



Welcome to The Hardware Design & Lab!

Fall 2024

Lab 2: Advanced Gate-Level Verilog

Prof. Chun-Yi Lee

Department of Computer Science
National Tsing Hua University

Agenda

- Lab 2 Outline
- Lab 2 Basic Questions
- Lab 2 Advanced Questions



Lab 2 Outline

- Basic questions (1.5%)
 - Individual assignment
 - Due on **9/19/2024 (Thu). In class.**
 - Only demonstration is necessary. Nothing to submit.
 - Please **draw the circuits of question 1, and explain the differences between the adders of question 3 in your report.**
- Advanced questions (5%)
 - Group assignment
 - EEClass submission due on **9/26/2024 (Thu). 23:59:59.**
 - Demonstration on your FPGA board (**In class**)
 - Assignment submission (**Submit to EEClass**)
 - Source codes and testbenches
 - Lab report in PDF

Lab 2 Rules

- Only gate-level description is permitted
 - Only basic logic gates are ALLOWED (AND, OR, NAND, NOR, NOT)
 - Sorry, no xor & xnor
- Please **AVOID** using
 - Continuous assignment (e.g., **assign =**, **wire =**) and conditional operators (e.g., **: ?**)
 - Behavioral operators (e.g., **=**, **!**, **%**, **&**, *****, **+**, **/**, **<**, **>**, **^**, **|**, **~**)

Lab 2 Submission Requirements

- Source codes and testbenches
 - Please follow the templates **EXACTLY**
 - We will test your codes by TAs' testbenches
- Lab 2 report
 - Please submit your report in a single **PDF** file
 - Please **draw** the gate-level circuits of your designs (**please use computer softwares to draw your figures**)
 - Please **explain** your designs in detail
 - **Please explain how you test your design**
 - Please describe the contributions of each member
 - What you have **learned** from Lab 2

Agenda

- Lab 2 Outline
- **Lab 2 Basic Questions**
- Lab 2 Advanced Questions



Basic Questions

- Individual assignment
- Verilog questions (due on **9/19/2024 (Thu). In class.**)
 - (Gate Level) NAND gate only
 - (Gate Level) 3-input majority gate
 - (Gate Level) 1-bit full adder & half adder
- Demonstrate your work by **waveforms**

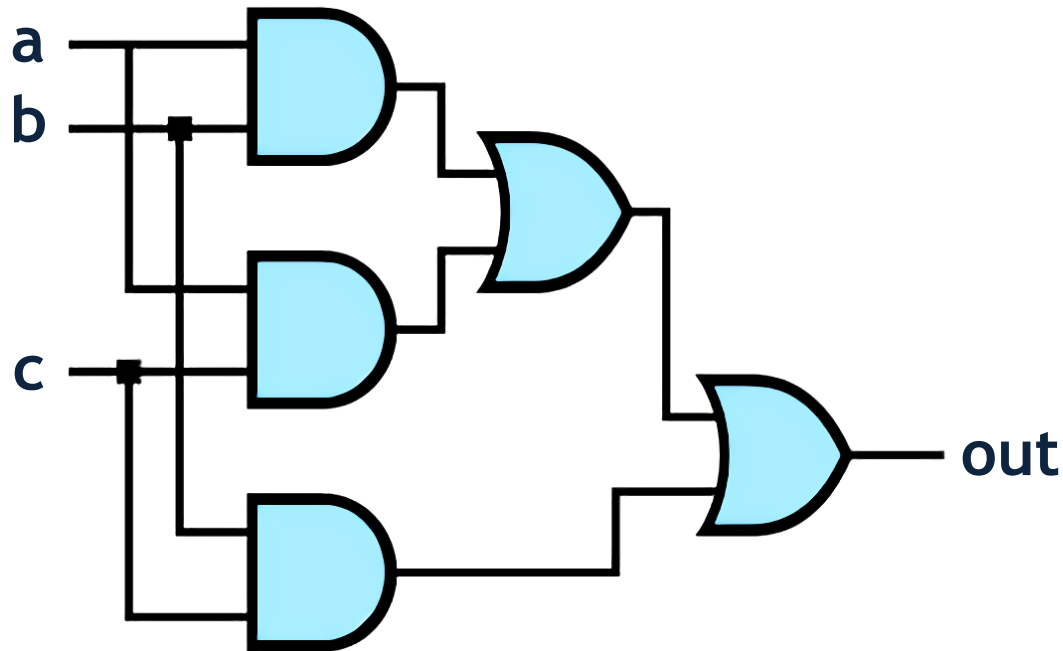
Verilog Basic Question 1

- (Gate Level) NAND gates only
 - Use **NAND gates only** to realize the following functions
 - **NOT, NOR, AND, OR, XOR, XNOR, NAND**
 - Input/Output: a (1bit), b (1bit), sel (3 bits), out (1 bit)
 - Please **draw your circuits** in your report

sel [2:0]	out
000	out = a nand b
001	out = a and b
010	out = a or b
011	out = a nor b
100	out = a xor b
101	out = a xnor b
110 & 111	out = ! a

