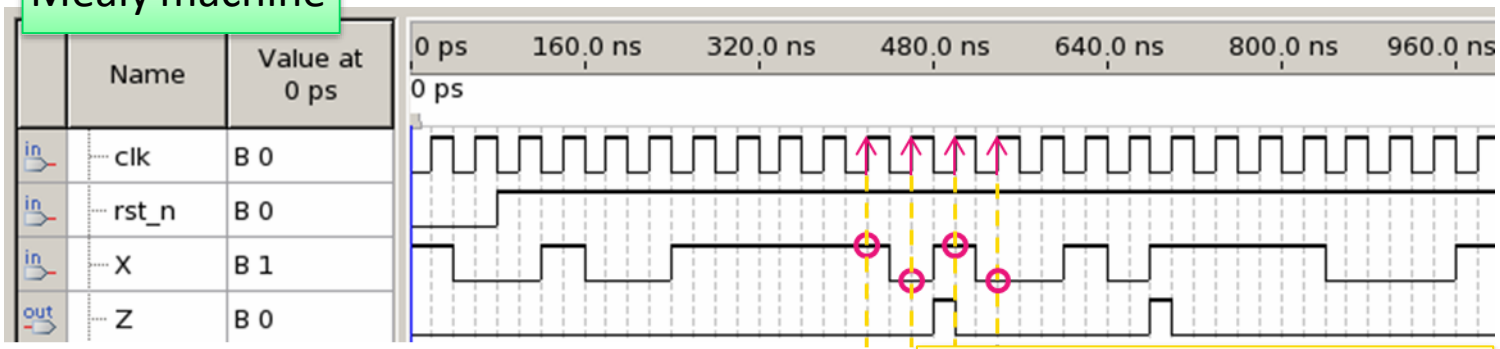
- Moore machine
  - $A^+ = ((A \oplus B) \oplus X) \cdot ((A'X)')$
  - $B^+ = (A \oplus B) + A'X$
  - $Z = AB'$





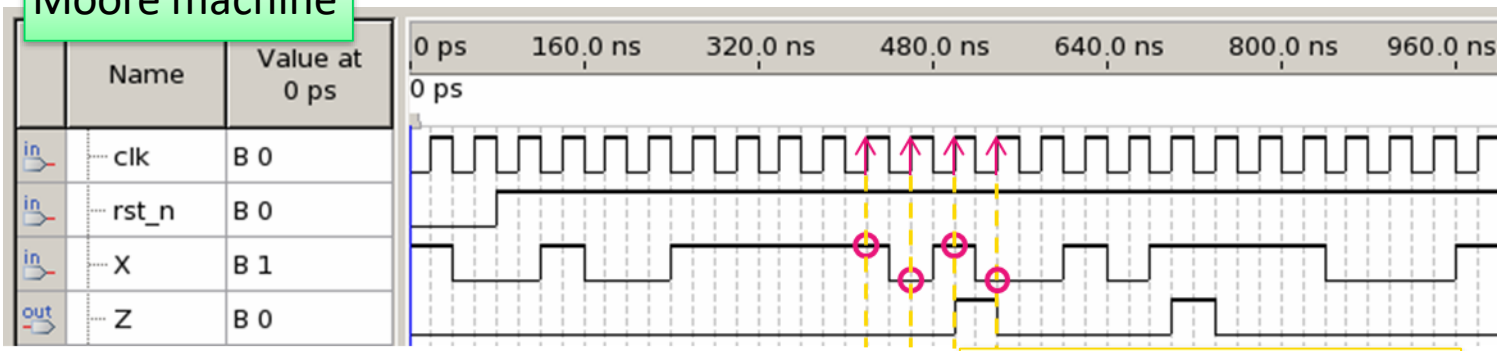
Result Comparison

Mealy machine



With false output but earlier

Moore machine



Output sustain one cycle

Preferred!

Tips for vwf:
To ensure that every inputs are caught properly at pos-edges
=> We change new inputs at neg-edges

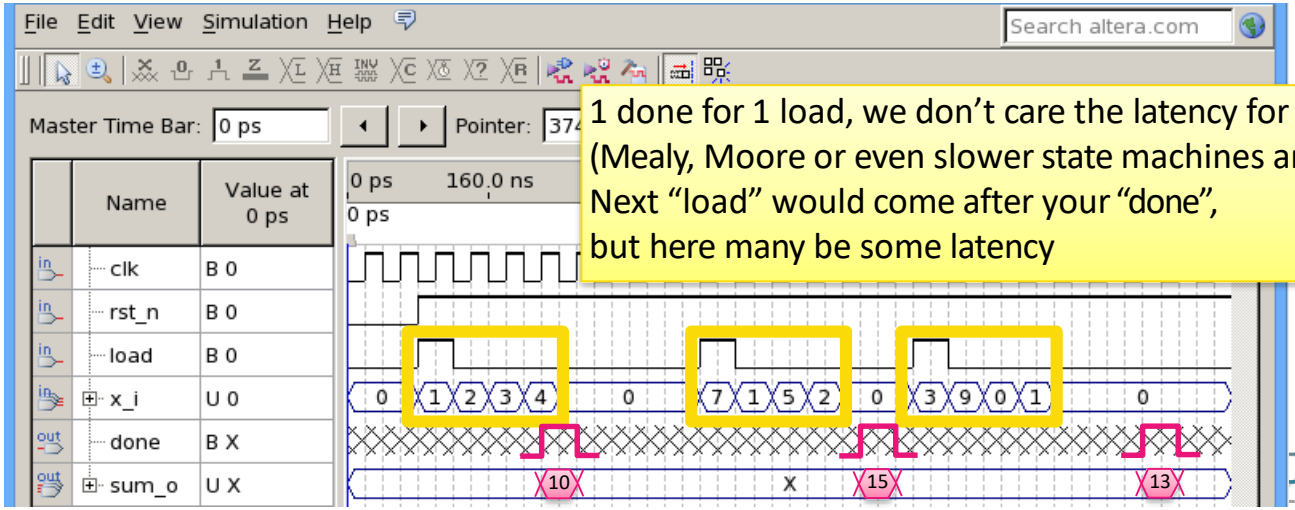
Topic Example: Accumulator

Notice

- An unsigned accumulator design is provided in this section.
- However, you are asked to design a 4-bit signed accumulator.
  - Any modification should be considered.

Short Topic: Unsigned Accumulator

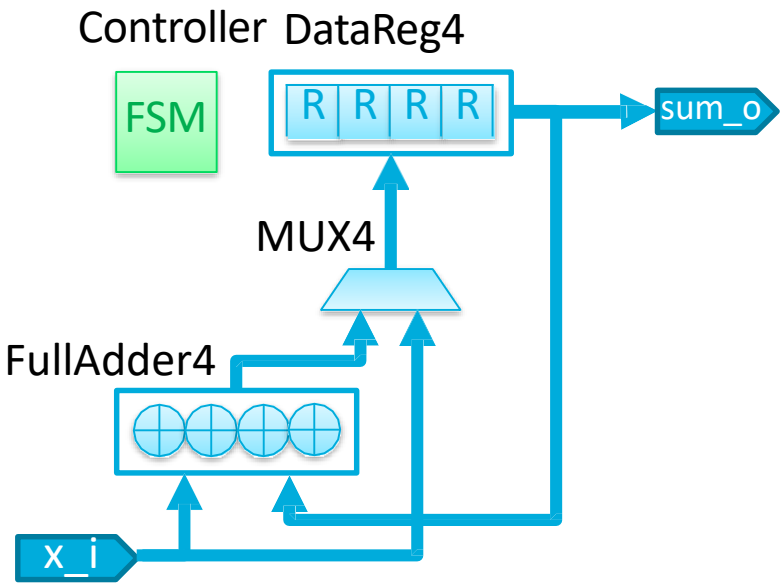
- Specification:
  - [Important] Must follow the spec. vigilantly in your final project!!
  - Function: summation of 4 4-bits unsigned integer
    - The first input (x_i[3..0]) would come with a “load” signal
      - All inputs changes at neg-edge, it’s easier for pos-edge design to take
    - After the 4 numbers are summed properly, give a “done” signal to denote the correct answers (sum_o[3..0])

