IC Design
HW4 Tutorial

Speaker : Ming-Hang Hsieh
Professor: Prof. Tzi-Dar Chiueh
Date : 2019/12/20

NTU MicroSystem Research Lab.

Outline

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

Notification of HW4

- Correctness Score (40%)
  - Design a circuit that can pass testbench
- Performance Score (30%)
  - Ranking according to (number of transistors) × (execution time)

Percentage of passing students	Performance Score
If your ranking > 90 %	30
80% ~ 90%	27
70% ~ 80%	24
60% ~ 70%	21
50% ~ 60%	18
40% ~ 50%	15
30% ~ 40%	12
20% ~ 30%	9
10% ~ 20%	6
0% ~ 10%	3
Using operands, not standard cell logic	0
Correctness failed	0
Plagiarism	0

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.

~~
```
wire a, b, c;
assign a = b & c;
```
~~

→ Behavioral Modeling

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

→ Structural Modeling

Notification of HW4

- Use FD2 (positive edge) module for flip flop.
- DO NOT change any module name and port name in COA.v

```
module COA(clk, rst_n, A, B, C, D, valid_in, valid_out, COA_num);

/* ---------------------------- DO NOT CHANGE ! ---------------------------- */
    input clk, rst_n;

    // Input/Output Data
    input       [5-1:0] A;
    input       [5-1:0] B;
    input       [5-1:0] C;
    output reg [10-1:0] D;

    // Handshake signal
    input  valid_in;
    output valid_out;

    // Area count
    output [50:0] COA_num;
/* ---------------------------- DO NOT CHANGE ! ---------------------------- */
```

Don't Change

Notification of HW4

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

```
`resetall
`timescale 1ns/1ps

`define HALF_CYCLE        5.0
`define CYCLE             (`HALF_CYCLE*2)
`define N                 1000
`define MAX_SIM_CYCLE     10000
`define REG_DELAY         0.441
`define DEBUG             0
```

- 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.

Debug mode

- Modify DEBUG in testbench.v to enable debugging
- Modify N to avoid terminal printing too much information.

```
`resetall
`timescale 1ns/1ps

`define HALF_CYCLE      5.0
`define CYCLE           (`HALF_CYCLE*2)
`define N               1000
`define MAX_SIM_CYCLE   10000
`define REG_DELAY       0.441
`define DEBUG           0
```

Debug mode

```
[RECEIVE]     0,  Waiting valid_out...

[RECEIVE]     1,  Waiting valid_out...

[RECEIVE]     2,  Waiting valid_out...
[SEND   ]     2,  index:     0,          A:   29, B:   30, C:    1

[RECEIVE]     3,  Waiting valid_out...
[SEND   ]     3,  index:     1,          A:    4, B:   28, C:    7

[RECEIVE]     4,  Waiting valid_out...
[SEND   ]     4,  index:     2,          A:   20, B:   13, C:   23

[RECEIVE]     5,  Waiting valid_out...
[SEND   ]     5,  index:     3,          A:    4, B:    4, C:   24

[RECEIVE]     6,  index:     0, D:   30
[SEND   ]     6,  index:     4,          A:    9, B:   29, C:    8

[RECEIVE]     7,  index:     1, D:  408
[SEND   ]     7,  index:     5,          A:   30, B:    6, C:   30

[RECEIVE]     8,  index:     2, D:  149
[SEND   ]     8,  index:     6,          A:    5, B:    2, C:    5
```

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 FD2(Q,D,CLK,RESET,number);
    output Q;
    input D,CLK,RESET;
    reg Q,realD,realRESET;
    parameter size = 10'd50;
    output [size:0] number;
    wire   [size:0] number;
    assign number=11'd27;
```

Number of Transistors

```
module COA(clk, rst_n, A, B, C, D, valid_in, valid_out, COA_num);

/* ---------------------------- DO NOT CHANGE ! ---------------------------- */
    input clk, rst_n;

    // Input/Output Data
    input       [5-1:0] A;
    input       [5-1:0] B;
    input       [5-1:0] C;
    output reg [10-1:0] D;

    // Handshake signal
    input  valid_in;
    output valid_out;

    // Area count
    output [50:0] COA_num;
/* ---------------------------- DO NOT CHANGE ! ---------------------------- */