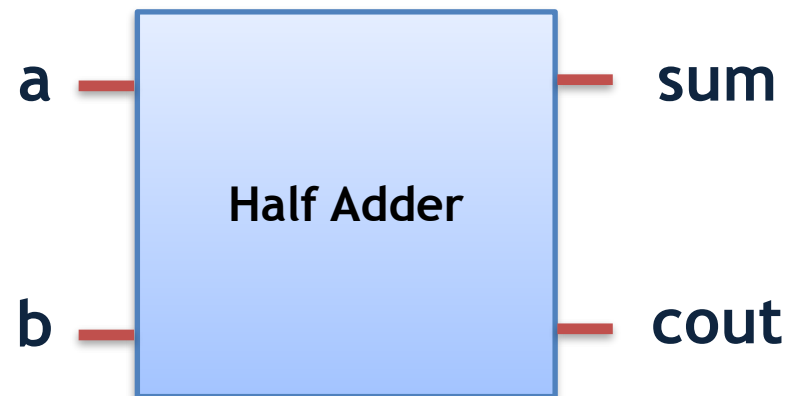
Verilog Basic Question 2

- (Gate Level) 3-input majority gate
 - Use **NAND gates only** to realize the following circuit
 - Please reuse the modules implemented in Question 1
 - Input/Output: a (1bit), b (1bit), c (1 bit), out (1 bit)



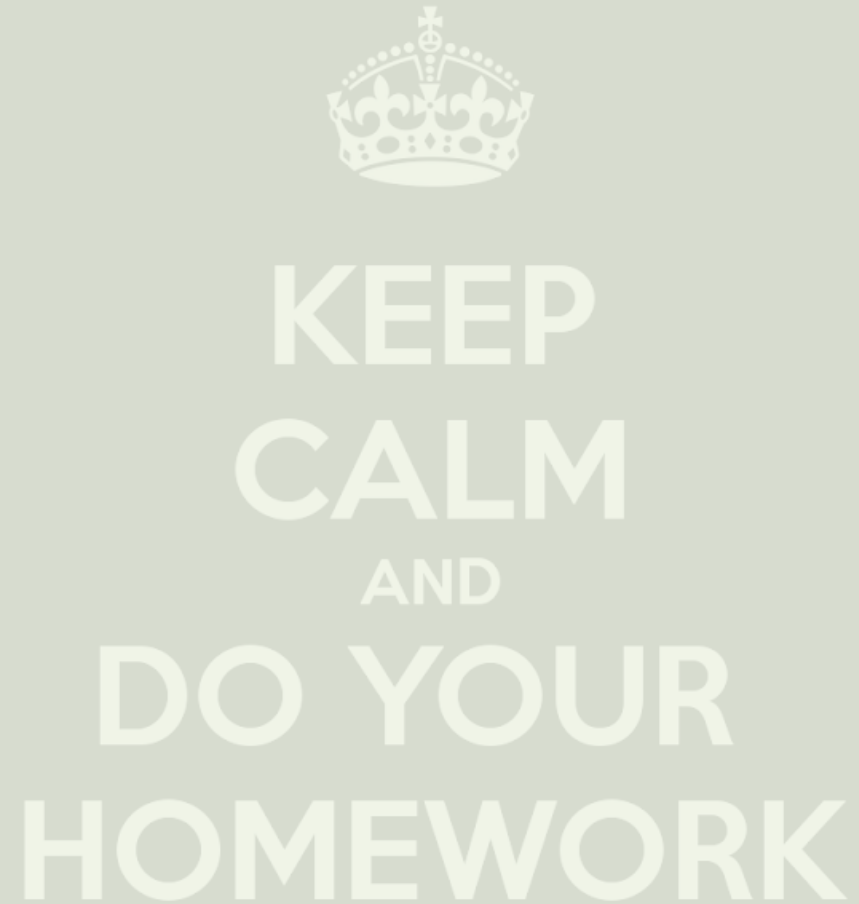
Verilog Basic Question 3

- (Gate Level) 1-bit full adder & half adder
- Please design two modules: one for a 1-bit full adder and one for a 1-bit half adder, use **NAND gates only**
- Please reuse the module of your majority gate from the basic question 2 for the 1-bit full adder design
- Please **explain** the difference between these two adders in your **report**.