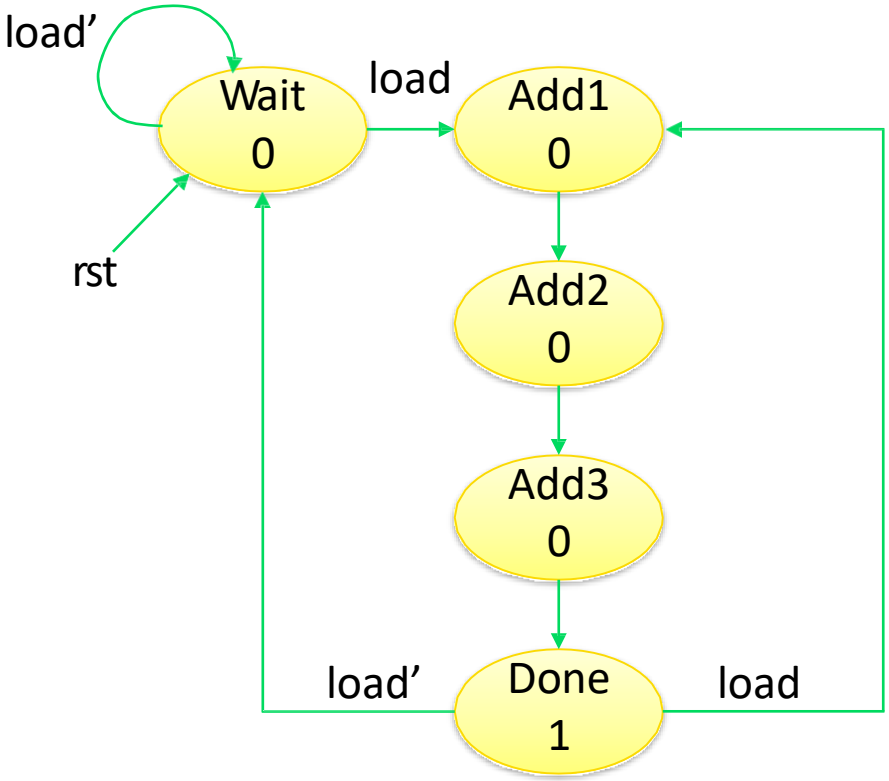


Steps to Design a Small System

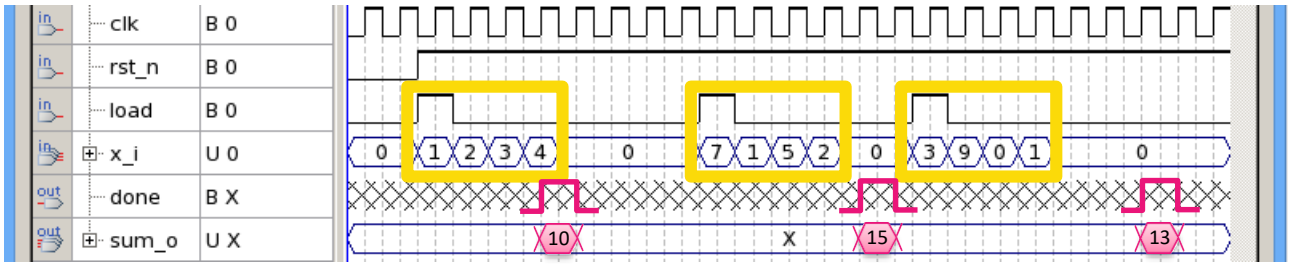
- Gate-level schematic design flow:
  1. Read the spec. vigilantly!!
  2. Divide system - Pre-draw a rough architecture
  3. Draw state graph - Think cautiously about your topic
  4. Build state table - (State assignment), truth table for Q^+ & output
  5. K-map - Simplify the logic (multi-output simplification)
  6. Boolean expression - The exact form you're going to make
  7. Draw it first - Be sure meet the requirements (ex. gate count)
  8. Design with tools - Implement it (.bdf or .v)
     - You can pack partial circuit as sub-module by .bsf + .bdf
  8. Verify & debug - Test some typical patterns, simulate it by .vwf

Draw State Graph





1 done for 1 load, we don't care the latency for done
Next "load" would come after your "done"

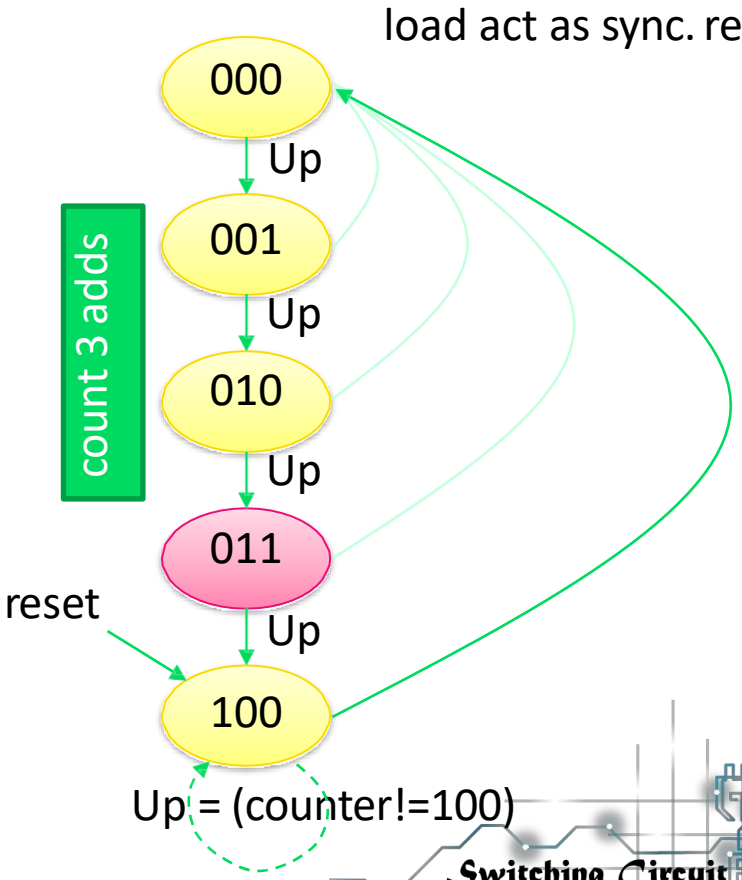


Build State Table

Traditional method

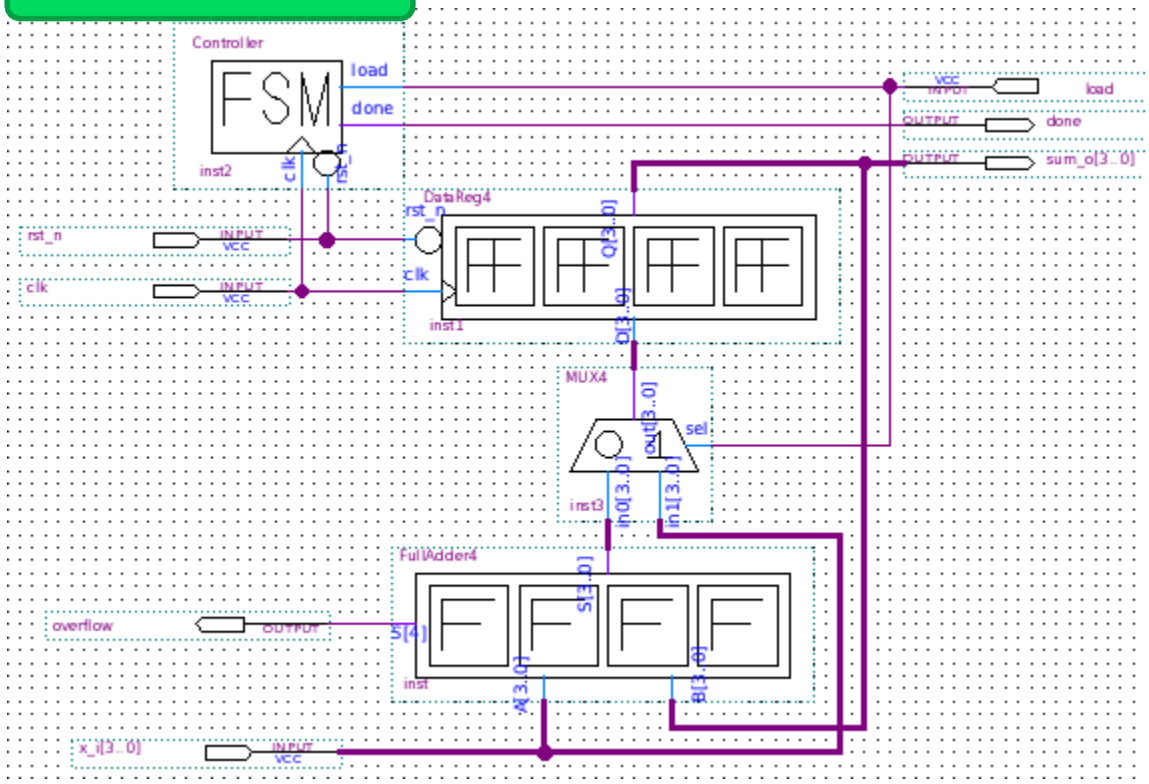
State	State+ load=0	State+ load=1	Output (done)
Wait	Wait	Add1	0
Add1	Add2	Add2	0
Add2	Add3	Add3	0
Add3	Done	Done	0
Done	Wait	Add1	1

