به نام خدا

دانشگاه تهران

پردیس دانشکده‌های فنی

دانشکده مهندسی برق و کامپیوتر

سیستم‌های دیجیتال 1

استاد: دکتر نوابی

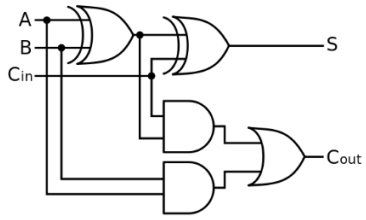
تمرین کامپیوتری شماره 3

فربد سیاه‌کلی

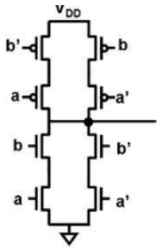
شماره دانشجویی: 810198510

فروردین 1400

1) شکل مدار یک Full Adder:



دیلی‌ها با توجه به پروژه اول، nmos #(3,4,5) و pmos #(5,6,7) هستند. که در نتیجه با توجه به cmos structure گیت xor:



گیت دیلی to1 xor: to0 inverter + 2*to1 pmos = 7 + 2*5 = 17ns

گیت دیلی to0 xor: to1 inverter + 2*toz pmos = 5 + 2*7 = 19ns

در نتیجه دیلی سیم S (19) + (17 | 19) می‌باشد که یعنی #(36,38) است.

همچنین به جای استفاده از and و or، از سه عدد nand استفاده شد که دیلی #(10,8) دارند

در نتیجه دیلی Co که از مسیر یک xor و دو nand می‌گذرد، برابر با #(35,37) می‌شود.

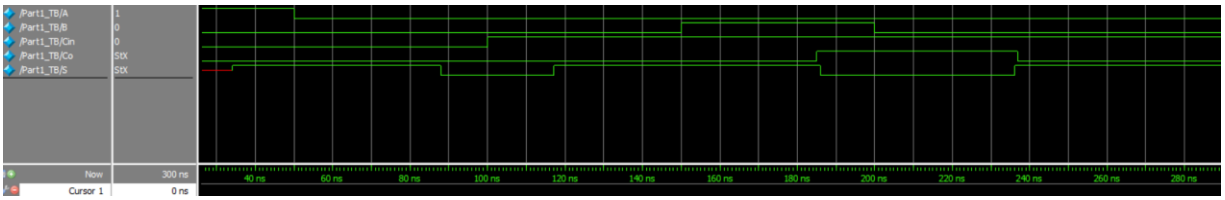
کد سیستم وریلاگ مدار Full Adder:

```
`timescale 1ns/1ns
module FullAdder(input A, B, Cin, output Co, S);
 wire [2:0] w;
 assign #(17,19) w[0] = A ^ B;
 assign #(17,19) S = w[0] ^ Cin;
 assign #(10,8) w[1] = ~(Cin & w[0]);
 assign #(10,8) w[2] = ~(A & B);
 assign #(10,8) Co = ~(w[1] & w[2]);
endmodule
```

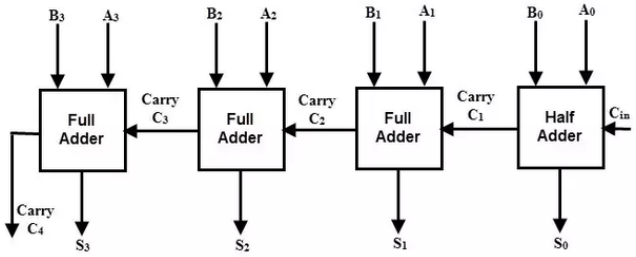
تست بنچ مربوطه:

```
`timescale 1ns/1ns
module Part1_TB();
 reg A = 1;
 reg B = 0;
 reg Cin = 0;
 FullAdder UUT(A, B, Cin, Co, S);
 initial begin
  #50; A = 0;
  #50; Cin = 1;
  #50; B = 1;
  #50; B = 0;
  #50; A = 0;
  #50; Cin = 1;
  #50 $stop;
 end
endmodule
```

شکل موج خروجی به صورت زیر است که در آن worst delayهای هر دو سیم خروجی دیده می‌شوند.



2) با cascade کردن FAها به شکل زیر، می‌توان یک n bit adder ساخت:



با استفاده از یک حلقه، n بار بیت‌های A و B را با ورودی Cin خودشان جمع می‌کنیم و سپس Co هر فول ادر را به فول ادر بعدی می‌دهیم. همچنین از بخش قبل دریافتیم که دیلی S (36,38)# و دیلی Co (37,35)# می‌باشد.