Agenda

- Lab 2 Outline
- Lab 2 Basic Questions
- **Lab 2 Advanced Questions**

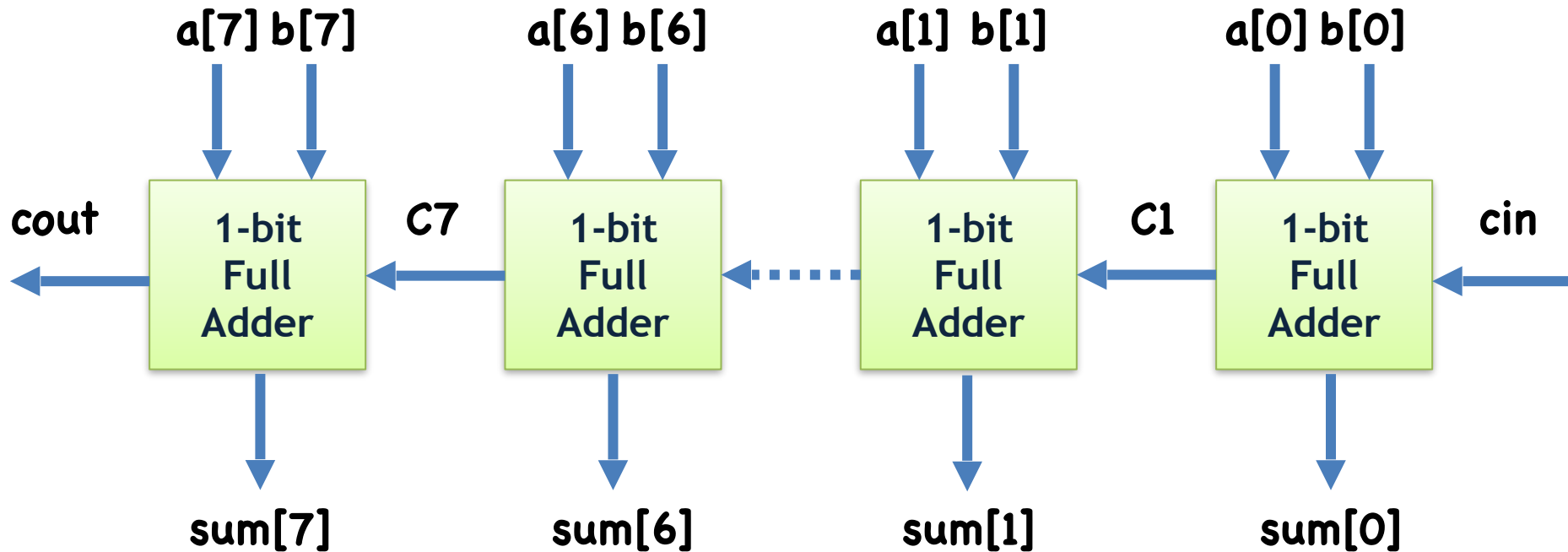


Advanced Questions

- Group assignment
- Verilog questions (due on 9/26/2024 (Thu). 23:59:59.)
 - **Necessary:** (Gate Level) 8-bit ripple carry adder (RCA)
 - **Necessary:** (Gate Level) Decode and execute
 - **Necessary:** (Gate Level) 8-bit carry-lookahead (CLA) Adder
 - **Necessary:** (Gate Level) 4-bit multiplier
 - **Necessary:** An exhaustive testbench design
- FPGA demonstration (due on 9/26/2024 (Thu). In class.)
 - **Necessary:** (Gate Level) Decode and execute

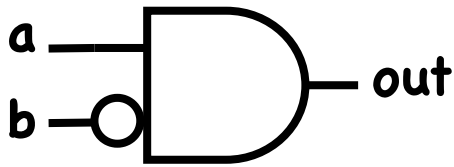
Verilog Advanced Question 1

- (Gate-level) 8-bit ripple-carry adder (RCA)
- Instantiate the 1-bit full adder module from the Basic Question 3
- Use **NAND gates only**



