

HW#3 Tianpu Zhao

#1 (a) CLA equations-

each $p_i = a_i \oplus b_i$ (takes 3ns), $g_i = a_i b_i$ (takes 1ns)

$$C_1 = g_c + p_a C_a$$

g_c takes shorter time than $p_a C_a$

$p_a C_a$ takes 3 (from p_a) + 1 (from OR) = 4ns

AND takes 1ns $\rightarrow C_1$ takes 5ns

$$C_2 = g_1 + p_r g_c + p_r p_a C_a$$

$p_r p_a C_a$ takes 2 (2x AND) + 3 (from p_r , which are parallel) = 5ns

$p_r g_c$ takes 3 (p_r) + 1 (AND) = 4ns

g_1 takes 1ns

evaluating $(g_1 + p_r g_c)$ takes the same time as $p_r p_a C_a$ so the largest time is 5+1=6ns, 1 count for the OR in $(g_1 + p_r g_c) + p_r p_a C_a$ [here I assume while computing $p_r p_a C_a$, we compute $g_1 + p_r g_c$ at the same time]

$$C_3 = g_1 + p_r g_c + p_r p_a C_a + p_2 p_1 p_a C_a$$

likewise the slowest operation is $p_2 p_1 p_a C_a + (\text{the rest})$, taking 6 ($p_2 p_1 p_a C_a$) + 1 (OR) = 7ns

Likewise C_4 takes 8ns

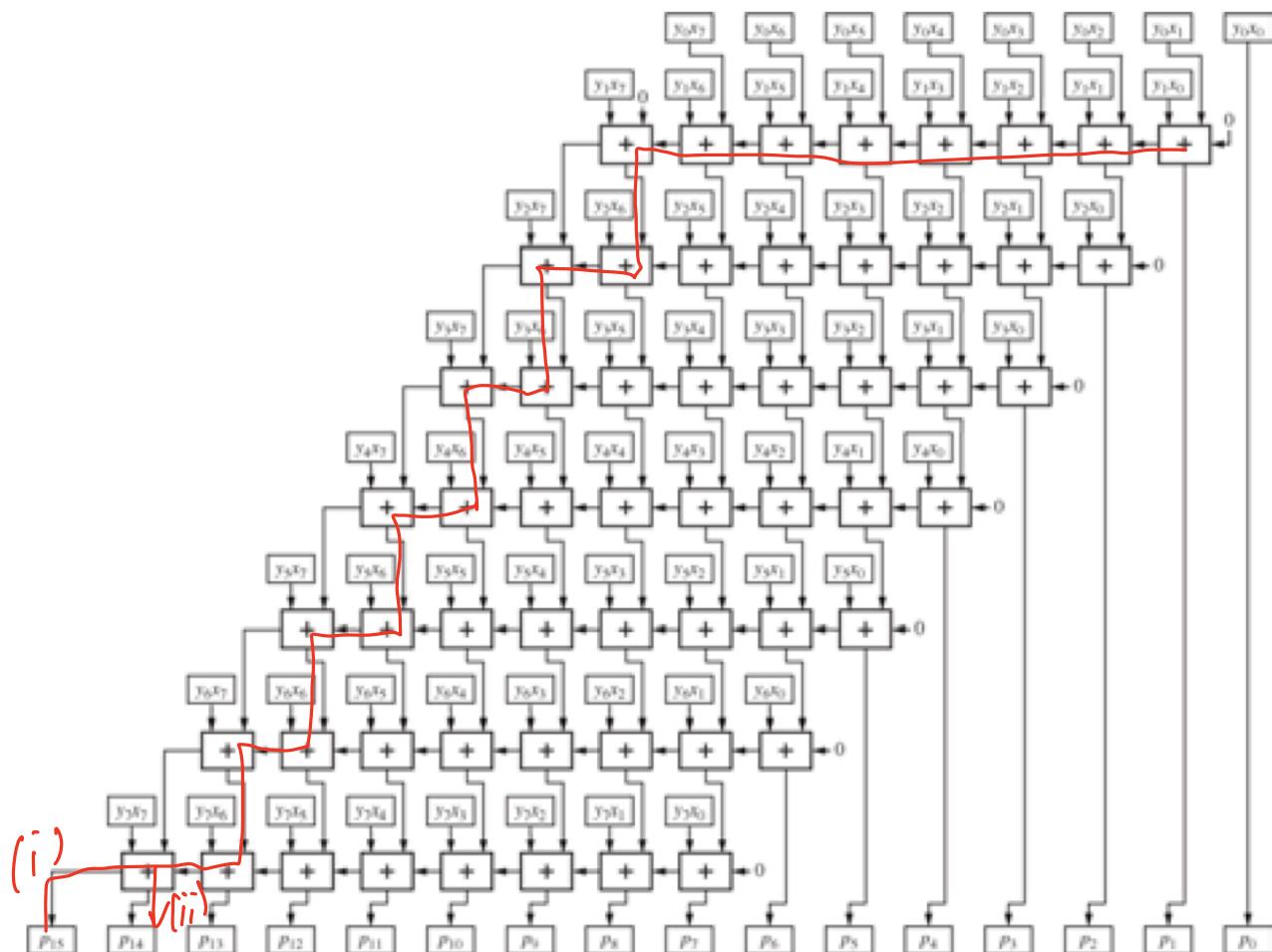
Worst-case delay from any input to carry output is the evaluation of C_4 , taking 8ns.