Tricky method : use counter

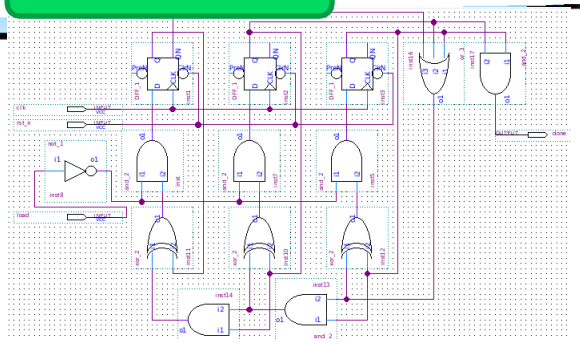


Quartus Files

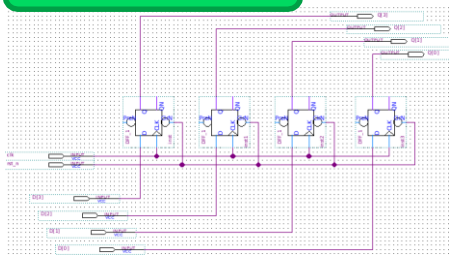
Accumulator4.bdf



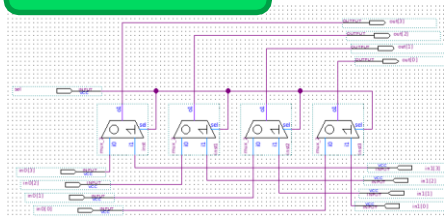
Controller.bdf



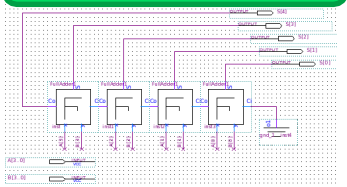
DataReg.bdf



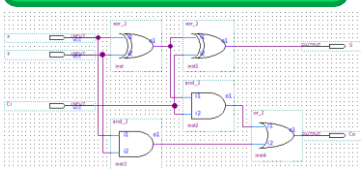
MUX4.bdf



FullAdder4.bdf



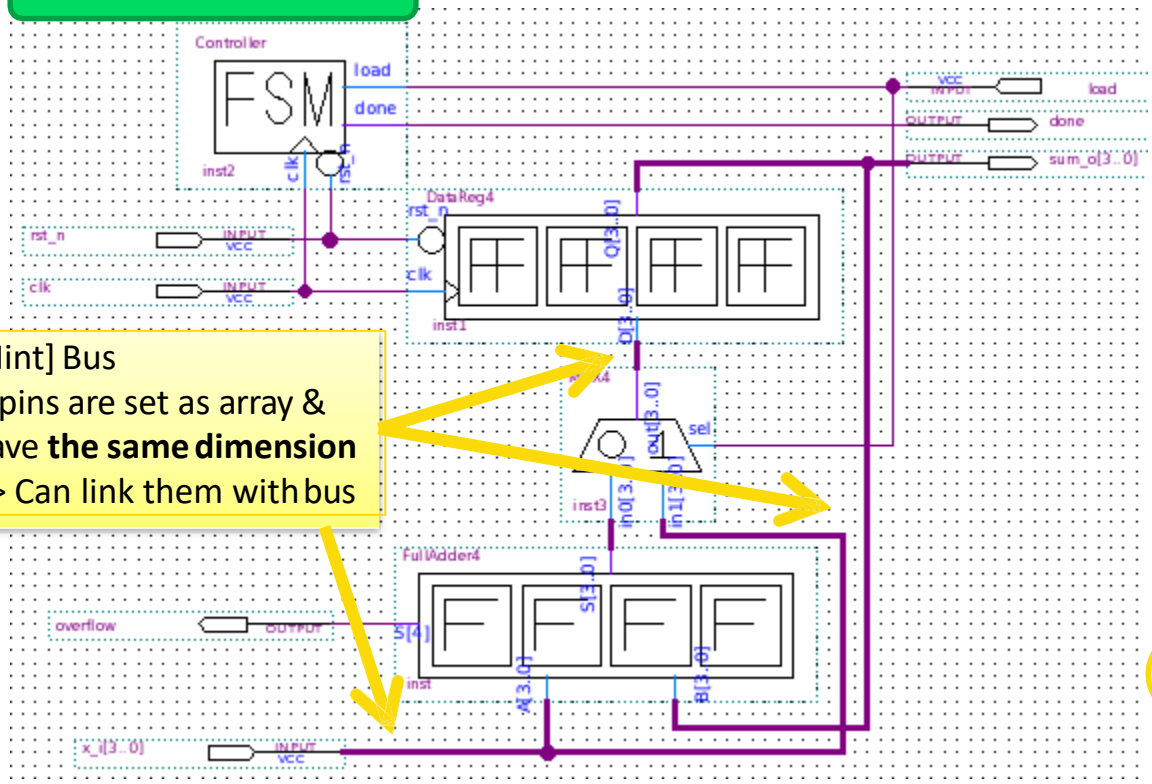
FullAdder1.bdf



Switching Circuit & Logic Design

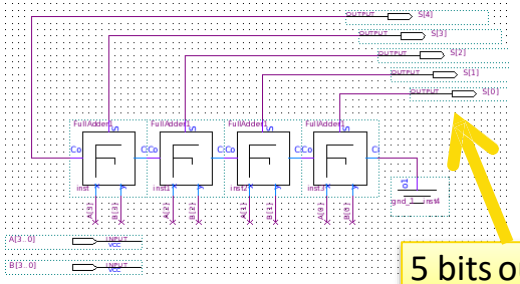
[Hint] Bus

Accumulator4.bdf



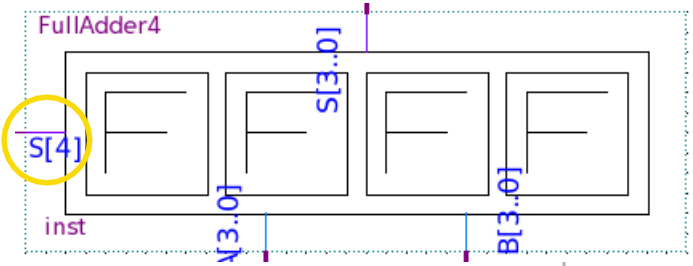
[Hint] Bus
if pins are set as array & have **the same dimension**
=> Can link them with bus

Q: How can I do if the dimension mismatch?