Verilog Advanced Question 2

- (Gate Level) Decode and execute
 - Please use the universal gate depicted on the bottom left corner only to implement the basic logic gates listed below.
 - Please draw your circuits of your basic logic gates (AND, OR, NOT ...) in your report
 - Implement your universal gate in Universal_Gate.v (the module name is also Universal_Gate) and instantiate it in your design. Do not submit this file and ensure that your design uses no primitive logic gates.
 - Use your own basic logic gate modules to realize the following functions specified in the table defined on the bottom-right corner
 - Input/Output: rs (4 bits), rt (4 bits), sel (3 bits), rd (4 bits)

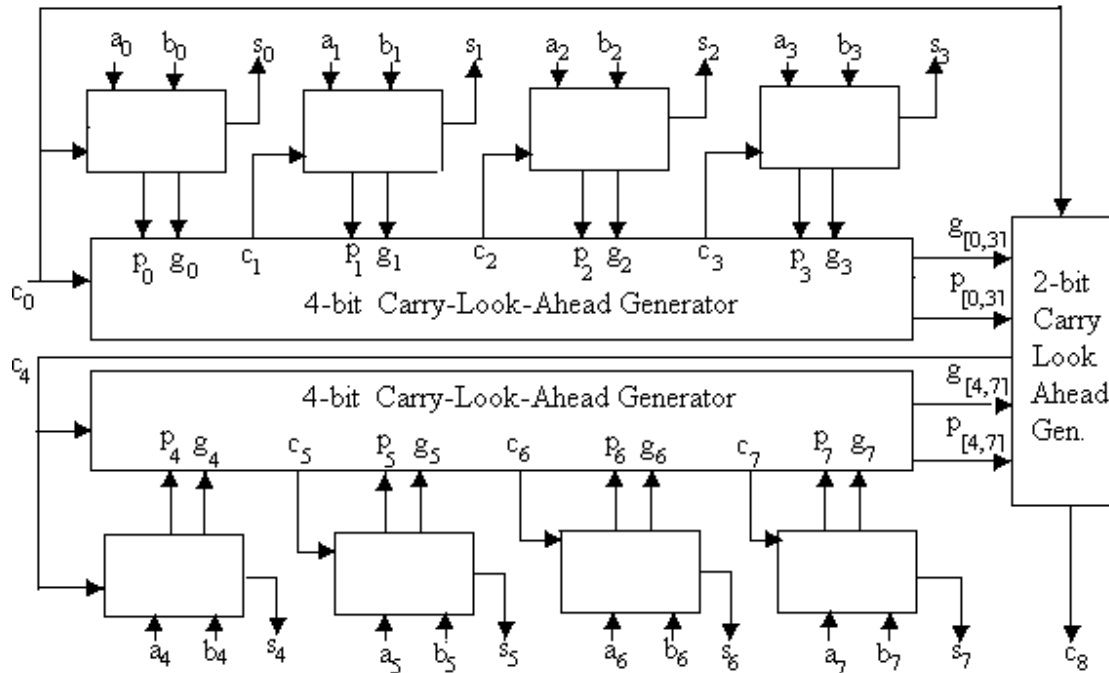


The universal gate
to be used

Instruction	OP_Code	Function
SUB	000	$rd = rs - rt$ (hint: two's complement)
ADD	001	$rd = rs + rt$
BITWISE OR	010	$rd = rs$ (bitwise OR) rt
BITWISE AND	011	$rd = rs$ (bitwise AND) rt
RT ARI. RIGHT SHIFT	100	$rd = \{rt[3], rt[3:1]\}$
RS CIR. LEFT SHIFT	101	$rd = \{rs[2:0], rs[3]\}$
COMPARE LT	110	$rd = \{3'b101, rs < rt\}$
COMPARE EQ	111	$rd = \{3'b111, rs == rt\}$