The sum: $S_i = P_i \oplus C_i$

$$\text{delay} = \max \{ \text{time}(P_i), \text{time}(C_i) \} + 3\text{ns} (\text{from } \oplus)$$
$$= 7 + 3 = 10\text{ ns}$$

the worst case is S_3 evaluation, taking 10ns.

(b)



The longest path from top left to bottom right traverses 20 sum blocks. It traverses ⁽ⁱ⁾ 6 sum and 14 carry to

get P15 or (ii) T_{sum} and 13 carry to get P14.

Let T_s = delay to sum, T_c = delay to carry.

(A) if $T_s = 2T$, $T_c = T$, then (i) takes longer time

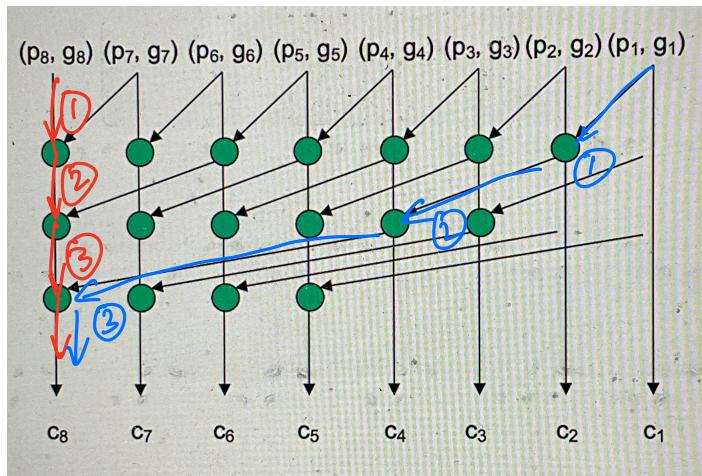
$$T_{delay} = 6T_s + 14T_c = 26T$$

(B) if $T_s = T$, $T_c = 2T$, then (ii) takes longer time

$$T_{delay} = 7T_s + 13T_c = 33T$$

To optimize the delay, I would focus on reducing the delay of carry path, (A) would be preferred over (B).

#2



$g_i = X_i Y_i$ each evaluation takes an AND operation
 $P_i = X_i \oplus Y_i$ each evaluation takes an XOR
Each processing component (node) evaluates

$$(g_{\text{out}}, P_{\text{out}}) = (g_{i_1} + P_{i_1}, g_{i_2} \cdot P_{i_1} \cdot P_{i_2})$$

Just looking at the diagram (neglecting necessary computations that obtains p_i , g_i , and c_i), the critical paths are those passing through 3 computational components. Two examples are labeled.

Each component takes 1 AND and 1 OR (2 units of time)
 \Rightarrow 3 components take 3 AND and 3 OR (6 units of time)

For the full computation, g_i evaluation takes 1 AND. P_i takes 1 XOR, so 1 unit in total for preparing (g_i, p_i) . Computing each c_i also takes an AND and an OR after computing all components (2 units in total).

