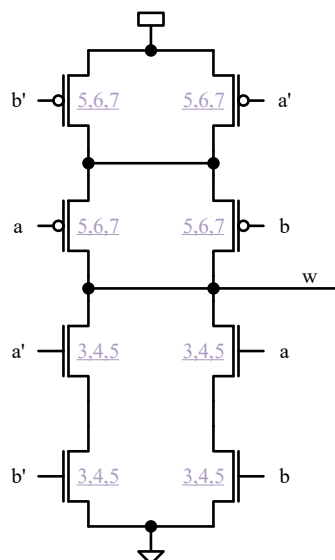


شماره سوال	محمد مهدی معینی منش	سیستم‌های دیجیتال	تاریخ تحویل:
۸۱۰۱۹۸۴۷۵	CA3	۱۴۰۰/۲/۲۶	۱/۱۴

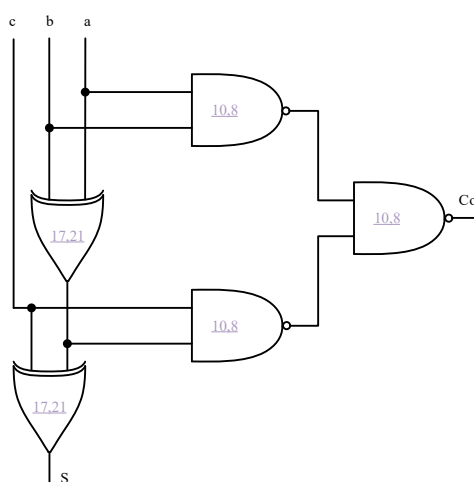
در CA1 تأخیر برای nmos (3,4,5) و برای pmos (5,6,7) در نظر گرفته شده بود. این مقادیر سبب می‌شوند گیت nand حداکثر تأخیر (10,8) را داشته باشد (این مورد در CA1 به دست آمد). اکنون می‌خواهیم با ترانزیستورهای مذکور حداکثر تأخیرهای گیت xor را محاسبه نماییم (زیرا در ساخت Full adder به آن نیاز داریم).

یک گیت xor ساختاری مشابه زیر دارد.



با توجه به مقادیر نوشته شده برای ترانزیستورها و همچنین در نظر گرفتن not‌ها - که در شکل بالا نشان داده نشده است - بدترین تأخیر  $t_{00}$  برابر 21ns (تبدیل حالت  $a=1$  و  $b=0$  به  $a=0$  و  $b=0$ ) و بدترین تأخیر  $t_{01}$  برابر 17ns (تبدیل حالت  $a=1$  و  $b=1$  به  $a=1$  و  $b=0$ ) می‌باشد.

با توجه به اینکه اکنون مقادیر worst case delay را برای همه گیت‌های مورد نیاز می‌دانیم، ساختار Full adder را رسم می‌نماییم.



برای تعریف کردن این ماژول در سیستم وریلاگ، کد زیر نوشته شد.

```

1  `timescale 1ns/1ns
2
3  module full_adder(input a, b, c, output Co, S);
4      nand #(10,8) nand1(i, a, b), nand2(j, k, c), nand3(Co, i, j);
5      xor #(17,21) xor1(k, a, b), xor2(S, k, c);
6  endmodule

```

۱

این مازول باید از جدول درستی زیر پیروی کند.

a	b	c	S	Co
1	1	1	1	1
1	1	0	0	1
1	0	1	0	1
1	0	0	1	0
0	1	1	0	1
0	1	0	1	0
0	0	1	1	0
0	0	0	0	0

در این قسمت می‌خواهیم بدترین تأخیرها را محاسبه کنیم. برای Co، بدترین تأخیر صفر شدن موقعی رخ می‌دهد که xor تغییر کند و به واسطه آن یک nand یک بفرستد و nand آخر صفر بفرستد. پس بدترین تأخیر to0 برای Co برابر 39ns می‌باشد.

برای to1 شدن S بدترین تأخیر زمانی است که یک xor صفر بفرستد و xor آخر یک بفرستد که منجر می‌شود بدترین تأخیر برای آن برابر 38ns شود. تبدیل حالت a=0, b=1, c=1 به a=0, b=0, c=1 این دو worst case را اثبات می‌کند.

