IC Design
HW4 Tutorial

Hao-Yu Lan
Advisor：Tzi-Dar Chiueh
2016/12/16

Outline

- Flow & Notification of HW4
- Standard Cell Library
- MAC example
- Pipeline MAC example
- Verification
- Reminder

Flow

- Correctness (50%)
- The product of the minimum half cycle time and the number of transistors (20%)
- First, design a circuit that can pass testbench
- Second, decrease the HALF_CYCLE in testbench_ppl.v until it failed (CYCLE < critical path). The minimum half cycle time is the minimum HALF_CYCLE for successfully pass testbench.

Flow

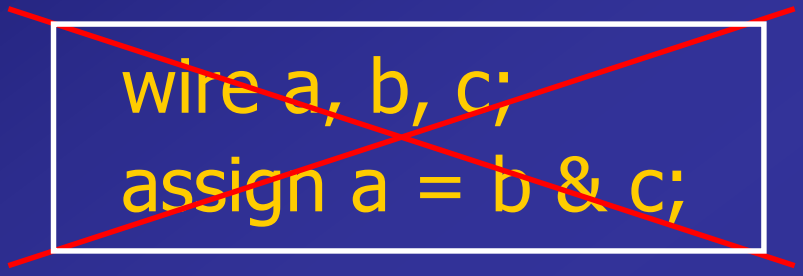
- Third, Find the number of transistors in your design by hand.

```
module AN3(Z,A,B,C,number);
    output Z;
    input A,B,C;
    parameter size = 10'd50;
    output [size:0] number;
    wire   [size:0] number;
    assign number=11'd8;
```

- Finally, modify you design the get better product of the minimum half cycle time and the number of transistors. Trade-off between them. Ex. Try different adders, carry-skip, carry-lookahead, etc.

Notification of HW4!!

- In this HW, all the logic operation MUST consist of standard cell (defined in lib.v). You can NOT use logic operators.



→ Behavioral Modeling

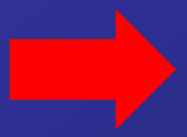
```
wire a, b, c;
AN2 an(a, b, c);
```

→ Structural Modeling

Notification of HW4!!

- Do NOT change any module name and port name in SQC_ppl.v, just modifiy the module description, otherwise you can't pass testbench.

```
module DOQE_ppl(clk,rst,A,B,C,D,DOQE_ppl_num);
// DO NOT CHANGE !
    input       clk;
    input       rst;
    input       [3:0]   A;
    input       [6:0]   B;
    input       [5:0]   C;

    output      [14:0]  D;
    output      [50:0]  DOQE_ppl_num;
```



Don't Change

- Use FD2 (positive edge) module for flip flop.

Standard Cell Library

Standard Cell Library (lib.v)

- Choose what you need
- Compose your circuit according to I/O connections

- IV // not
- AN3
- AN4
- AN2
- EN // xnor
- EN3
- EO // xor
- EO3
- FA1 // full adder
- FD1 // negative edge DFF
- FD2 // positive edge DFF

- ND2 // nand
- ND3
- ND4
- NR2 // nor
- NR3
- OR2 // or
- OR3
- OR4
- HA1 // half adder
- MUX21H // 2-to-1 MUX

Specify Path Delay and Area

A → Z 0.275
B → Z 0.275
C → Z 0.275

```
module AN3(Z,A,B,C,number);

   output Z;
   input A,B,C;
  parameter size = 10'd50;
 output [size:0] number;
 wire  [size:0] number;
 assign number=11'd8;
   // netlist
   and g1(Z,A,B,C);
   // specify block, declare local
   // timing constant
   specify
      // delay parameters
      specparam Tp_A_Z = 0.275;
      specparam Tp_B_Z = 0.275;
      specparam Tp_C_Z = 0.275;
      // path delay (full connection)
      ( A *> Z ) = ( Tp_A_Z );
      ( B *> Z ) = ( Tp_B_Z );
      ( C *> Z ) = ( Tp_C_Z );
   endspecify

endmodule
```