Verilog Advanced Question 3

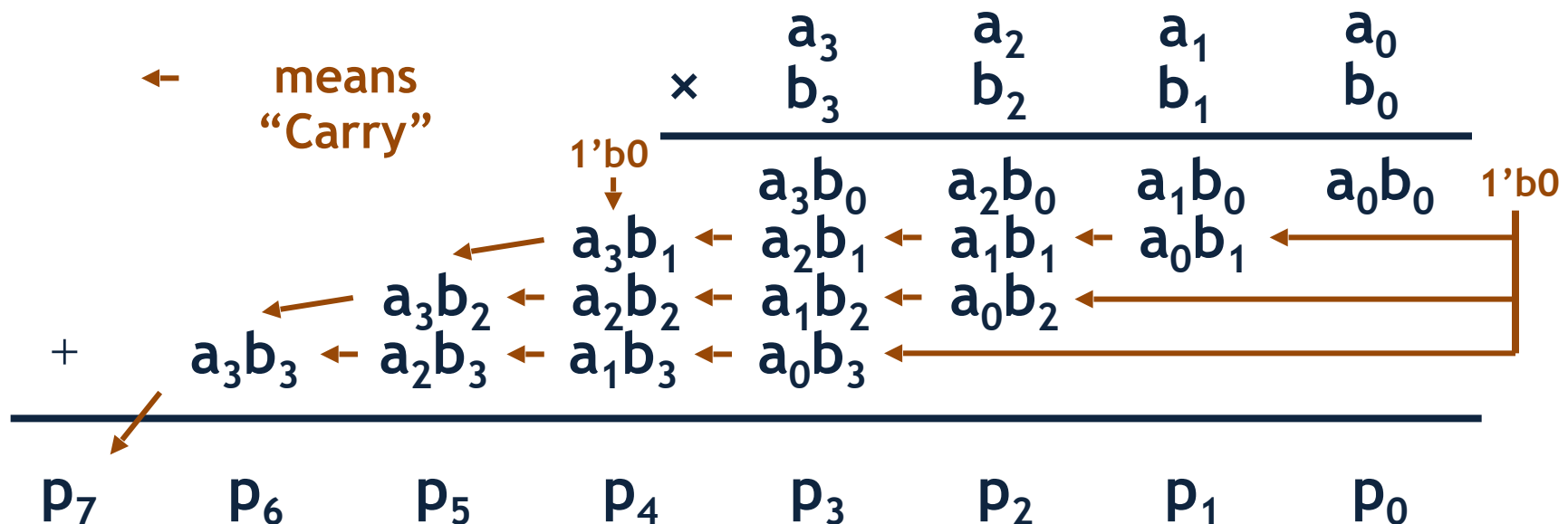
- (Gate Level) 8-bit carry-lookahead (CLA) adder
 - Using **NAND** gates only
 - Please design your CLA using **hierarchical modules**, and follow the figure below
 - Please explain **the circuit of CLA**, the benefits of it, and how it works in your report
 - Please draw your 4-bit CLA generator design in your report
- Go to Wikipedia to check out the details of it
 - https://en.wikipedia.org/wiki/Carry-lookahead_adder



- **Adder inputs:**
 - The operands: $[7:0]$ a (8 bits), $[7:0]$ b (8 bits)
 - The carry in: c_0 (1 bit)
- **Adder outputs:**
 - The sum: $[7:0]$ s (8 bits)
 - The carry out: c_8 (1 bit)

Verilog Advanced Question 4

- (Gate Level) 4-bit multiplier
 - Design a 4-bit unsigned multiplier using **your full adder and half adder**
 - Using **NAND** gates only
 - Please explain how it works
 - Please draw your block diagram using your **adders** and **logic gates**
 - **Hint:** accumulate the partial products using **adders**
- **Inputs:** $a[3:0]$ and $b[3:0]$; **Output:** $p[7:0]$