همچنین بدترین تأخیر یک شدن Co موقعی رخ می‌دهد که xor تغییر کند و به واسطه آن یک nand صفر بفرستد و nand آخر یک بفرستد. پس بدترین تأخیر to1 برای Co برابر 35ns می‌باشد. تبدیل حالت a=0, b=0, c=1 به a=1, b=0, c=1 این worst case را اثبات می‌کند.

برای to0 شدن S هم باید هر دو xor صفر بفرستند که بدترین تأخیر to0 آن بشود 42ns. تبدیل حالت a=0, b=1, c=0 به a=1, b=1, c=0 این worst case را اثبات می‌کند.

بنابراین داریم:

$$S: \begin{cases} to1: 38ns \\ to0: 42ns \end{cases}$$

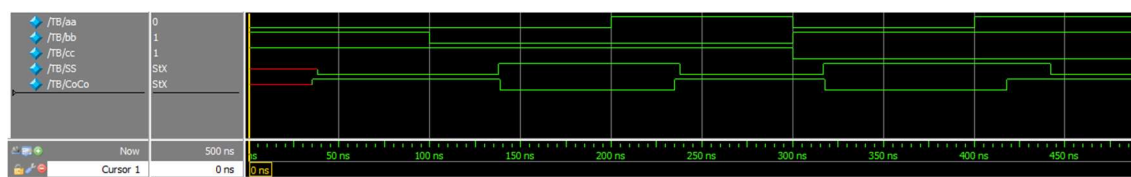
$$Co: \begin{cases} to1: 35ns \\ to0: 39ns \end{cases}$$

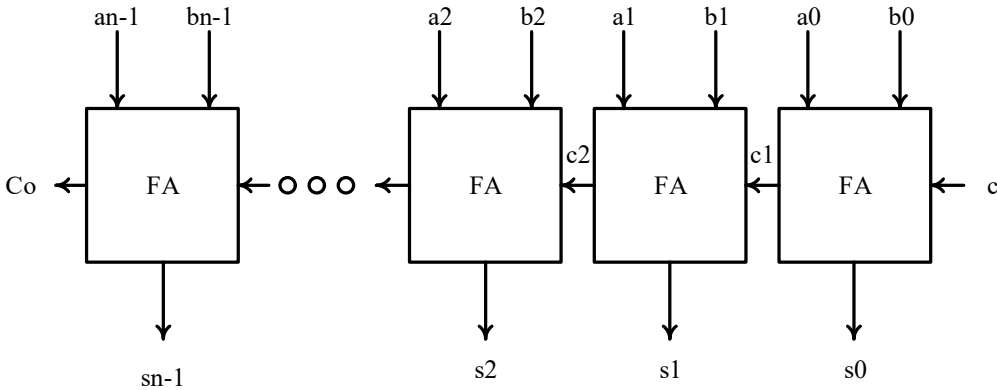
در اینجا، حالات گفته شده تست و بررسی می‌شود تا نشان داده شود مازول به درستی کار می‌کند و همچنین نتیجه دلخواه ما در پیدا کردن worst case delay را برآورده می‌سازد.

```

8 module TB ();
9     reg aa = 0, bb = 1, cc = 1; //First Initialize
10    wire SS, CoCo;
11    full_adder my_ic(aa, bb, cc, CoCo, SS);
12    initial begin
13        #100 bb=0; //Co to 0 -- S to 1
14        #100 aa=1; //Co to 1
15        #100 aa=0; bb=1; cc=0; //Reinitialize
16        #100 aa=1; //SS to 0
17        #100 $stop;
18    end
19 endmodule

```



<div>شماره سوال</div> <div>۸۱۰۱۹۸۴۷۵</div>	<div>محمد مهدی معینی منش</div> <div>سیستم‌های دیجیتال</div> <div>CA3</div>	<div>تاریخ تحویل:</div> <div>۱۴۰۰/۲/۲۶</div>	<div>۳/۱۴</div>
	<div>۲</div> <div>با توجه به ساختار زیر برای جمع‌کننده خواسته شده در صورت پروژه، این قسمت انجام می‌گردد.</div> <div>  </div> <div>کد زیر برای این جمع‌کننده n-بیتی نوشته شد.</div> <div> <pre> 1  `timescale 1ns/1ns 2 3  module nbit_full_adder(a, b, c, Co, S); 4      parameter n = 6; 5      input [n-1:0] a; 6      input [n-1:0] b; 7      input c; 8      output Co; 9      output [n-1:0] S; 10     wire [n-1:0] inner_S; 11     wire inner_Co; 12 13     assign {inner_Co, inner_S} = a + b + c; 14     assign #(38*n, 42*n) S = inner_S; 15     assign #(35*n, 39*n) Co = inner_Co; 16 endmodule </pre> </div>		