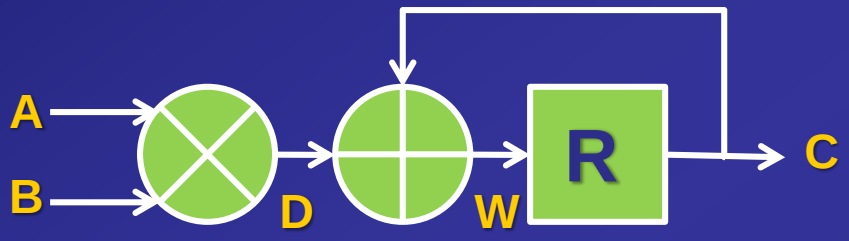
Number of Transistors

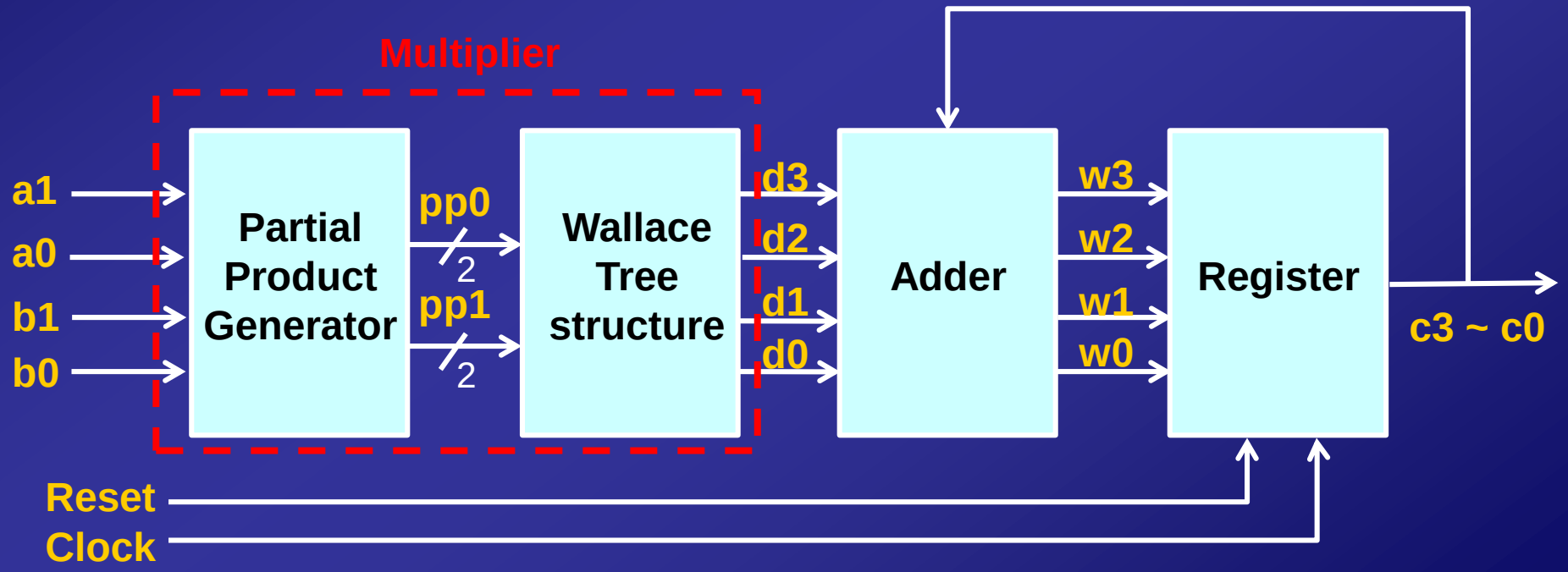
```
module Reg3(Q,DD,CLK,RESET,Reg3_num);
   output [2:0] Q;
   input [2:0] DD;
   input CLK,RESET;

   output wire [50:0]  Reg3_num;
   wire [50:0] FD_num0,FD_num1,FD_num2;

    assign Reg3_num=
                FD_num0+FD_num1+FD_num2;

   FD2 fd0(Q[0],DD[0],CLK,RESET,FD_num0);
   FD2 fd1(Q[1],DD[1],CLK,RESET,FD_num1);
   FD2 fd2(Q[2],DD[2],CLK,RESET,FD_num2);

endmodule
```

```
module DOQE_ppl(clk,rst,
                        A,B,C,D,DOQE_ppl_num);
            input                   clk;
            input                   rst;
            input                   [3:0]       A;
            input                   [6:0]       B;
            input   [5:0] C;
            output      [14:0]D;
             output wire [50:0]  DOQE_ppl_num;

    wire [50:0] num0,num1;
    assign DOQE_ppl_num=num0+num1;

    wire [2:0] A_reg,B_reg;
    Reg3 rr0(A_reg,A[2:0],clk,rst,num0);
    Reg3 rr1(B_reg,B[2:0],clk,rst,num1);

endmodule
```