5 bits output

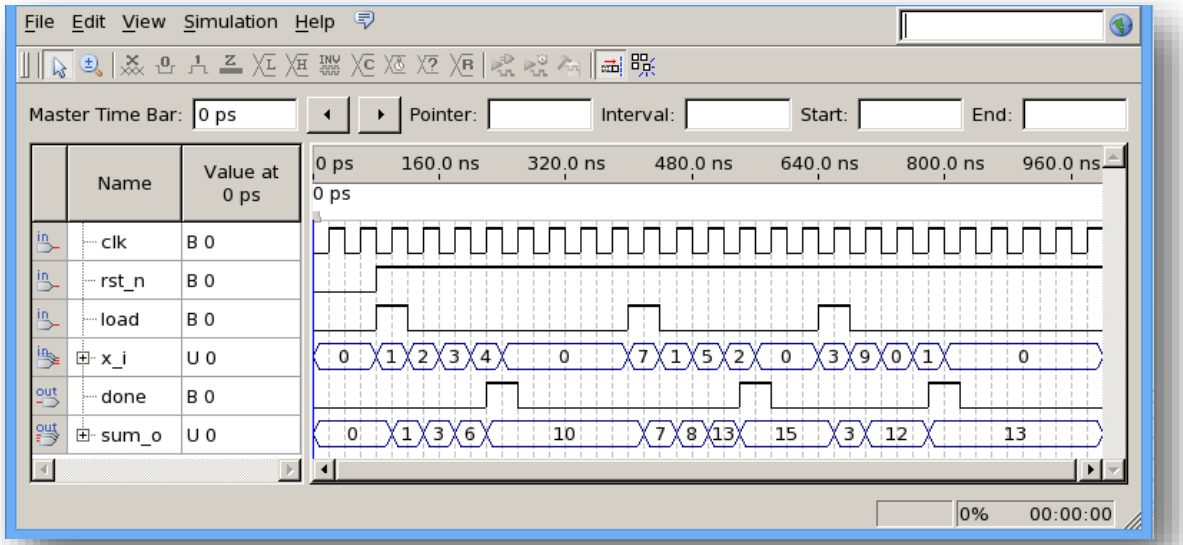
A: Can separate them in bsf



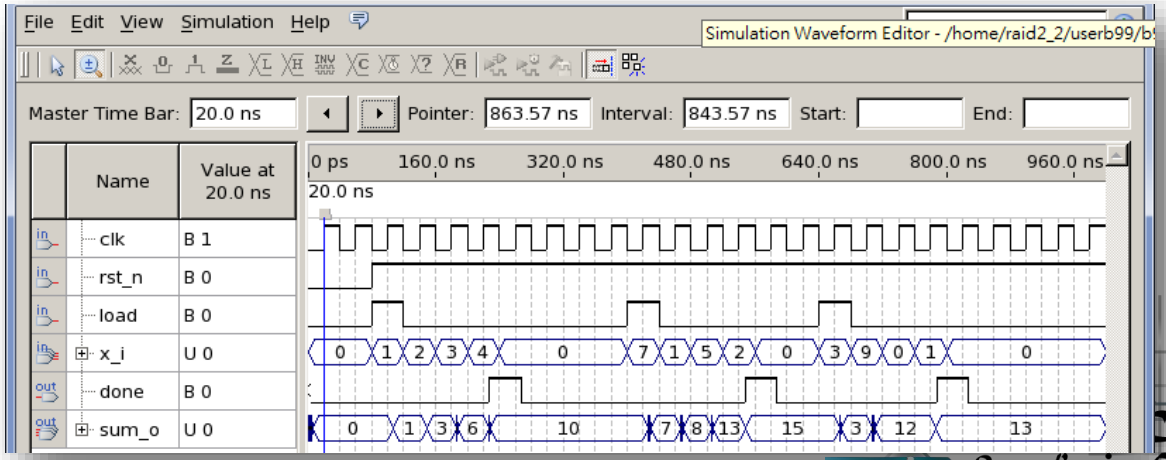
Switching Circuit & Logic Design

Run Simulation to Verify

Functional Simulation

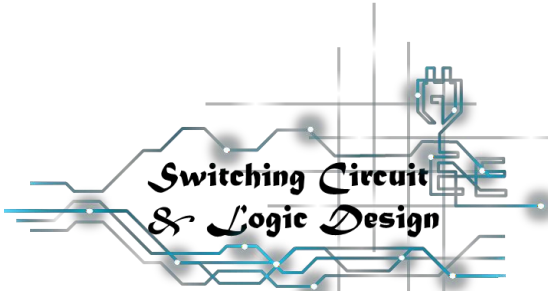


Timing Simulation

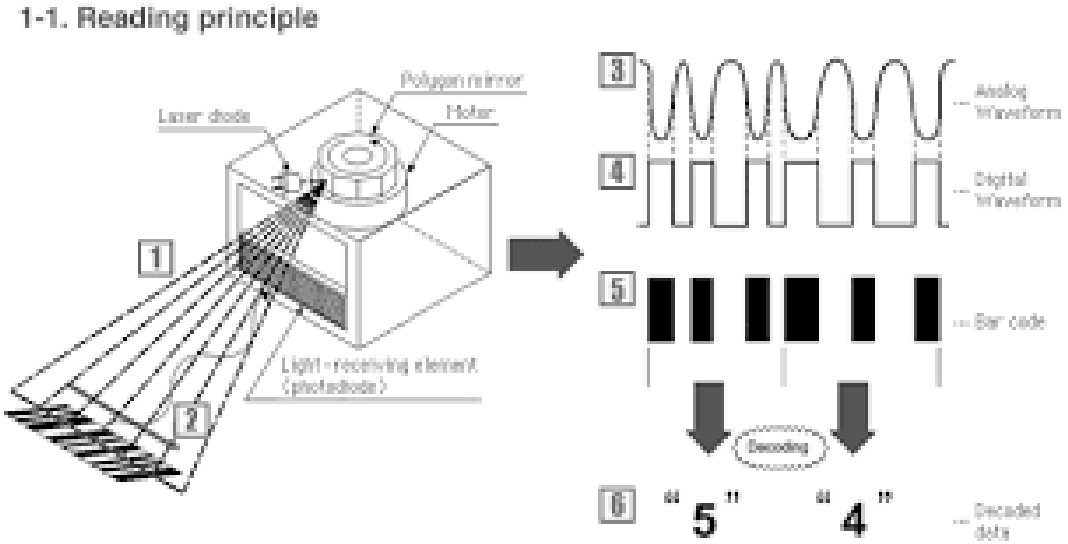


Lab2_3: Weighted Sequence Counter

Advanced exercise



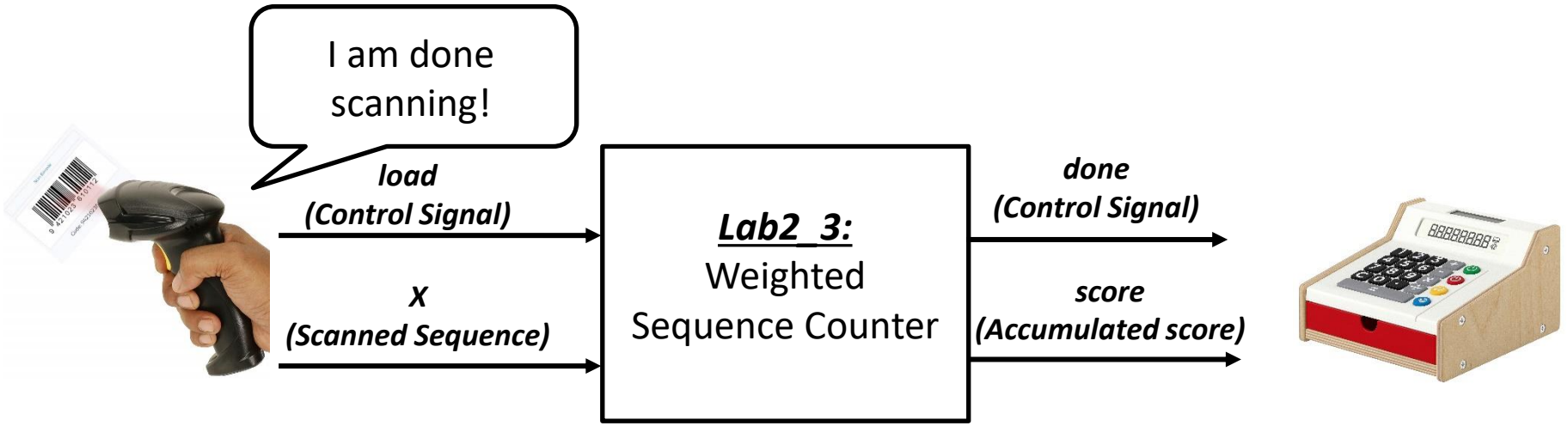
Bar Code Scanner

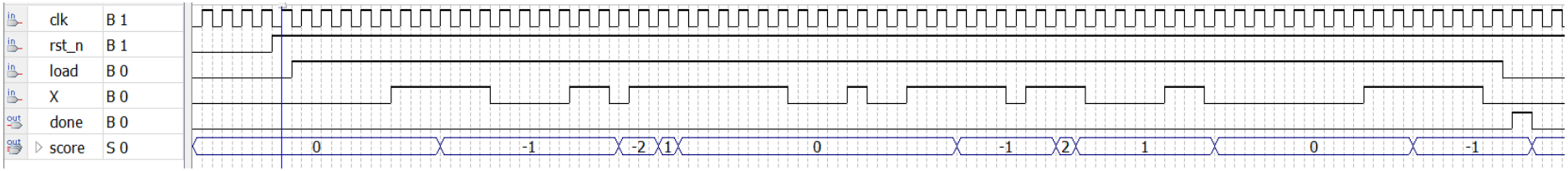


0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0

Lab2_3: Do something on detected sequence!