So in total from X_i, Y_i to C_i , it takes $1+6+2 = 9$ units.

#3

The module code

```
// Q3_multiplier.v
`timescale 1ns/1ps
module multiplier_3x3 (
    input [2:0] A,
    input [2:0] B,
    output [5:0] P
);
// intermediate products
wire [5:0] P0, P1, P2;
// widen B
wire [5:0] B_6bit;

assign B_6bit = {3'b0, B};

// multiply A[0] by B
assign P0 = A[0]? B_6bit : 6'b0;
assign P1 = A[1]? B_6bit << 1 : 6'b0;
assign P2 = A[2]? B_6bit << 2 : 6'b0;
assign P = P0 + P1 + P2;
endmodule
```

The testbench

```
// Q3_tb_multiplier.v
`timescale 1ns/1ps
module tb_multiplier_3x3;
reg [2:0] A, B;
wire [5:0] P;
integer i;
multiplier_3x3 dut (.A(A), .B(B), .P(P));

initial begin
    $display("---- 3-bit Multiplier Test ----");

    // initialize
    A=0; B=0;
    #1;
    // test 1: set A = 5, loop over all possible B
    $display("test 1: A=5, B runs from 0 to 7");
    for (i = 0; i < 8; i = i + 1) begin
        A = 5;
        B = i;
        #1;
        $display("A=%0d (%b) B=%0d (%b) ->
P=%0d (%b)",
            A, A, B, B, P, P);
    end
    // test 2: set B = 2, loop over all possible A
    $display("test 2: A runs from 0 to 7, B=2");
    for (i = 0; i < 8; i = i + 1) begin
        A = i;
        B = 2;
        #1;
        $display("A=%0d (%b) B=%0d (%b) ->
P=%0d (%b)",
            A, A, B, B, P, P);
    end

    $display("All tests completed.");
    $finish;
end
```

Simulation result printout

Console - SimVision

File Edit View Simulation Windows Help

cadence

Text Search:

17,000ps + 0

```
xcelium>
xcelium> source /vol/cadence2018/XCELLIUM1809/tools/xcelium/files/xmsimrc
xcelium> run
---- 3-bit Multiplier Test ----
test 1: A=5, B runs from 0 to 7
A=5 {101} B=0 {000} -> P=0 {000000}
A=5 {101} B=1 {001} -> P=5 {000101}
A=5 {101} B=2 {010} -> P=10 {001010}
A=5 {101} B=3 {011} -> P=15 {001111}
A=5 {101} B=4 {100} -> P=20 {010100}
A=5 {101} B=5 {101} -> P=25 {011001}
A=5 {101} B=6 {110} -> P=30 {011110}
A=5 {101} B=7 {111} -> P=35 {100011}
test 2: A runs from 0 to 7, B=2
A=0 {000} B=2 {010} -> P=0 {000000}
A=1 {001} B=2 {010} -> P=2 {000010}
A=2 {010} B=2 {010} -> P=4 {000100}
A=3 {011} B=2 {010} -> P=6 {000110}
A=4 {100} B=2 {010} -> P=8 {001000}
A=5 {101} B=2 {010} -> P=10 {001010}
A=6 {110} B=2 {010} -> P=12 {001100}
A=7 {111} B=2 {010} -> P=14 {001110}
All tests completed.
Simulation complete via $finish(1) at time 17 NS + 0
./q3_tb_multiplier.v:34      $finish;
```