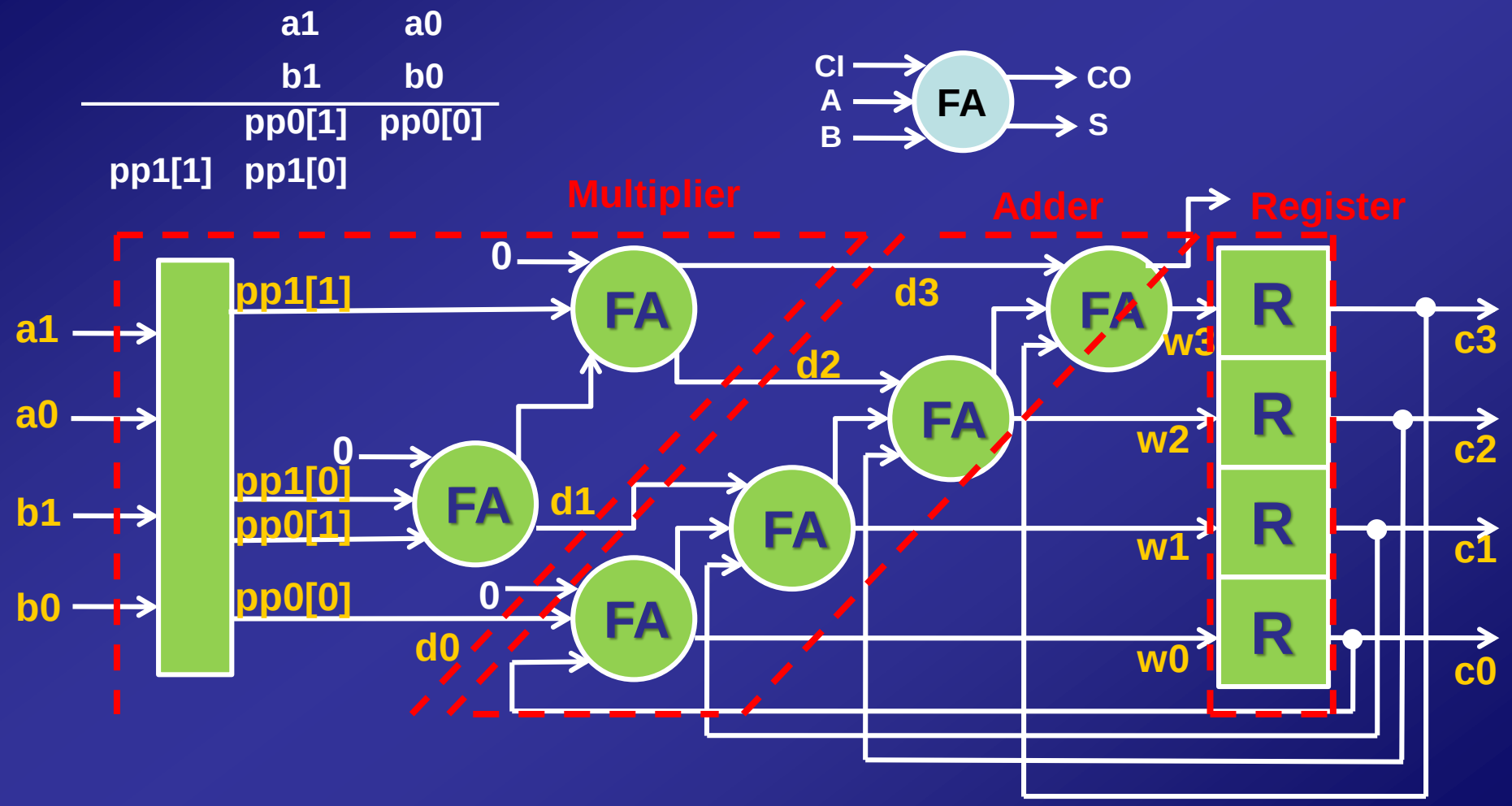
MAC(multiplier–accumulator) Example

2-bit MAC Example (1/2)





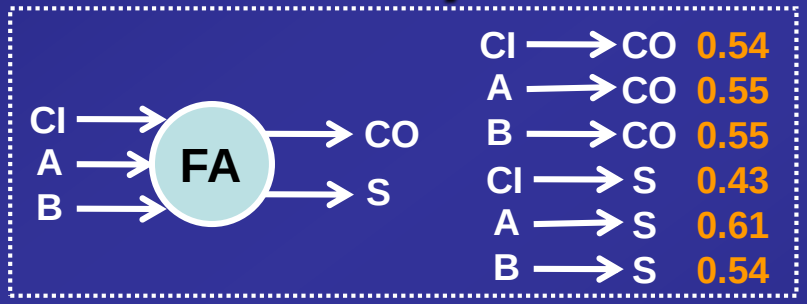
2-bit MAC Example (2/2)