System Schematic Diagram

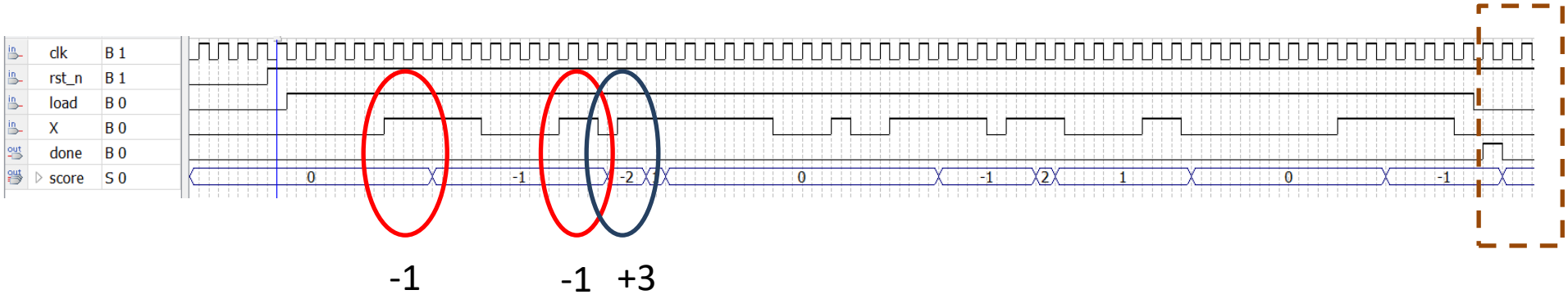




Switching Circuit & Logic Design

Function of Weighted Sequence Counter

Reset to zero and start detecting the next sequence



Input:

Zero is detected:
Need to output score

load: 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1……,0 ,0

X: 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1……,0 ,0

1 2 3 1

Score = -1
Score = 3

Output:

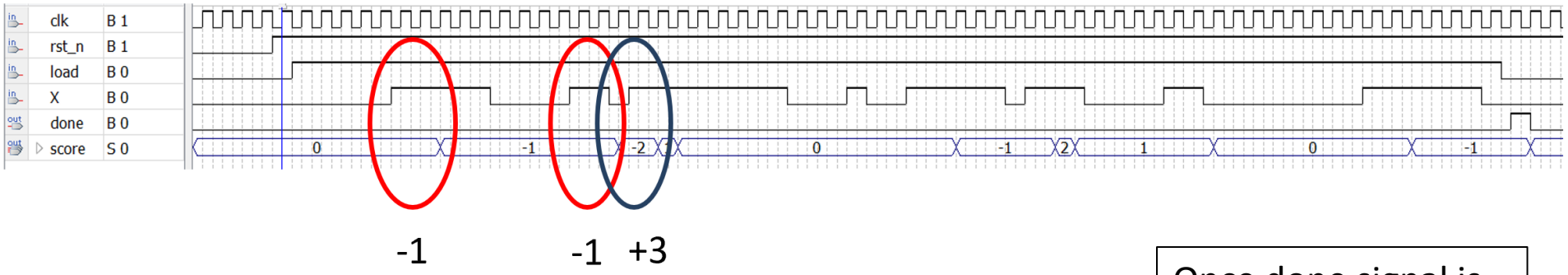
done: 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0……,0 ,1

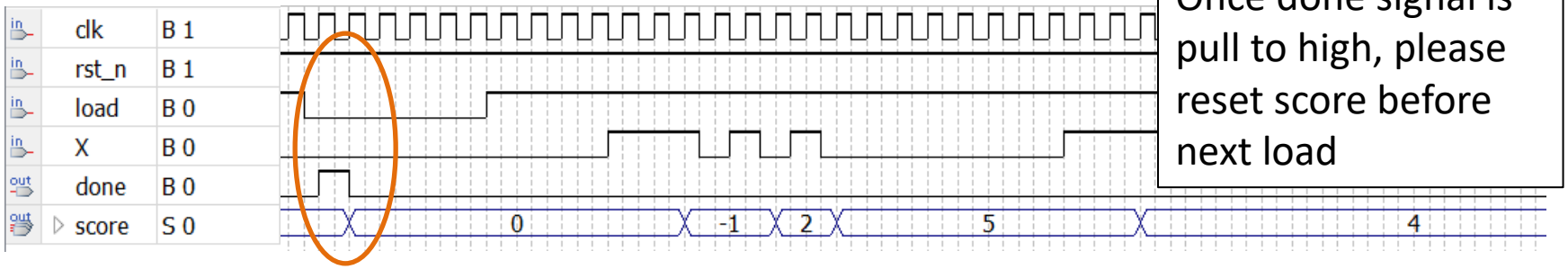
score: 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0……,0 ,-1

Score is valid when done signal is pull to high

Switching Circuit & Logic Design

Specification for Weighted Sequence Counter

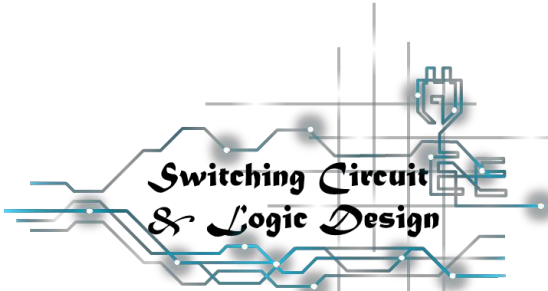




Once done signal is pull to high, please reset score before next load

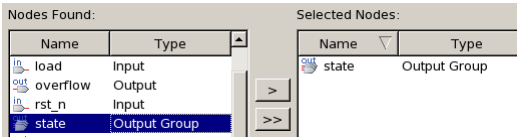
Name	Width	Type	Radix	Description
clk	1	in		Clock, duty cycle 50%
rst_n	1	in		Asyc. reset when (rst_n==0)
load	1	in		Marks a new operation & inputs
X	1	in		Sequence
done	1	Out		Marks the output & end of a work
score[3..0]	4	Out	Signed	Accumulated Score