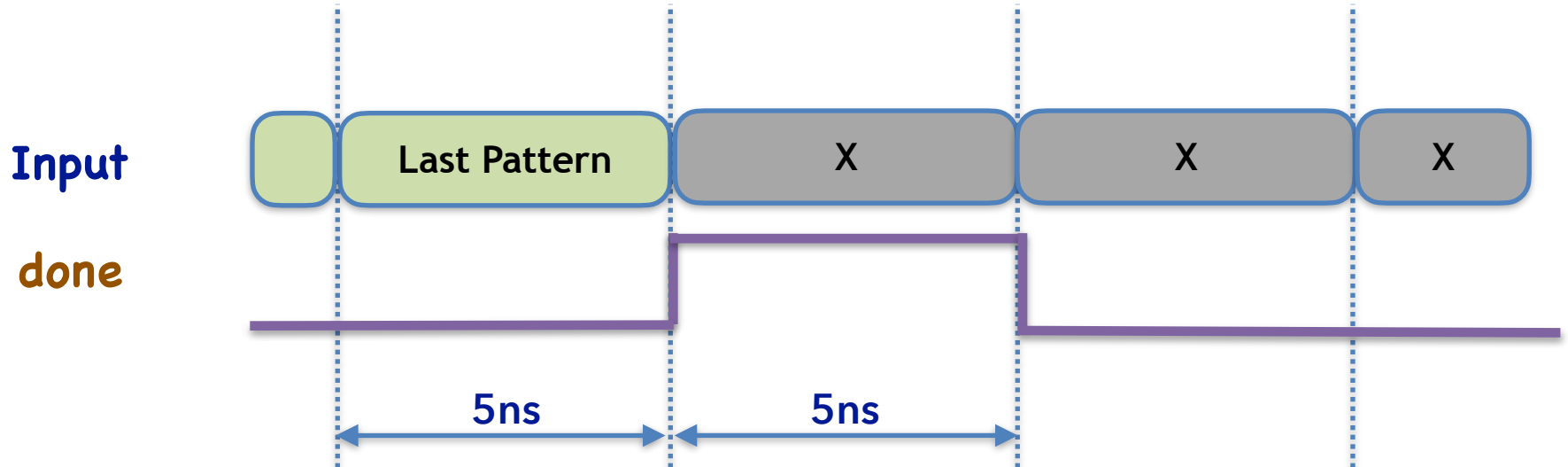
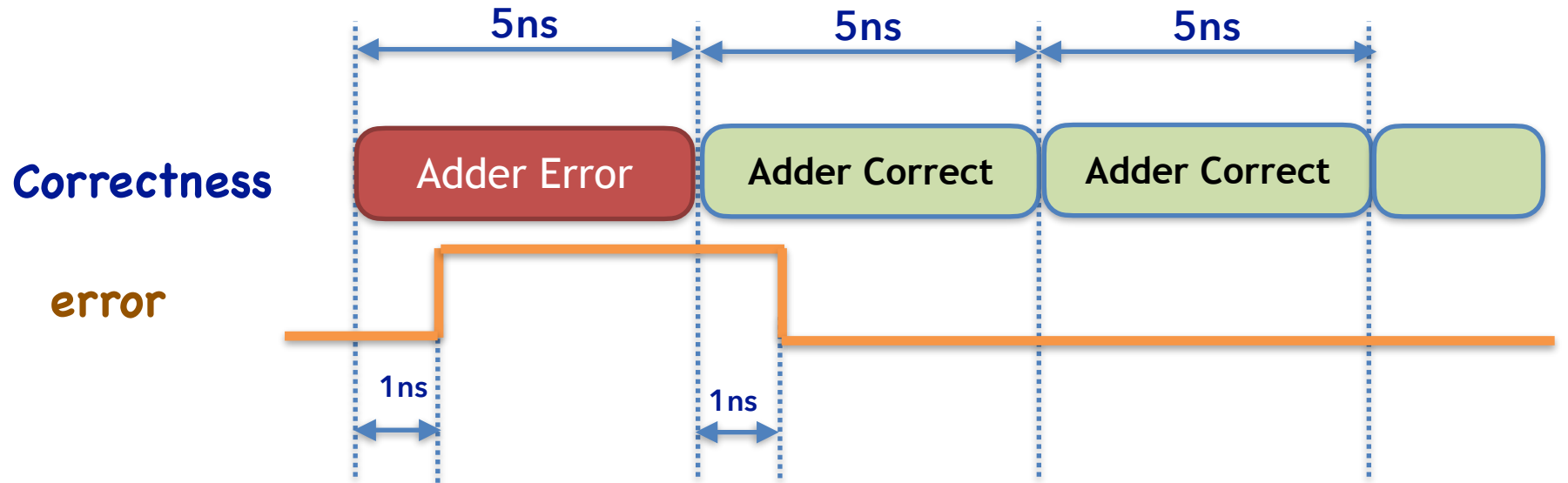
Verilog Advanced Question 5

- (Test Bench) An exhaustive testbench design
 - In this question, please design a testbench for a 4-bit adder circuit
 - We will use faulty designs to check if your test bench can find the intentionally inserted errors
 - We will check whether all the input patterns are covered
- Testbench requirements
 - Please follow the template for your testbench I/Os, which have two additional pins: **error** and **done**.
 - Please change input to the test instance every five nanoseconds.
 - One nanosecond after any input is given, set **error** to 1'b1 if an error is detected. Similarly, if no error is detected, set **error** to 1'b0 one nanosecond after the input is given.
 - Set the values of **done** and **error** to 1'b0 at the beginning of the testbench
 - Set **done** to 1'b1 no earlier than five nanoseconds after the last pattern is provided.