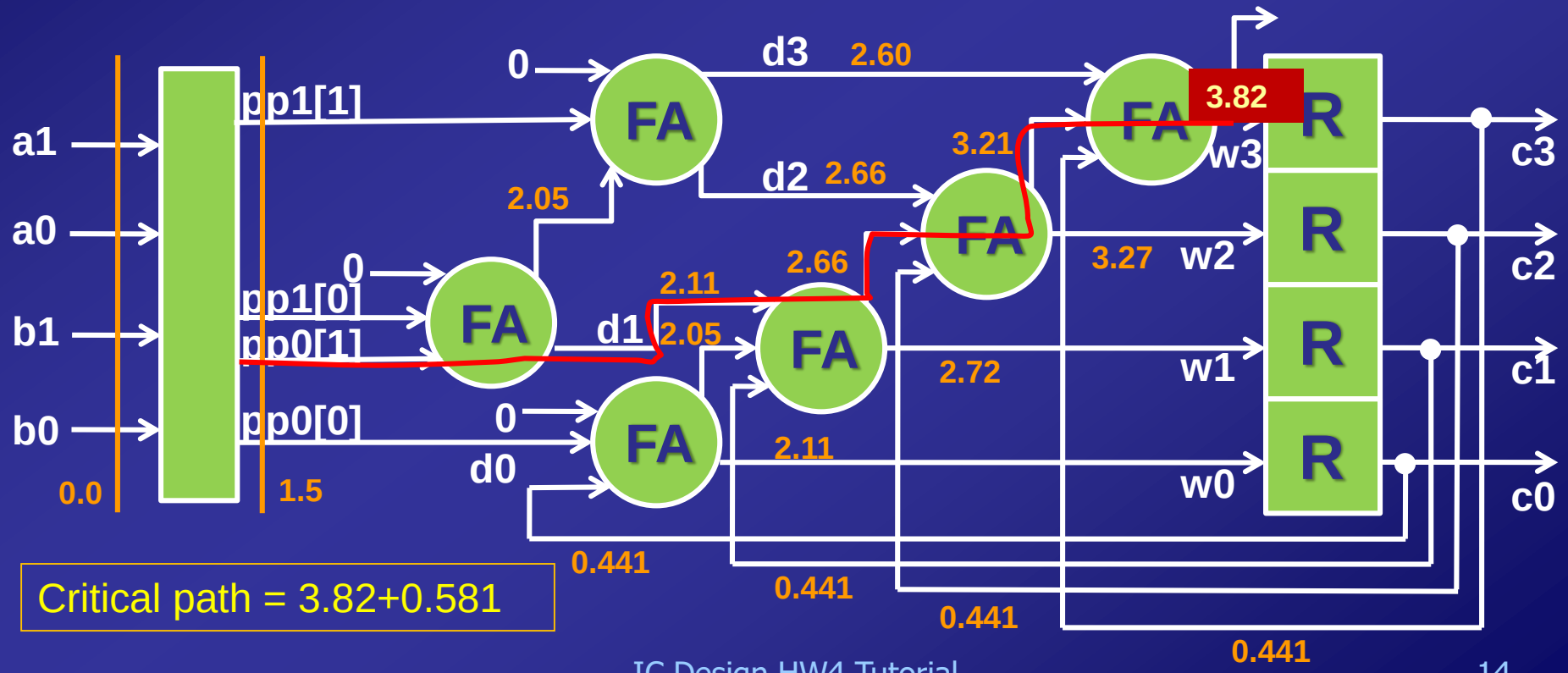


Timing Path Calculation By Hand

Register delay time 0.441
Register setup time 0.581

Booth encoder delay 1.5 (assume)





Critical path = 3.82+0.581

Timing Path Calculation By Simulation

- Modify HALF_CYCLE in testbench.v to test your critical path

```
`resetall
`timescale 1ns/1ps
`define     HALF_CYCLE          3.924
`define     CYCLE               (`HALF_CYCLE*2)
`define     REG_SETUP_TIME      0.581
`define     REG_DELAY           0.441
```