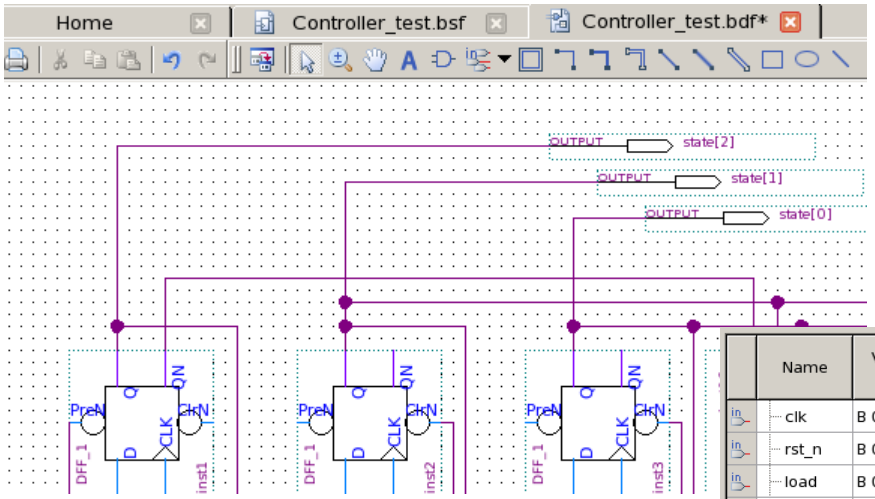
Debugging Experience

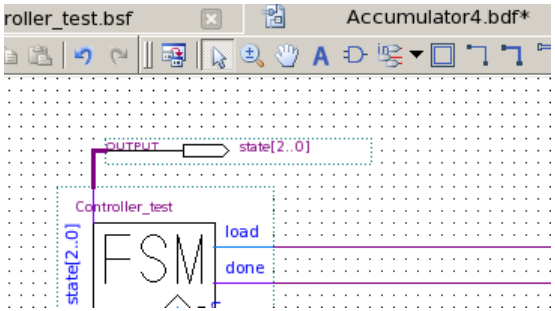


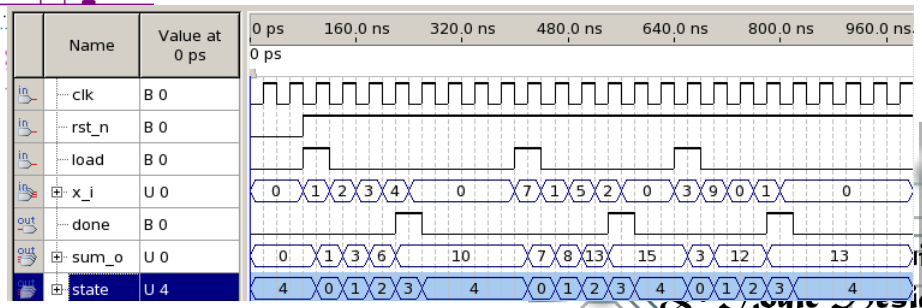
Tips 1: Grab Signal by Output Pins

- Like “cout” in C++, let’s to do something to “see” them
- For digital ckt debugging, we usually watch the “states” first
  - Leave corresponding output pins in bdf & bsf => watch in vwf
    - Edit→Insert→Insert Node or Bus..
  - Unconnected output is OK for us



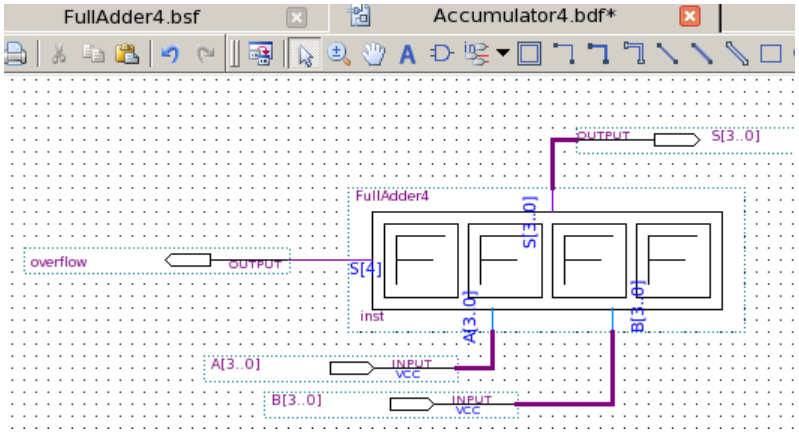


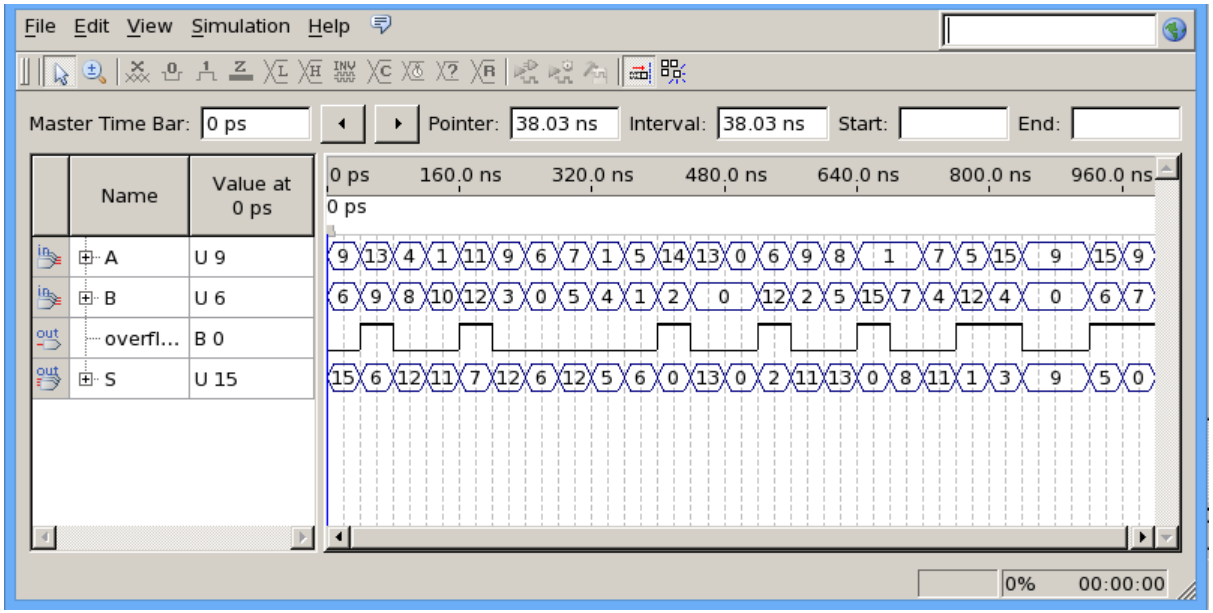




Tips 2: Test by Parts

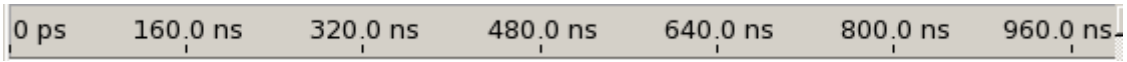
- Test each module separately
- Put a single module in top bdf
  - Add corresponding pins
  - Compile
  - Make a vwf yourself, simulate

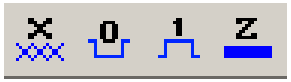


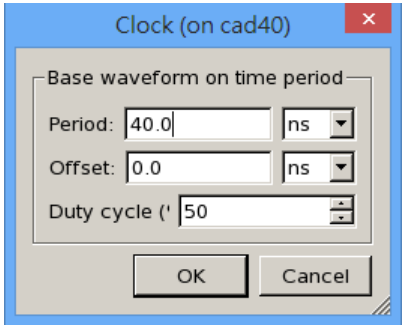


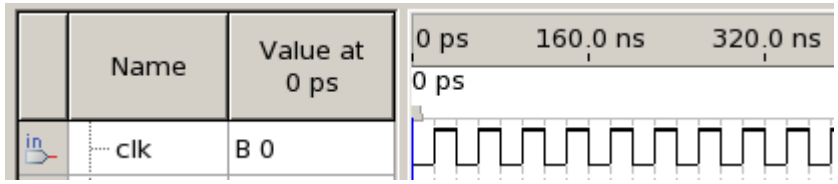
Switching Circuit & Logic Design

More about VWF (1/2)

- Time scale 

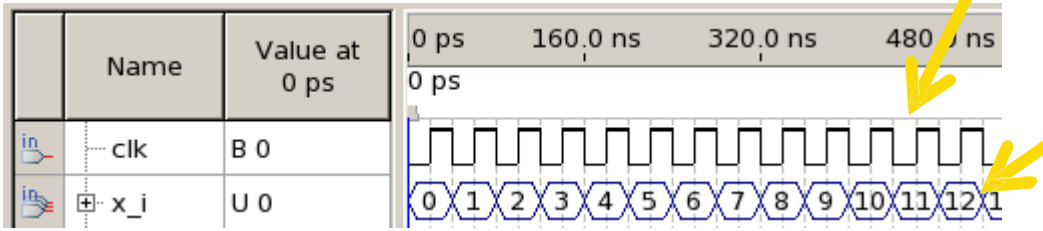
  - Edit→Set End Time… (default: 1µs)
  - Edit→Grid Size… (default: 10ns)
- Signal Tool Bar 
  - Basic Signal Setting 
  - Clock Setting 