Verilog Advanced Question 5 (con't)

- Important Reminder
 - **Do not** use the **\$finish** system task in your testbench
 - **Do not** include your ripple carry adder design in the submitted file
 - **Do not** remove or alter any **existing code** in the template. However, you are free to add any signals or tasks
 - Violating any of the above mentioned rules can lead to incorrect simulation results or cause the simulation to crash. In these cases, you will get 0 point for this question
 - You can use any Verilog modeling technique when developing your testbench

Verilog Advanced Question 5 (Con't)



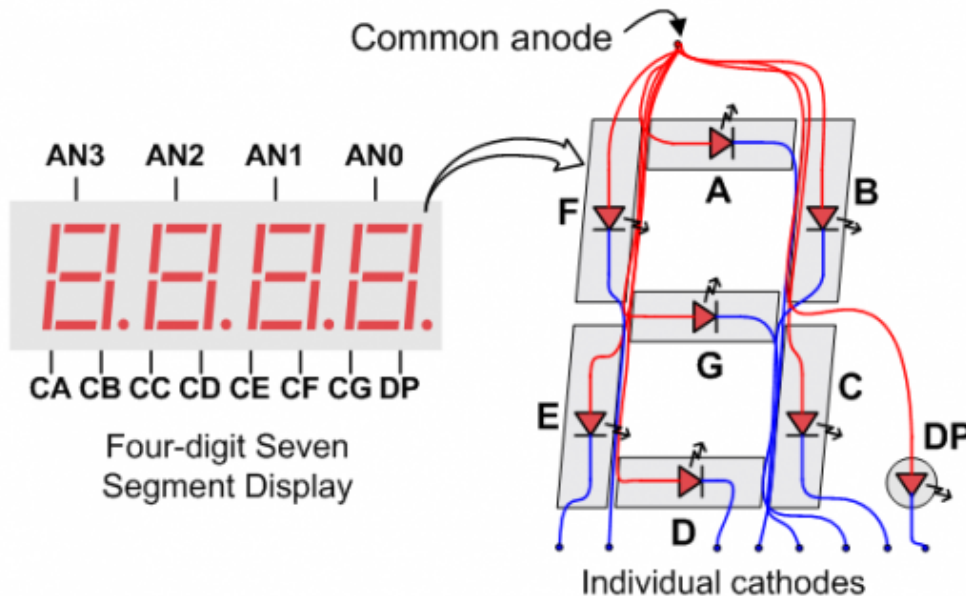
Advanced Questions

- Group assignment
- Verilog questions (due on **9/26/2024 (Thu). 23:59:59.**)
 - **Necessary:** (Gate Level) 8-bit ripple carry adder (RCA)
 - **Necessary:** (Gate Level) Decode and execute
 - **Necessary:** (Gate Level) 8-bit carry-lookahead (CLA) Adder
 - **Necessary:** (Gate Level) 4-bit multiplier
 - **Necessary:** An exhaustive testbench design
- **FPGA demonstration** (due on **9/26/2024 (Thu). In class.**)
 - **Necessary:** (Gate Level) Decode and execute

7-Segment Display Control (1/2)

- Four enable signals for the four display units
 - **AN[3:0]** correspond to **the four digits**
 - **AN** is used to **ENABLE** one of **the four digits**
 - Set one bit of AN[3:0] to **LOW** to illuminate one digit