/* -------------------- Write your code from this line -------------------- */


/* -------------------- Add up your number of transistors ------------------- */
    assign COA_num = ;

endmodule
```

Standard Cell Library (lib.v)

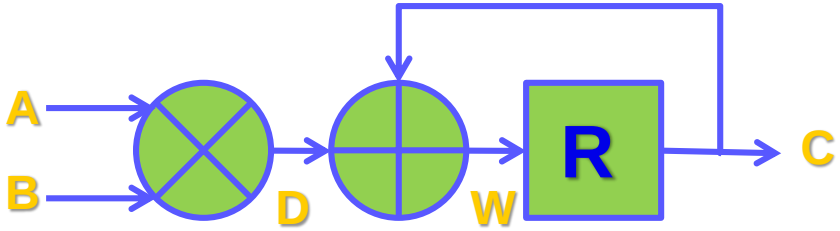
- Choose what you need
- Compose your circuit according to I/O connections

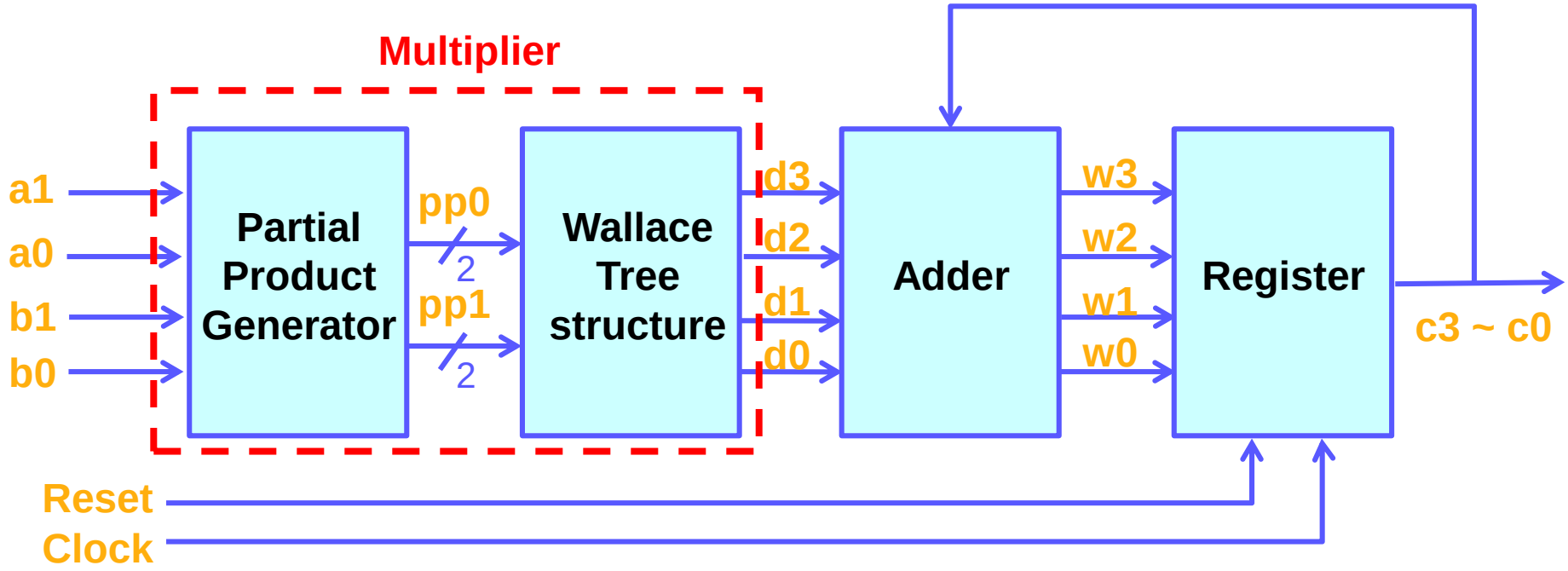
- IV // not
- AN3
- AN4
- AN2
- EN // xnor
- EN3
- EO // xor
- EO3
- FA1 // full adder
- FD1 // negative edge DFF