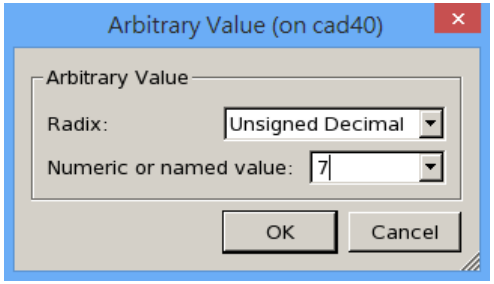
More about VWF (2/2)

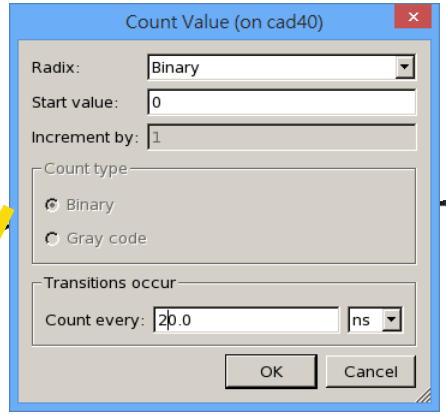
- Signal Tool Bar
  - – Counter 
    - Another way to set clock…

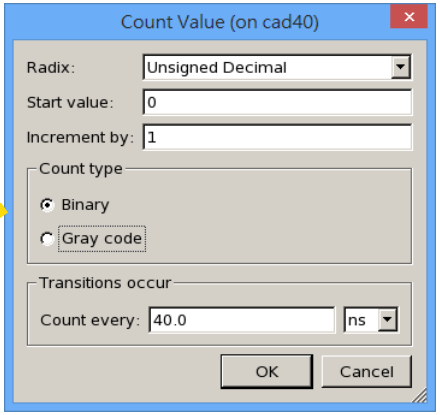


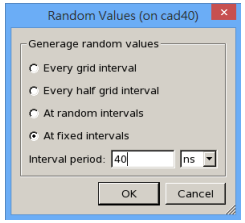
  - – Random value 

  - – Set value manually 



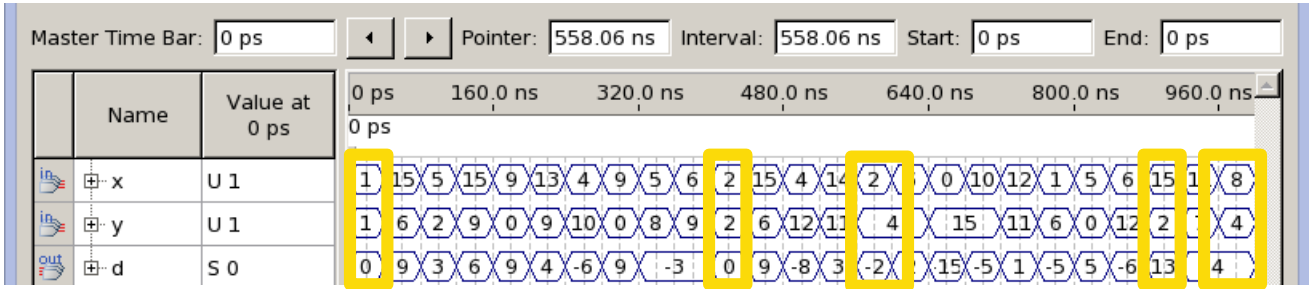






Tips 3: Check Simple Cases First

- If you encounter a bug and have little idea
  - > Checking the simple cases in vwf is easier to find out what's wrong
  - ex. FullSubtractor4



Tips 4: Check Special Cases

- If you pass some simples tests
  - Does not indicate that your system is reliable
  - Test some extreme, rare cases to see whether your system is robust
    - Often find bugs here
  - ex. FullSubtracter4
    - 0 – 15, 0 – 1, 8 – 1, 1 – 2, 1 - 15

Common Problems (1/4)

- ★ Please read the problem description carefully.
- About file names:
  - Only the name of the top module is specified (e.g. FA4). Random names are allowed for the sub-circuits (e.g. 1-bit full adder can be named FA, fulladder, fa1,…).
  - Please make sure all the names of the sub-modules are consistent in higher-level designs.
- Please examine there is no Chinese text or space in the directory name.
- In the case that the created files are not under the project, which makes them unable to be found.
  - Please add those files into the project manually.

Common Problems (2/4)

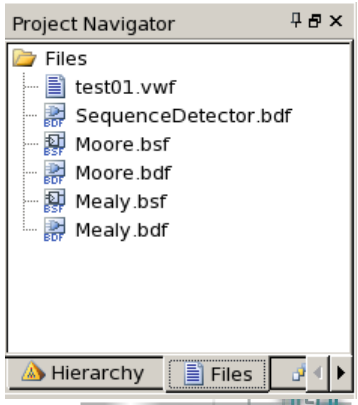
- .bsf files
  - Update your new bsf file in your top module
    - Right click symbol on top module and click “update symbol or block”
- Top design module
  - Top module name must be identical to project name

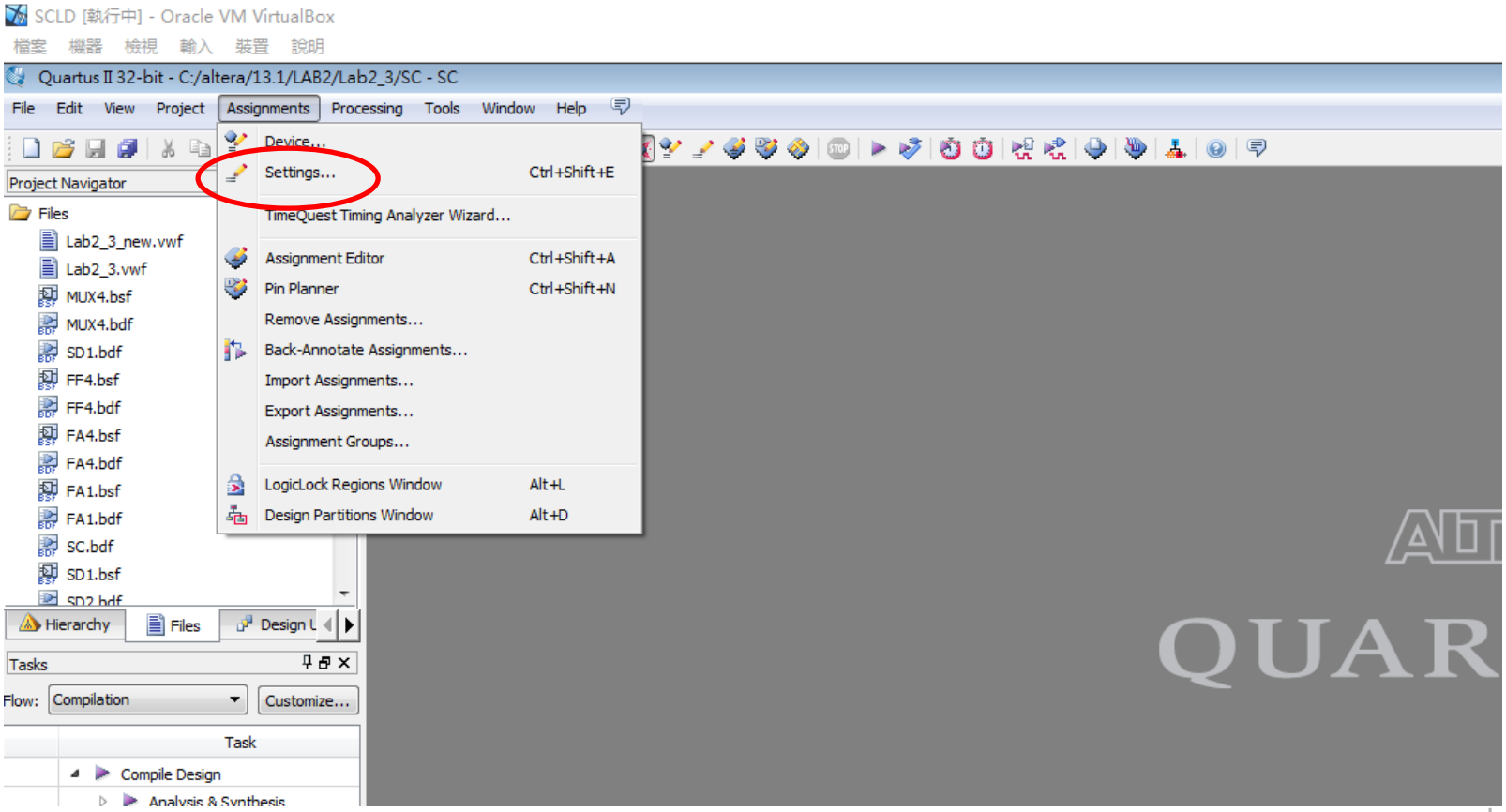
Common Problems (3/4)