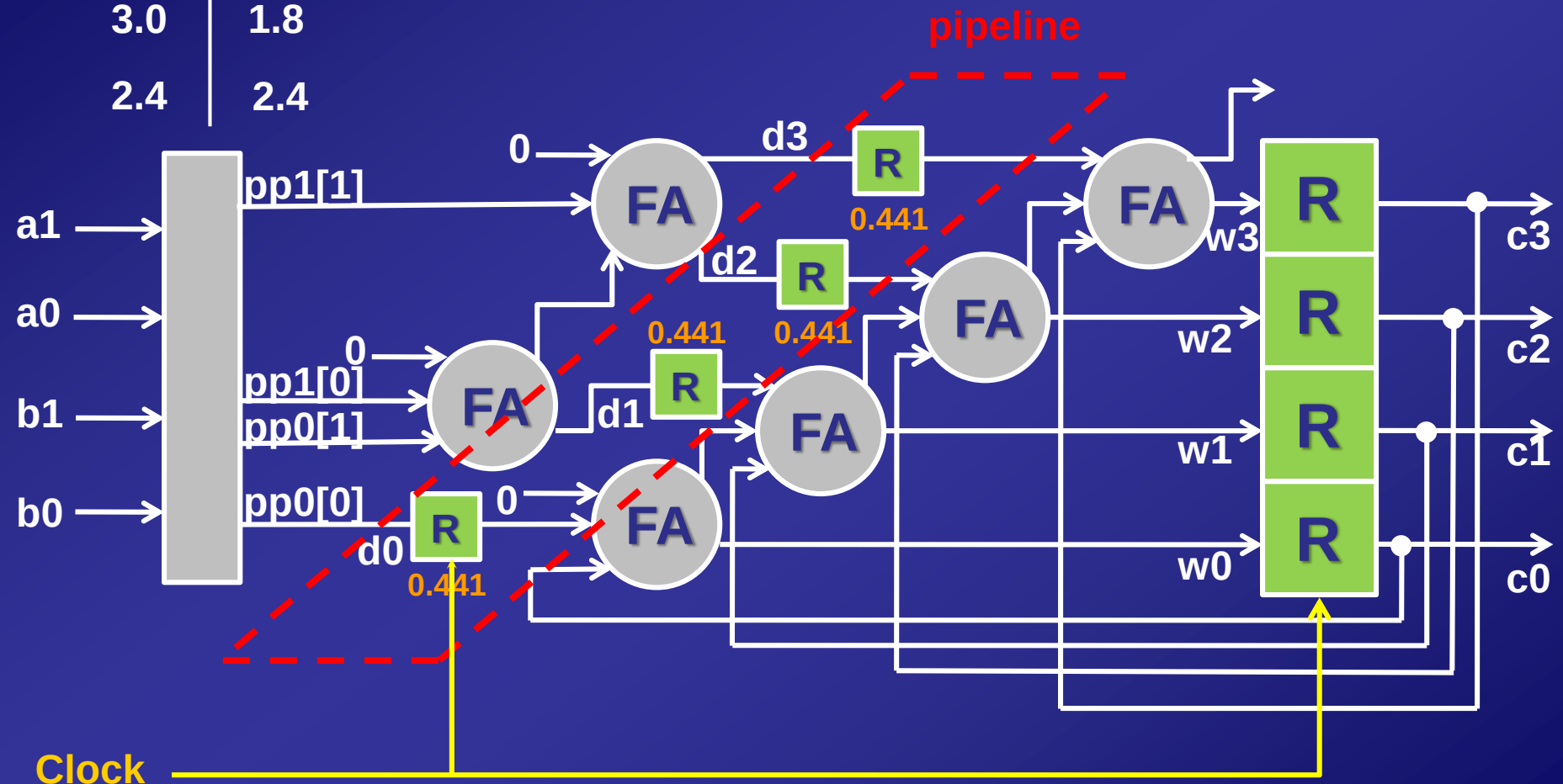
- First, loosen the clock cycle when you're checking your circuit logic. Once the logic is correct, start to shorten the clock period to find the critical path.