- FD2 // positive edge DFF

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

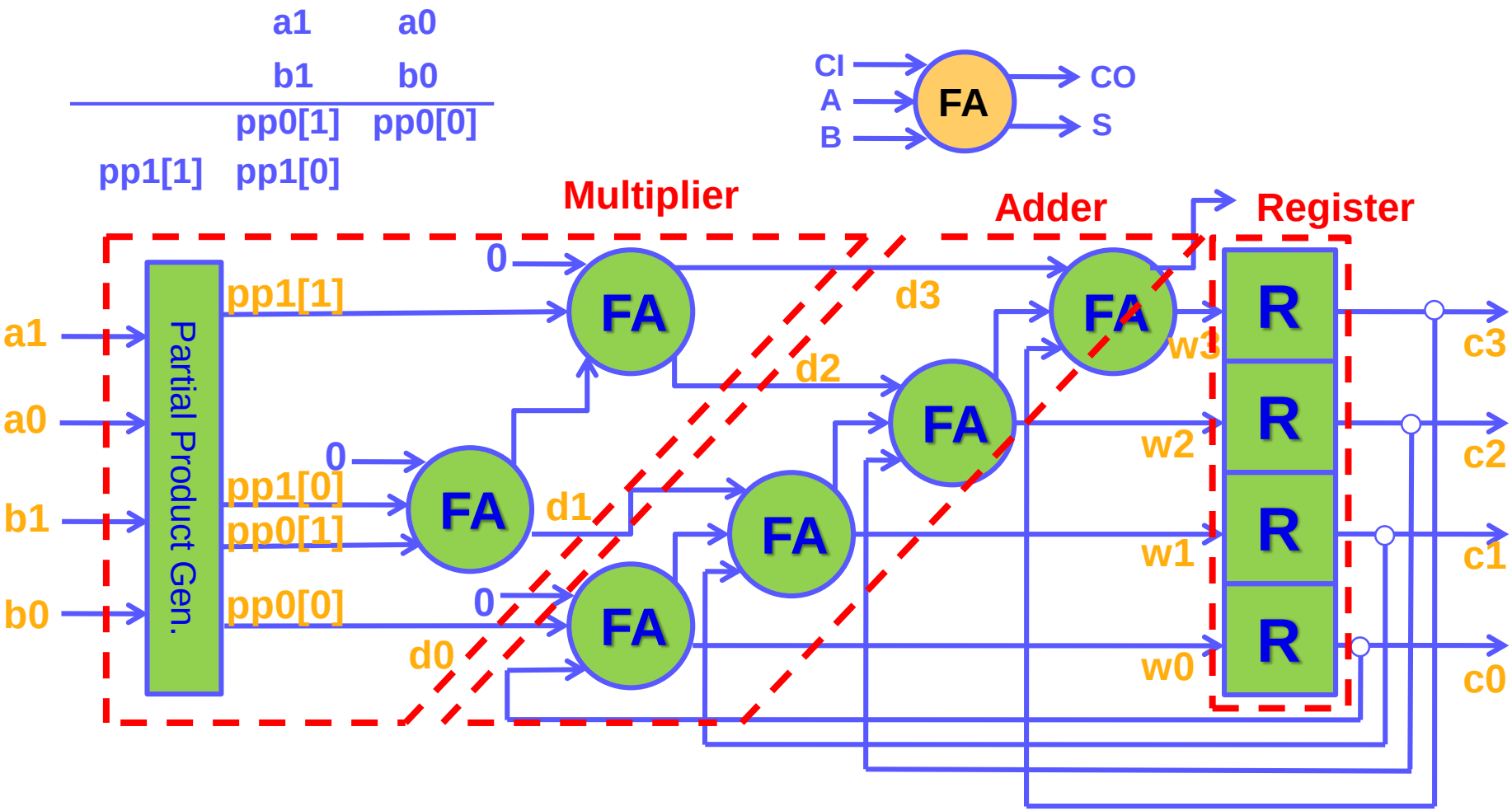
MAC(multiplier-accumulator) Example

2-bit MAC Example (1/2)





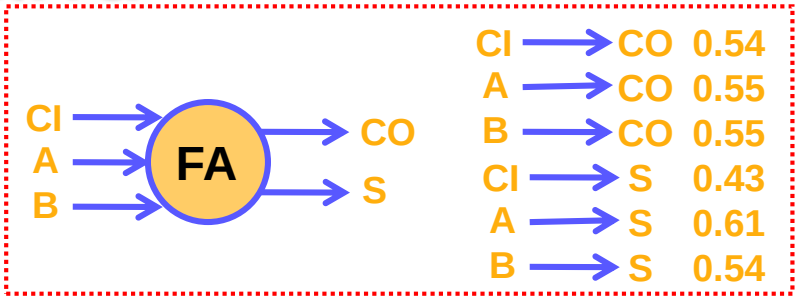
2-bit MAC Example (2/2)

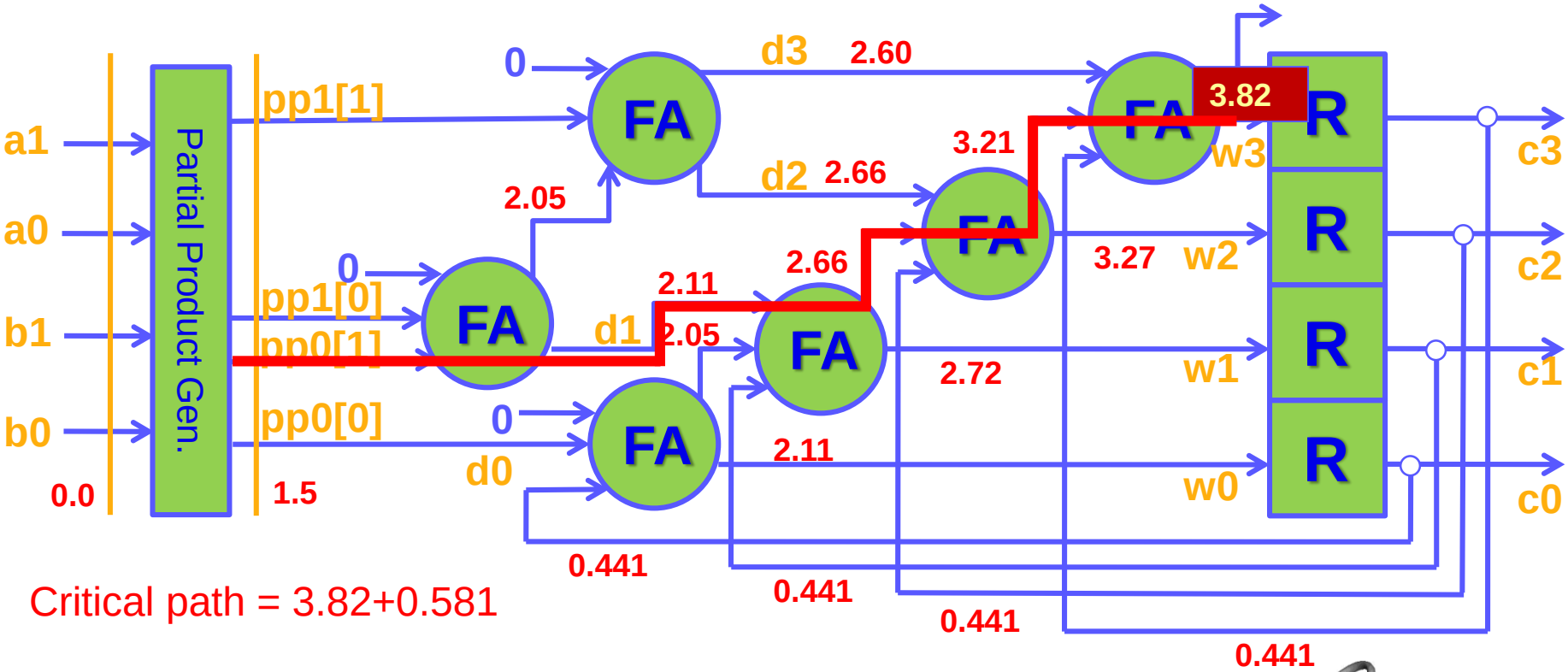


Timing Path Calculation By Hand

Register delay time 0.441
Register setup time 0.581

Partial Product Gen. 1.5 (assume)

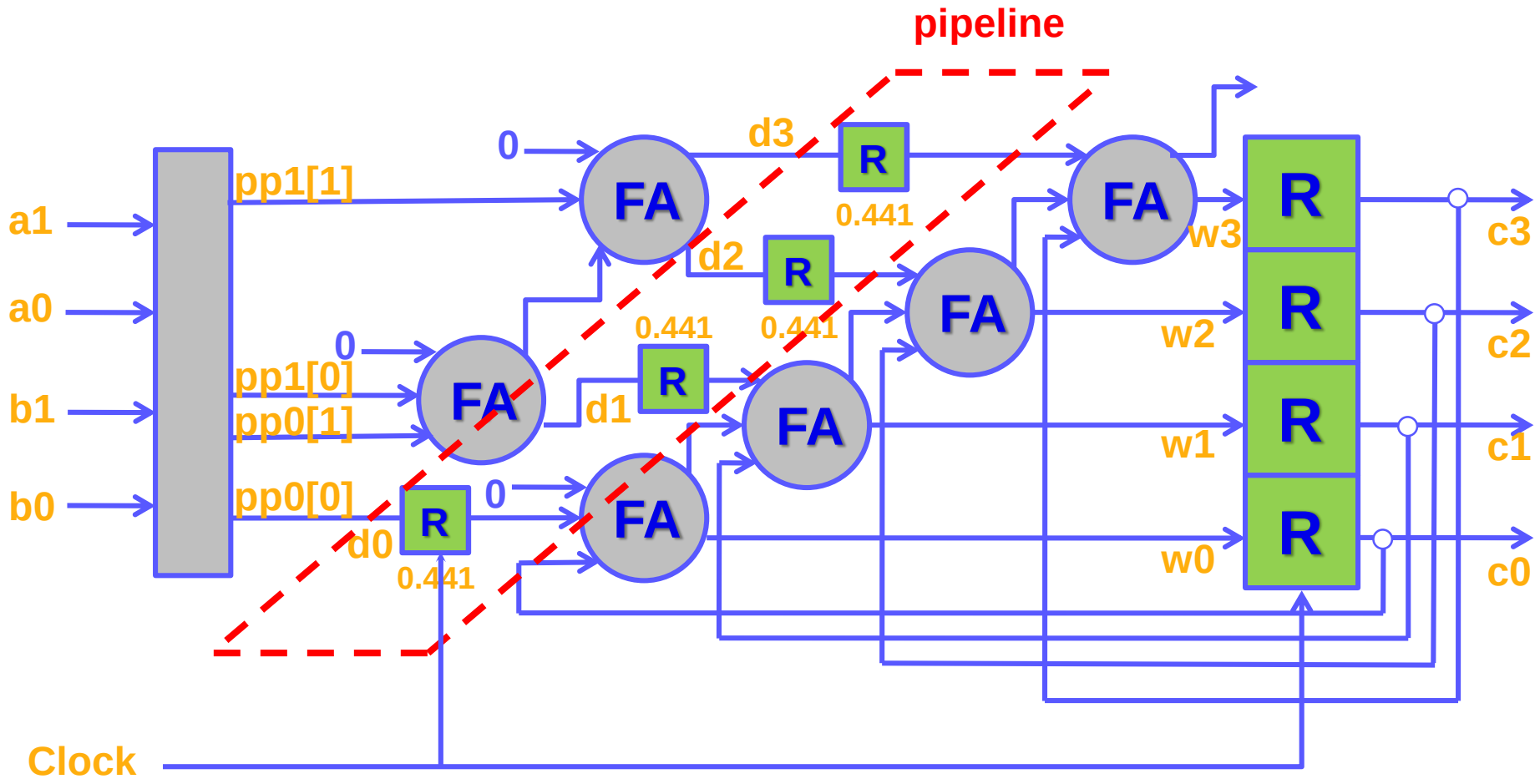




Critical path = 3.82+0.581

Pipeline MAC Example