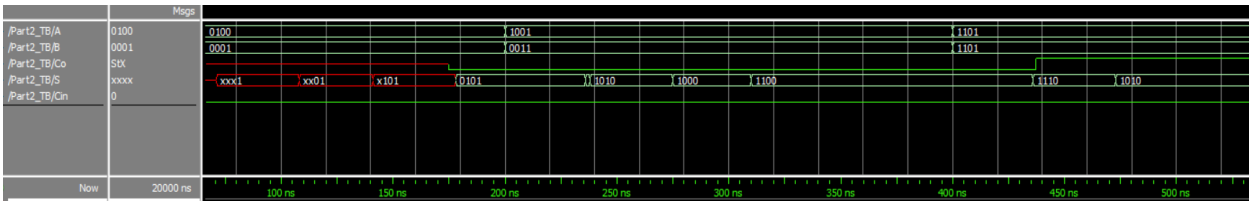
کد سیستم وریلاگ این n bit adder به شرح زیر می‌شود:

```
module n_bit_adder #(parameter n = 4) (output [n - 1 : 0] S, output Co, input [n - 1 : 0] A, B, input Cin);
 wire [n : 0] #(37,35) Ci;
 wire [n - 1 : 0] #(36,38) s;
 assign Ci[0] = Cin;
 genvar i;
 for(i = 0  ; i < n ; i = i + 1) begin
    assign {Ci[i+1], s[i]} = A[i] + B[i] + Ci[i];
 end
     assign  S = s;
 assign Co = Ci[n];
endmodule
```

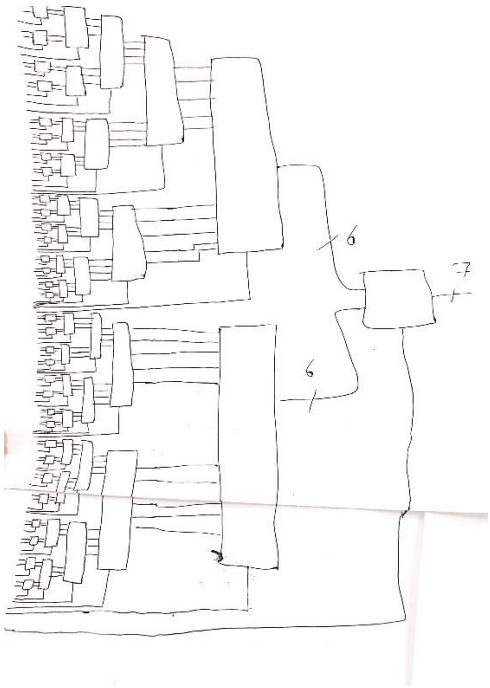
3) تست بنچ مربوطه که در آن n را برابر 4 می‌گذاریم و با استفاده از $random و repeat هر بار مقادیری برای A و b می‌سازیم:

```
`timescale 1ns/1ns
module Part2_TB();
 logic [3:0] A = 4'b1001;
 logic [3:0] B = 4'b1100;
 wire Co;
 wire [3:0] S;
 reg Cin = 0;
 integer seed;
 n_bit_adder #4 UUT(S, Co, A, B, Cin);
 initial begin
repeat (20) begin
  #0 A = $random(seed);
  #0 B = $random(seed);
  #200;
  end
 $stop;
 end
endmodule
```

شکل موج خروجی که چند مثال را نشان می‌دهد:



4) مدار one’s counter:



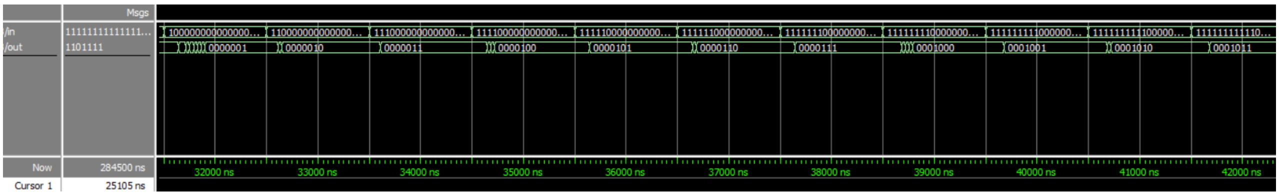
کد سیستم وریلاگ توصیف کننده مدار که در آن با حلقه‌های متفاوت لایه‌های مدار را طراحی می‌کنیم:

```
module onescounter127bit #(parameter n = 6) (input [2**(n+1) : 0] I, output [n : 0] out);
 wire [65 : 0] w1;
 wire [47 : 0] w2;
 wire [31 : 0] w3;
 wire [19 : 0] w4;
 wire [11 : 0] w5;
 wire [6 : 0] w6;
 generate
 genvar i;
  for(i = 0; i < 32; i = i + 1) begin
    n_bit_adder #1 nbitadder(I[3*i], I[3*i + 1], I[3*i + 2],  w1[2*i], w1[2*i + 1]);
  end
  for(i = 0; i < 16; i = i + 1) begin
    n_bit_adder #2 nbitadder(w1[4*i+1 : 4*i], w1[4*i+3 : 4*i+2], I[96+i],  w2[3*i+1 : 3*i], w2[3*i+2]);
  end
  for(i = 0; i < 8; i = i + 1) begin
    n_bit_adder #3 nbitadder(w2[6*i+2 : 6*i], w2[6*i+5 : 6*i+3], I[112+i],  w3[4*i+2 : 4*i], w3[4*i+3]);
  end
  for(i = 0; i < 4; i = i + 1) begin
    n_bit_adder #4 nbitadder(w3[8*i+3 : 8*i], w3[8*i+7 : 8*i+4], I[120+i],  w4[5*i+3 : 5*i], w4[5*i+4]);
  end
  for(i = 0; i < 2; i = i + 1) begin
    n_bit_adder #5 nbitadder(w4[10*i+4 : 10*i], w4[10*i+9 : 10*i+5], I[124+i],  w5[6*i+4 : 6*i], w5[6*i+5]);
  end
  n_bit_adder #6 nbitadder(w5[5:0], w5[11:6], I[126],  w6[5 : 0], w6[6]);
 endgenerate
 assign out = w6;
endmodule
```