xcelium>

SimVision simulator

File Project Run Window Help

The module code

```
// Q4_barrel.v

`timescale 1ns/1ps
module Vrbarrel16 (DIN, S, C, DOUT);
    input [15:0] DIN;
    input [3:0] S;
    input [2:0]C;
    output [15:0] DOUT;
    reg [15:0] DOUT;
    parameter Lrotate = 3'b000, //the coding of different shift modes
        Rrotate = 3'b001,
        Llogical = 3'b010,
        Rlogical = 3'b011,
        Larith = 3'b100,
        Rarith = 3'b101;

    function [15:0] Vrol;
        input [15:0] D;
        input [3:0] S;
        integer ii, N;
        reg [15:0] TMPD;
        begin
            N = S;
            TMPD = D;
            for (ii = 1; ii<=N; ii = ii+1)
                TMPD = {TMPD[14:0], TMPD[15]};
            Vrol = TMPD;
        end
    endfunction

    function [15:0] Vror;
        input [15:0] D;
        input [3:0] S;
        integer ii, N;
        reg [15:0] TMPD;
        begin
            N = S;
            TMPD = D;
            for (ii = 1; ii<=N; ii = ii+1)
                TMPD = {TMPD[0], TMPD[15:1]};
            Vror = TMPD;
        end
    endfunction

    function [15:0] Vsll;
        input [15:0] D;
        input [3:0] S;
        integer ii, N;
        reg [15:0] TMPD;
        begin
            N = S;
            TMPD = D;
            for (ii = 1; ii<=N; ii = ii+1)
                TMPD = {TMPD[14:0], 1'b0};
            Vsll = TMPD;
        end
    endfunction

```

// continue

```
function [15:0] Vsll;
  input [15:0] D;
  input [3:0] S;
  integer ii, N;
  reg [15:0] TMPD;
begin
  N = S;
  TMPD = D;
  for (ii = 1; ii<=N; ii = ii+1)
    TMPD = {1'b0, TMPD[15:1]};
  Vsll = TMPD;
end
endfunction

function [15:0] Vslr;
  input [15:0] D;
  input [3:0] S;
  integer ii, N;
  reg [15:0] TMPD;
begin
  N = S;
  TMPD = D;
  for (ii = 1; ii<=N; ii = ii+1)
    TMPD = {TMPD[14:0], 1'b0};
  Vslr = TMPD;
end
endfunction

function [15:0] Vsra;
  input [15:0] D;
  input [3:0] S;
  integer ii, N;
  reg [15:0] TMPD;
begin
  N = S;
  TMPD = D;
  for (ii = 1; ii<=N; ii = ii+1)
    TMPD = {TMPD[15], TMPD[15:1]};
  Vsra = TMPD;
end
endfunction
```

```
always @ (DIN or S or C)
  case (C)
    Lrotate: DOUT = Vrol(DIN, S);
    Rrotate: DOUT = Vror(DIN, S);
    Llogical: DOUT = Vsll(DIN, S);
    Rlogical: DOUT = Vslr(DIN, S);
    Larith: DOUT = Vsla(DIN, S);
    Rarith: DOUT = Vsra(DIN, S);
    default: DOUT = DIN;
  endcase
endmodule
```

```

// testbench

`timescale 1ns/1ps
module tb_barrel;
  reg [15:0] DIN;
  reg [3:0] S;
  reg [2:0] C;
  wire [15:0] DOUT;
  integer i;
  Vrbarrel16 dut
  (.DIN(DIN), .S(S), .C(C), .DOUT(DOUT));

initial begin
  $display("---- 16-bit Barrel Shifter Test ----");

  // initialize
  DIN=0; S=0; C=0;
  #1;
  // DIN = [1001011101010011]
  // S = [0011]
  // run all possible C values
  DIN = 16'b1001011101010011;
  S = 4'b0011;
  $display("DIN = %b", DIN);
  $display("S = %0d (%b)", S, S);
  #1
  for (i = 0; i < 6; i = i + 1) begin
    C = i;
    #1;
    $display("C=%b -> DOUT=%b",
      C, DOUT);
  end

  $display("All tests completed.");
  $finish;
end
endmodule

```

Screenshot result

Console - SimVision

File Edit View Simulation Windows Help

cadence

xcelium>
xcelium> source /vol/cadence2018/XCELUM1809/tools/xcelium/files/xmsimrc
xcelium> run
---- 16-bit Barrel Shifter Test ----
DIN = 1001011101010011
s = 3 (0011)
c=000 -> DOUT=1011101010011100
c=001 -> DOUT=01110001011101010
c=010 -> DOUT=101110101010011000
c=011 -> DOUT=00010001011101010
c=100 -> DOUT=101110101010011000
c=101 -> DOUT=11110010111101010
All tests completed.
Simulation complete via \$finish(1) at time 8 NS + 0
. ./q4_tb_barrel.v:32 \$finish;
xcelium>

SimVision simulator