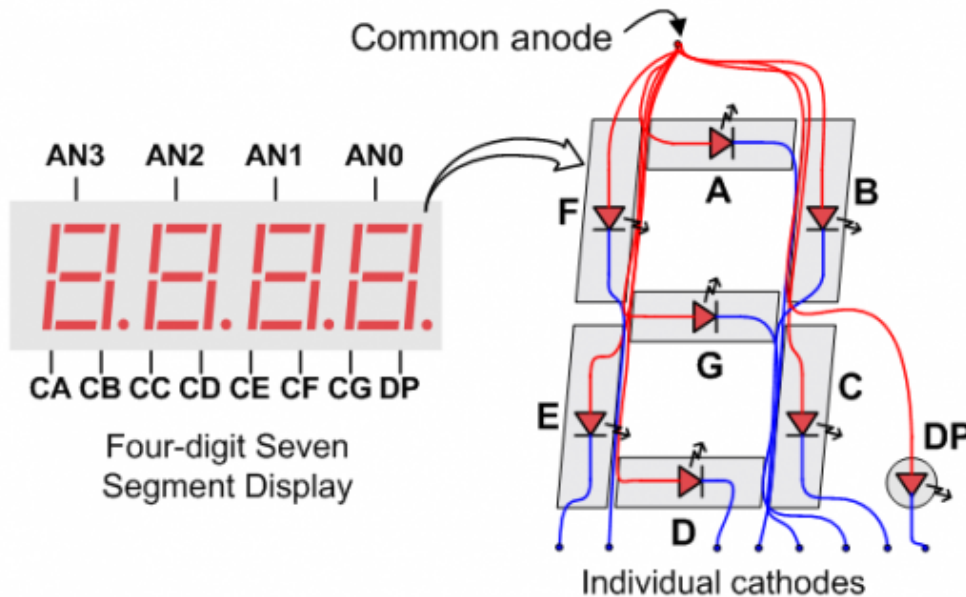
AN Bit	FPGA Pin
3	W4
2	V4
1	U4
0	U2



Digit	AN[3:0]
3	0111
2	1011
1	1101
0	1110

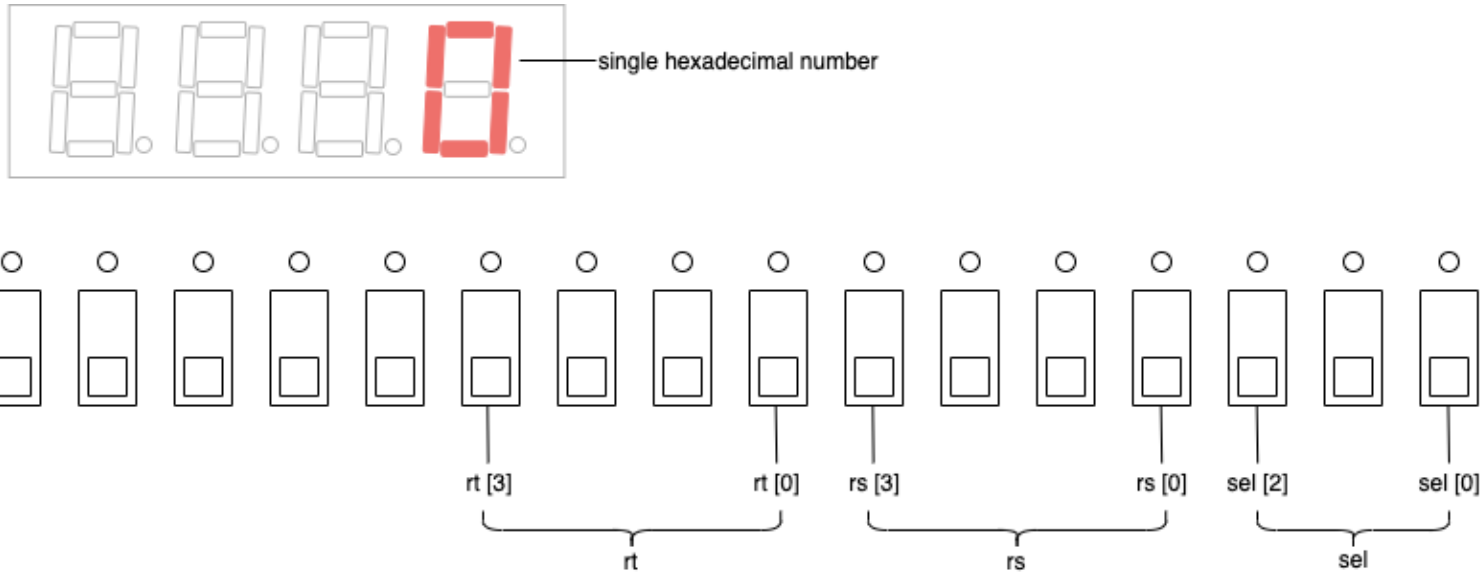
7-Segment Display Control (2/2)

- Eight signals for the segments and dot
 - **seg[0:6]** in the XDC file correspond to **segments A~G** shown below
 - **dp** in the XDC file corresponding to **DP (dot)**
 - To illuminate a segment, set its value to **LOW**



Symbol	FPGA Pin
CA	W7
CB	W6
CC	U8
CD	V8
CE	U5
CF	V5
CG	U7
DP	V7

FPGA Demonstration 1



- (Gate Level) Decode and execute
- Implement the decode and execute module in **Advanced Question 2** onto your FPGA, and represent the output signal **rd** in a **single hexadecimal number**
 - Please assign your inputs/outputs as:
 - **SW[2:0]** stands for '**sel**', **SW[6:3]** stands for '**rs**', **SW[10:7]** stands for '**rt**'
 - Use the **rightmost 7-segment display** to show your **rd**
- You are allowed to use any modeling technique for transforming **rd** to a **single hexadecimal number** on the seven segment display.
 - However, you will get bonus points if you implement it as a gate-level circuit

Thank you for your attention!



*Yosemite Valley view taken at Glacier Point, Yosemite National Park, CA.
This picture is taken by Chun-Yi Lee himself, who is also a fan of photography