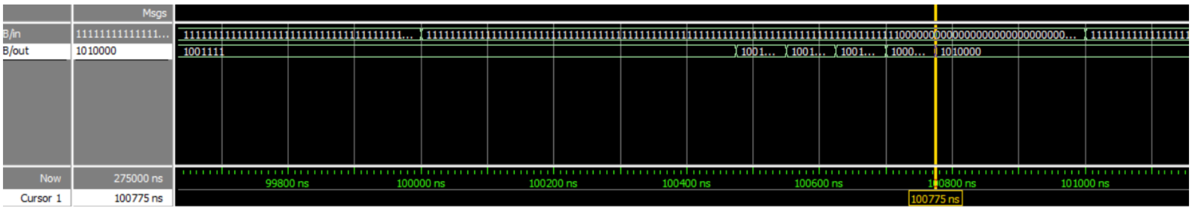
5) تست بنچ مربوطه که در آن ابتدا 20 بار مقادیر رندوم گرفتیم و پس از آن marching را شروع کردیم:

```
`timescale 1ns/1ns
module Part4_TB();
 reg [126 : 0] in;
 wire [6 : 0] out;
 integer seed, i;
 onescounter127bit  #6 onecounter(out, in);
 initial begin
  repeat(20) begin
    #1000 in = $random(seed);
  end
  #1000 in = 127'b1000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000;
  repeat(126) begin
    #1000 in = {~in[0], in[126 : 1]};
  end
  #1000;
  $stop;
 end
endmodule
```

بخشی از شکل موج one’s counter در حالت marching 1:



بیشترین دیلی دیده شده در marching:



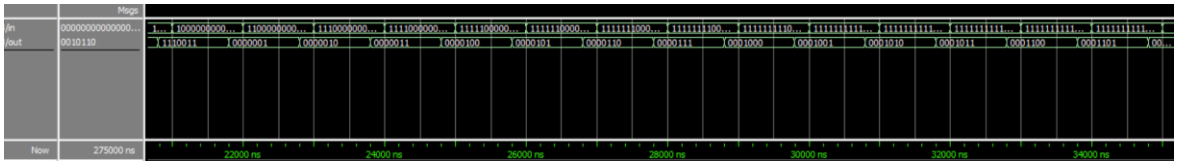
6) کد سیستم وریلاگ توصیف کننده one’s counter با استفاده از always که دیلی آن نیز با توجه به 6 level داشتن سیستم (که هرکدام با توجه به شماره لایه، همان تعداد FA دارد) برابر (1+2+3+4+5+6+7)*38 یا همان 798 نانوثانیه است:

```
module onescount127bit_always #(parameter n = 6) (input [126 : 0] in, output [n : 0] out);
        integer i, temp = 0;
        always@(in) begin
                temp = 0;
                for(i = 0; i < 127; i = i + 1) begin
                        temp = temp + in[i];
                end
        end
        assign #798 out = temp;
endmodule
```

تست بنچ مربوطه که در آن n را برابر 6 می‌گذاریم:

```
`timescale 1ns/1ns
module Part4_TB();
 reg [126 : 0] in;
 wire [6 : 0] out;
 integer seed, i;
 onescounter127bit  #6 onecounter(out, in);
 initial begin
  repeat(20) begin
    #1000 in = $random(seed);
  end
  #1000 in = 127'b10000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000;
  repeat(126) begin
    #1000 in = {~in[0], in[126 : 1]};
  end
  #1000;
  $stop;
 end
endmodule
```

خروجی تست بنچ:



7) تعداد گیت‌های به کار رفته در سنتز بخش 4:

```
ABC RESULTS:               NAND cells:       30
ABC RESULTS:                NOR cells:       41
ABC RESULTS:                NOT cells:       13
ABC RESULTS:          internal signals:       23
ABC RESULTS:             input signals:       13
ABC RESULTS:            output signals:        7
```

تعداد گیت‌های به کار رفته در سنتز بخش 6:

```
ABC RESULTS:               NAND cells:      643
ABC RESULTS:                NOR cells:      830
ABC RESULTS:                NOT cells:      272
ABC RESULTS:          internal signals:      704
ABC RESULTS:             input signals:      134
ABC RESULTS:            output signals:        7
```

همانطور که مشاهده می‌شود نتیجه بخش 4 به مراتب بسیار بهینه‌تر از سنتز بخش 6 شده است. که با توجه به اینکه در بخش 4 ما عملا levelهای مدار را طراحی کرده بودیم و اما در بخش 6 طراحی مدار را به برنامه سپرده بودیم، این تفاوت محسوس در نتایج، قابل پیش بینی بود.