شماره سوال	محمد مهدی معینی منش	سیستم‌های دیجیتال	تاریخ تحویل:
۸۱۰۱۹۸۴۷۵	CA3	۱۴۰۰/۲/۲۶	۴/۱۴

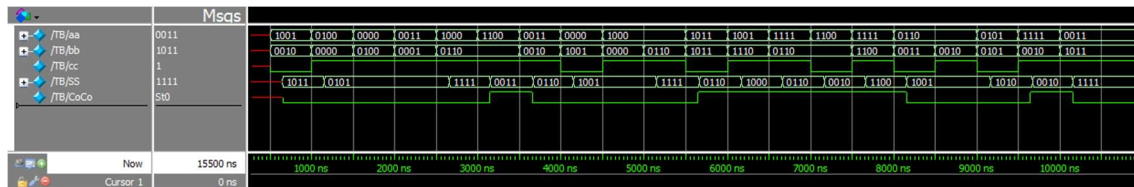
در این قسمت تست‌بنچی برای این سوال به کمک repeat و random نوشته می‌شود.

```

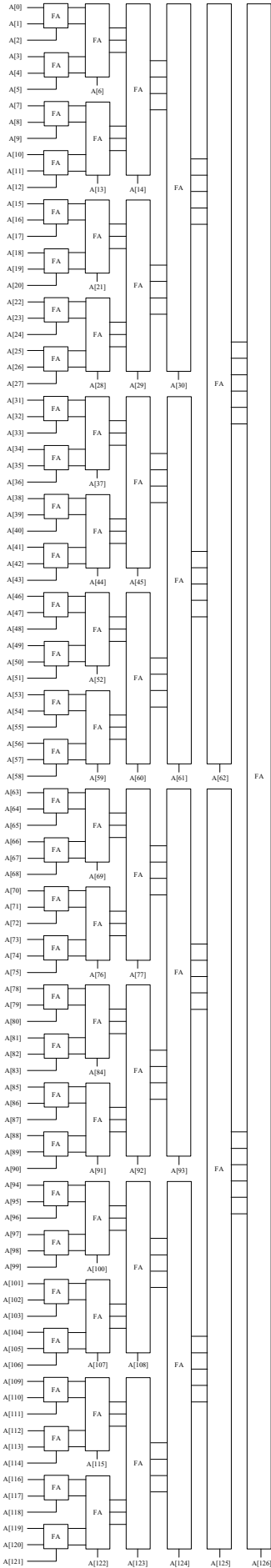
1  `timescale 1ns/1ns
2
3  module TB ();
4      reg [3:0] aa;
5      reg [3:0] bb;
6      reg cc;
7
8      wire [3:0] SS;
9      wire CoCo;
10     nbit_full_adder #4 my_ic(aa, bb, cc, CoCo, SS);
11     initial begin
12         repeat (20) #500 {aa,bb,cc}=$random;
13         #200 $stop;
14     end
15 endmodule

```

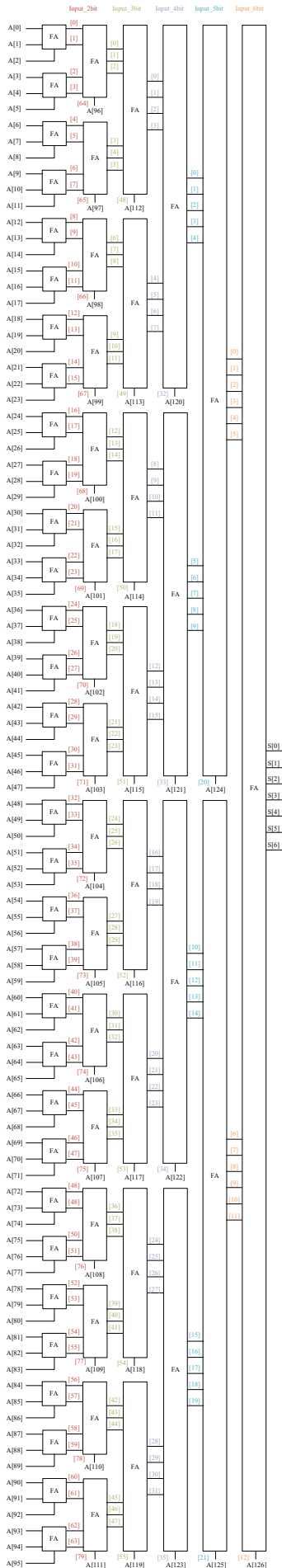
خروجی نرم‌افزار ModelSim به صورت زیر است.