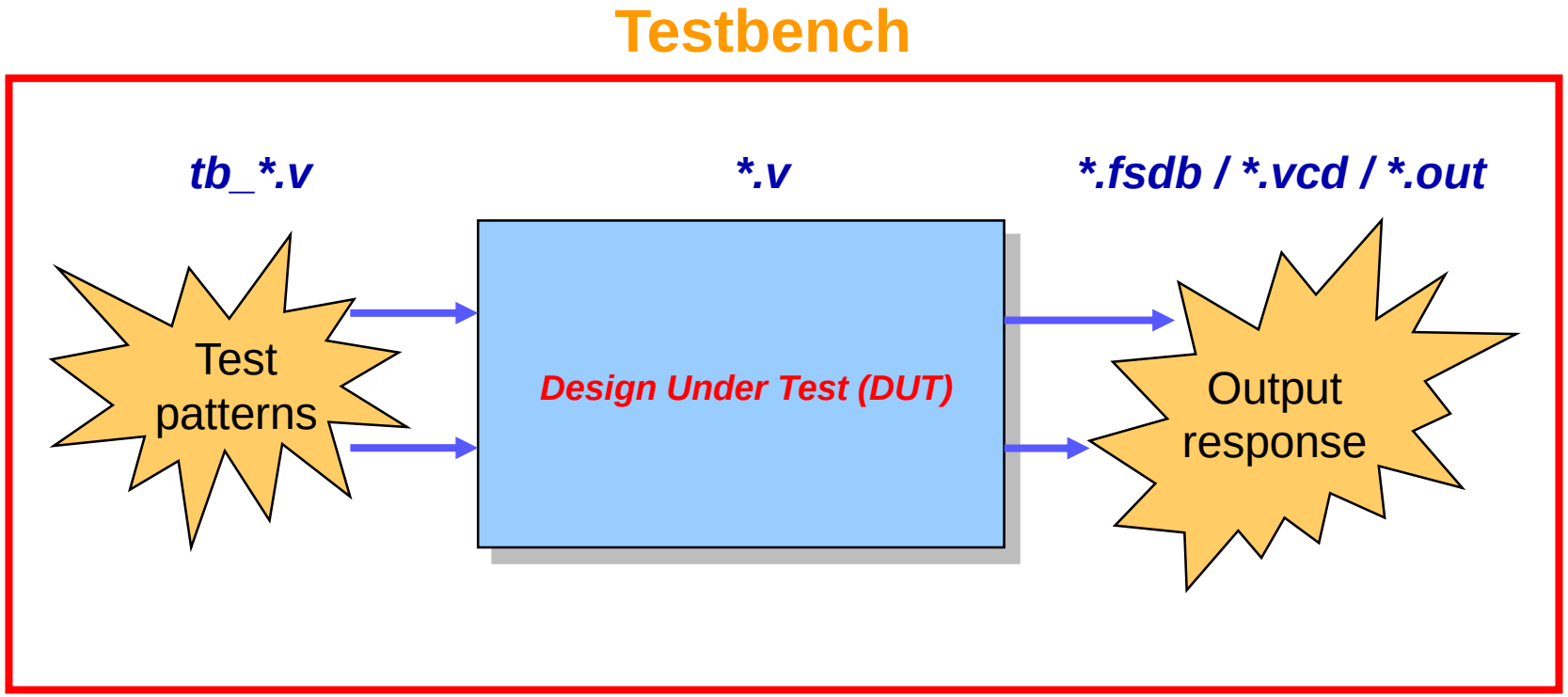
Pipelined Structure



Verification

Test and Verify Your Circuit

- By applying input patterns and observing output responses



Simulation

- Source
  - source /usr/cadence/cshrc
- Include the testbench & lib.v files to run simulation
  - ncverilog +access+r testbench.v COA.v lib.v

```
---- All passed. Congratulations! ----

Your score is    40.

=====================================
Summary:
-------------------------------------
Cycle            :            10.00 ns
Number of Cycles:           1006
Execution Time   :          10060.00 ns
Area             :           5798
Area x Time      :       58327880.00
=====================================
```

```
[ERROR ] index:     7, D:   173, Answer:   165
[ERROR ] index:     8, D:    20, Answer:    27
[ERROR ] index:    10, D:   326, Answer:    11
[ERROR ] index:    11, D:   149, Answer:   202
There are   950 errors.

Your score is    2.

=====================================
Summary:
-------------------------------------
Cycle            :            10.00 ns
Number of Cycles:           1006
Execution Time   :          10060.00 ns
Area             :           5798
Area x Time      :       58327880.00
=====================================
```

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 2020/01/03 14:00
- For any further questions , contact TAs !
  - 謝明航 r07943012@ntu.edu.tw
- To know more about Verilog, refer to
  - http://www.ece.umd.edu/courses/enee359a.S2008/verilog_tutorial.pdf