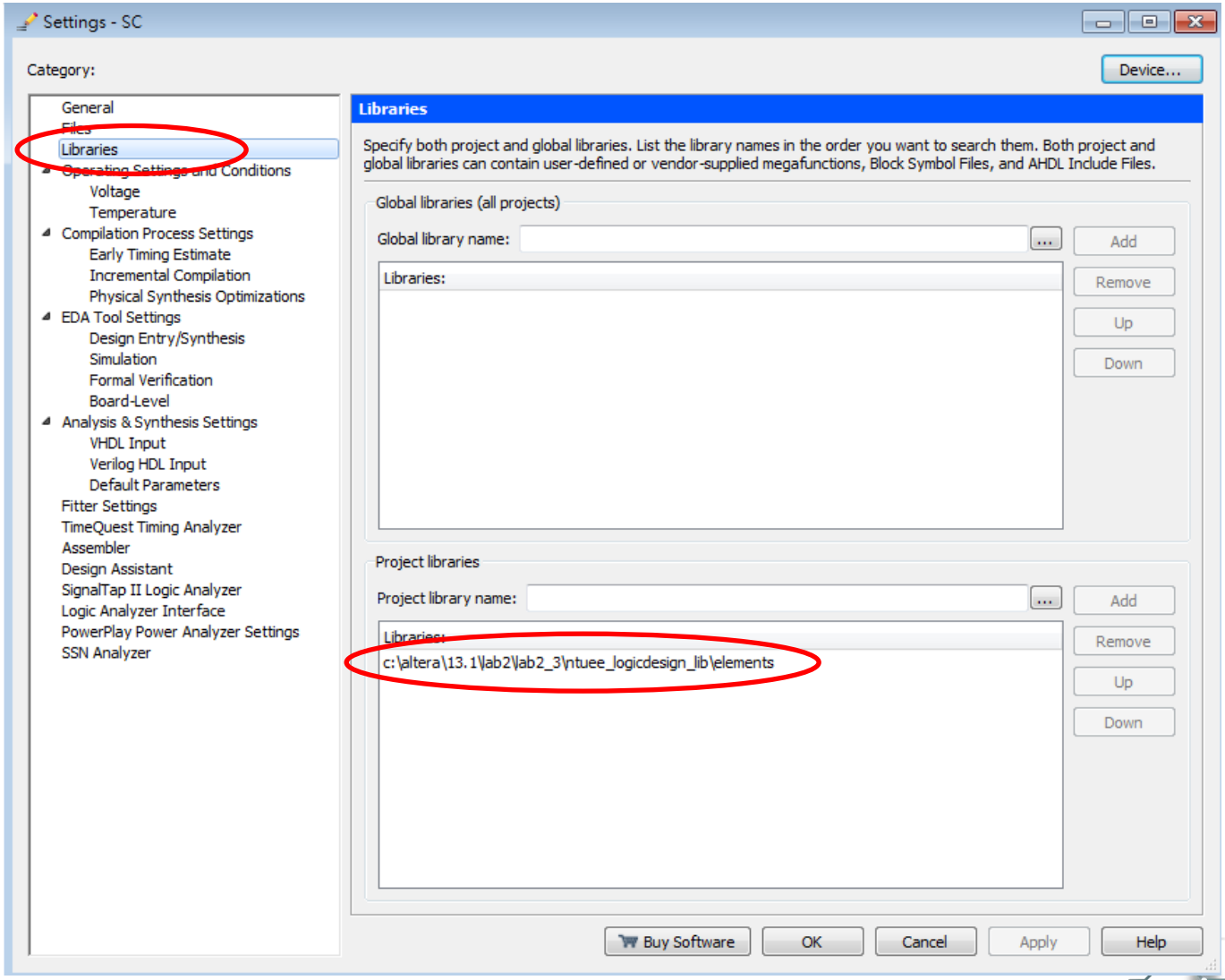
- About .vwf files
  - Only output the result generated by compiled circuit
    - Re-compile first if you have modified the circuit
  - Make sure the “radix” setting matches those in specification
  - Simulation results are all XX
    - Check the pin names
    - Check whether there exists some un-connected wires in your design
      - Be careful when you connect wires to module. (Only the boundary is available)

Common Problems (4/4)

- Common reasons for compile error
  - wire names contain (` ) ` +
    - Must not appear in the variable names in programming code such as C++ …
  - repetitive instance names ...
    - This is a bug of Quartus. It should avoid repeating instance names, but sometimes it will fail.
  - Mismatch between .bdf and .bsf files
    - File name mismatch
    - I/O mismatch
  - File/Naming conflict
    - Check if something missed in Project Navigator -> Files.
    - The module instance should be linked to the newest bsf.
    - The names of modules cannot be identical to the element names in the libraries.




SCLD [執行中] - Oracle VM VirtualBox
檔案 機器 檢視 輸入 裝置 說明
Quartus II 32-bit - C:/altera/13.1/LAB2/Lab2_3/SC - SC
File Edit View Project Assignments Processing Tools Window Help
Device...
Settings... Ctrl+Shift+E
TimeQuest Timing Analyzer Wizard...
Assignment Editor Ctrl+Shift+A
Pin Planner Ctrl+Shift+N
Remove Assignments...
Back-Annotate Assignments...
Import Assignments...
Export Assignments...
Assignment Groups...
LogicLock Regions Window Alt+L
Design Partitions Window Alt+D
Project Navigator
Files
Lab2_3_new.vwf
Lab2_3.vwf
MUX4.bsf
MUX4.bdf
SD1.bdf
FF4.bsf
FF4.bdf
FA4.bsf
FA4.bdf
FA1.bsf
FA1.bdf
SC.bdf
SD1.bsf
Hierarchy
Files
Tasks
Flow: Compilation
Customize...
Task
Compile Design
Analysis & Synthesis



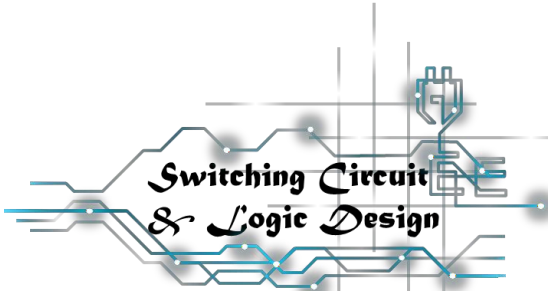
Late submission policy

- Deadline
  - Lab1: 11/24 23:59:59
  - Lab2: 12/8 23:59:59

- 20% off for 1 day late, 0 point for late submission over 5 days
  - E.g. Lab1 submitted on 11/25 will get 90 points at most
    Lab1 submitted on 11/26 will get 80 points at most
    Lab1 submitted on/after 11/30 will get 0 point

- Start it earlier!

Reference



Reference

- Textbook
  - Fundamentals of Logic Design, Charles H. Roth, Jr., Larry L. Kinney
- Dclab lecture
  - Verilog Coding Guideline, 吳柏辰
  - My First FPGA for Altera DE2-115 Board, 吳柏辰
- CVSD lecture
  - Computer-aided VLSI System Design Linux / Unix Tutorial, 陳滿蓉
- Author of these slides
  - Ver. 1: Ming-Lun Hsieh
  - Ver. 2: Hung-Yu Tseng
  - -- Ver. 3： 吳岳霖
  - -- Ver. 4: Cheng-You Tsai