Pipeline MAC Example

Pipelined Structure

Critical path = 4.8

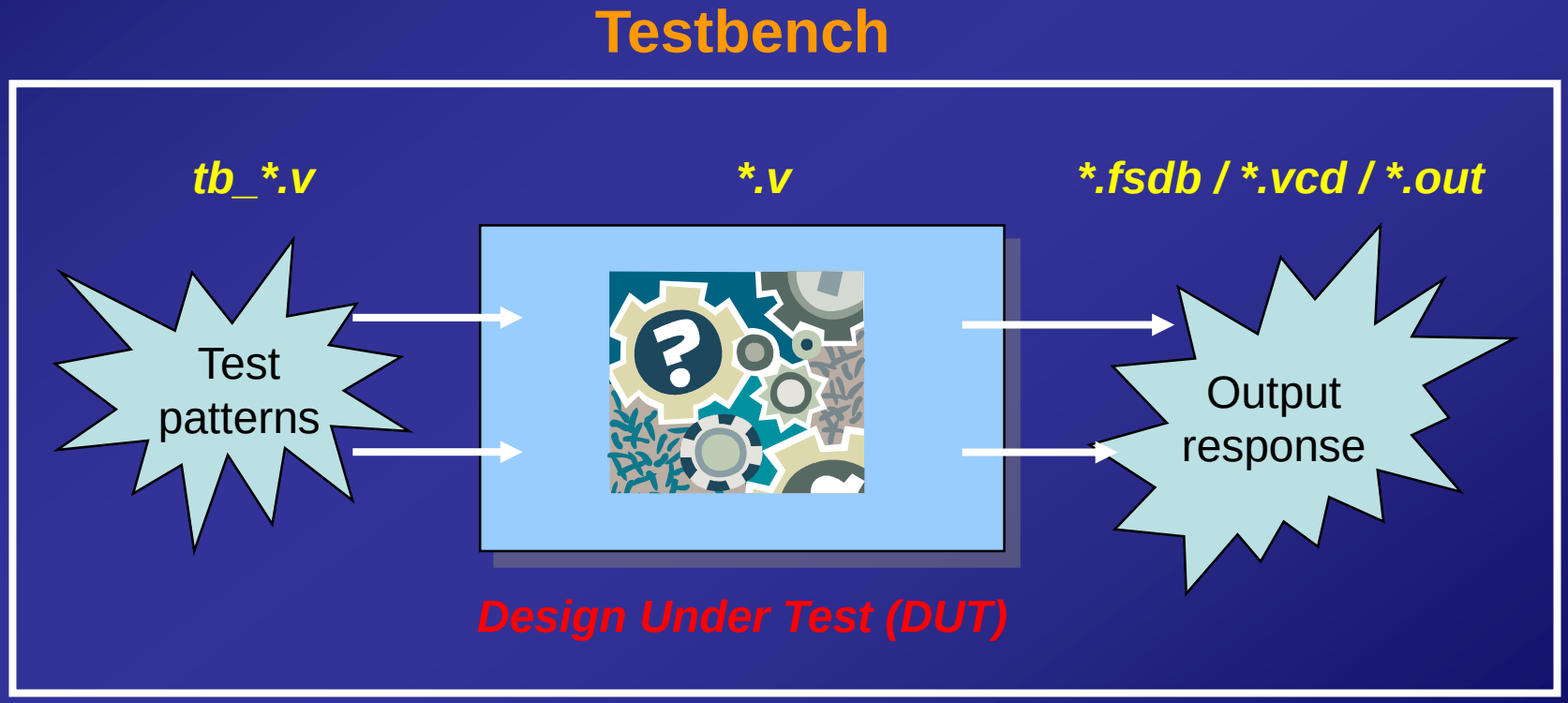
3.8	1.0
3.0	1.8
2.4	2.4



Verification

Test and Verify Your Circuit

- By applying input patterns and observing output responses



Verilog Testbench

- Check testbench_ppl.v

```
`timescale 1ns/1ps
`define   HALF_CYCLE        3.924
`define   CYCLE             (`HALF_CYCLE*2)
`define   REG_SETUP_TIME    0.581
`define   REG_DELAY         0.441
```

- Modify HALF_CYCLE to test your critical path

Compile and debug

- Source
  - source /usr/cadence/cshrc
  - source /usr/spring_soft/CIC/verdi.cshrc
- Include the testbench & lib.v files to run simulation
  - ncverilog +access+r testbench_ppl.v DOQE_ppl.v lib.v

Compile and debug(2/2)

```
        Writing initial simulation snapshot: worklib.test:v
Loading snapshot worklib.test:v ..................... Done
ncsim> source /usr/cadence/IUS/cur/tools/inca/files/ncsimrc
ncsim> run
$readmem error: invalid syntax in file "A.dat" at line 100
$readmem error: invalid syntax in file "B.dat" at line 100
$readmem error: invalid syntax in file "C.dat" at line 100
$readmem error: invalid syntax in file "answer.dat" at line 100
Congratulations! Your score is 50!

Number of transistors:         6969.

Simulation complete via $finish(1) at time 1038978 PS + 0
./testbench_ppl.v:82                $finish;
ncsim> exit
```

Reminder (1/2)

- Loosen the clock cycle when you're checking your circuit logic.
- Once the logic is correct, start to shorten the clock period to find the critical path.
- Use basic gates provided in lib.v to design your circuit. No behavior level code will be accepted.

Reminder (2/2)

- Due on 2017/01/06 13:20
- For any further questions , contact TAs !
  - 藍浩佑 r04943028@ntu.edu.tw
- To know more about Verilog, refer to
  - http://www.ece.umd.edu/courses/enee359a.S2008/verilog_tutorial.pdf