<div>۵/۱۴</div>	<div>تاریخ تحویل:</div> <div>۱۴۰۰/۲/۲۶</div>	<div>سیستم‌های دیجیتال</div> <div>CA3</div>	<div>محمد مهدی معینی منش</div> <div>۸۱۰۱۹۸۴۷۵</div>	<div>شماره</div> <div>سوال</div>
			<div>ابتدا ساختاری که این شمارنده دارد، رسم می‌شود.</div>	<div>۴</div>



شماره سوال	محمد مهدی معینی منش	سیستم‌های دیجیتال	تاریخ تحویل:	۶/۱۴
۴	پیااده‌سازی ساختار اخیر، از لحاظ کدنویسی (خصوصاً با ساختار generate) دشوار است. بنابراین، ساختار را به شکل زیر تغییر می‌دهیم.	CA3	۸۱۰۱۹۸۴۷۵	۱۴۰۰/۲/۲۶



شماره سوال	محمد مهدی معینی منش	سیستم‌های دیجیتال	تاریخ تحویل:
۸۱۰۱۹۸۴۷۵	CA3	۱۴۰۰/۲/۲۶	۷/۱۴
۴	برای تعریف کردن این ماژول در SystemVerilog کد زیر نوشته شد.		
<pre>1 module ones_counter(input [126:0] i, output [6:0] S); 2 3     wire [95:0] input_1bit; 4     wire [79:0] input_2bit; 5     wire [55:0] input_3bit; 6     wire [35:0] input_4bit; 7     wire [21:0] input_5bit; 8     wire [12:0] input_6bit; 9     genvar n; 10 11     assign input_1bit [95:0] = i[95:0]; 12     assign input_2bit [79:64] = i[111:96]; 13     assign input_3bit [55:48] = i[119:112]; 14     assign input_4bit [35:32] = i[123:120]; 15     assign input_5bit [21:20] = i[125:124]; 16     assign input_6bit [12] = i[126]; 17 18     generate 19         for (n=0;n&lt;32;n=n+1) begin: one_bits 20             nbit_full_adder #1 one_bit(input_1bit[3*n+1],input_1bit[3*n],input_1bit[3*n+2],input_2bit[2*n+1],input_2bit[2*n]); 21         end 22 23         for (n=0;n&lt;16;n=n+1) begin: two_bits 24             nbit_full_adder #2 two_bit(input_2bit[4*n+1:4*n],input_2bit[4*n+3:4*n+2],input_2bit[n+64],input_3bit[3*n+2],input_3bit[3*n+1:3*n]); 25         end 26 27         for (n=0;n&lt;8;n=n+1) begin: three_bits 28             nbit_full_adder #3 three_bit(input_3bit[6*n+2:6*n],input_3bit[6*n+5:6*n+3],input_3bit[n+48],input_4bit[4*n+3],input_4bit[4*n+2:4*n]); 29         end 30 31         for (n=0;n&lt;4;n=n+1) begin: four_bits 32             nbit_full_adder #4 four_bit(input_4bit[8*n+3:8*n],input_4bit[8*n+7:8*n+4],input_4bit[n+32],input_5bit[5*n+4],input_5bit[5*n+3:5*n]); 33         end 34 35         for (n=0;n&lt;2;n=n+1) begin: five_bits 36             nbit_full_adder #5 five_bit(input_5bit[10*n+4:10*n],input_5bit[10*n+9:10*n+5],input_5bit[n+20],input_6bit[6*n+5],input_6bit[6*n+4:6*n]); 37         end 38 39         nbit_full_adder #6 six_bit(input_6bit[5:0],input_6bit[11:6],input_6bit[12],S[6],S[5:0]); 40 41     endgenerate 42 endmodule</pre>			

شماره سوال	محمد مهدی معینی منش	سیستم‌های دیجیتال CA3	تاریخ تحویل: ۱۴۰۰/۲/۲۶	۸/۱۴
---------------	---------------------	--------------------------	---------------------------	------

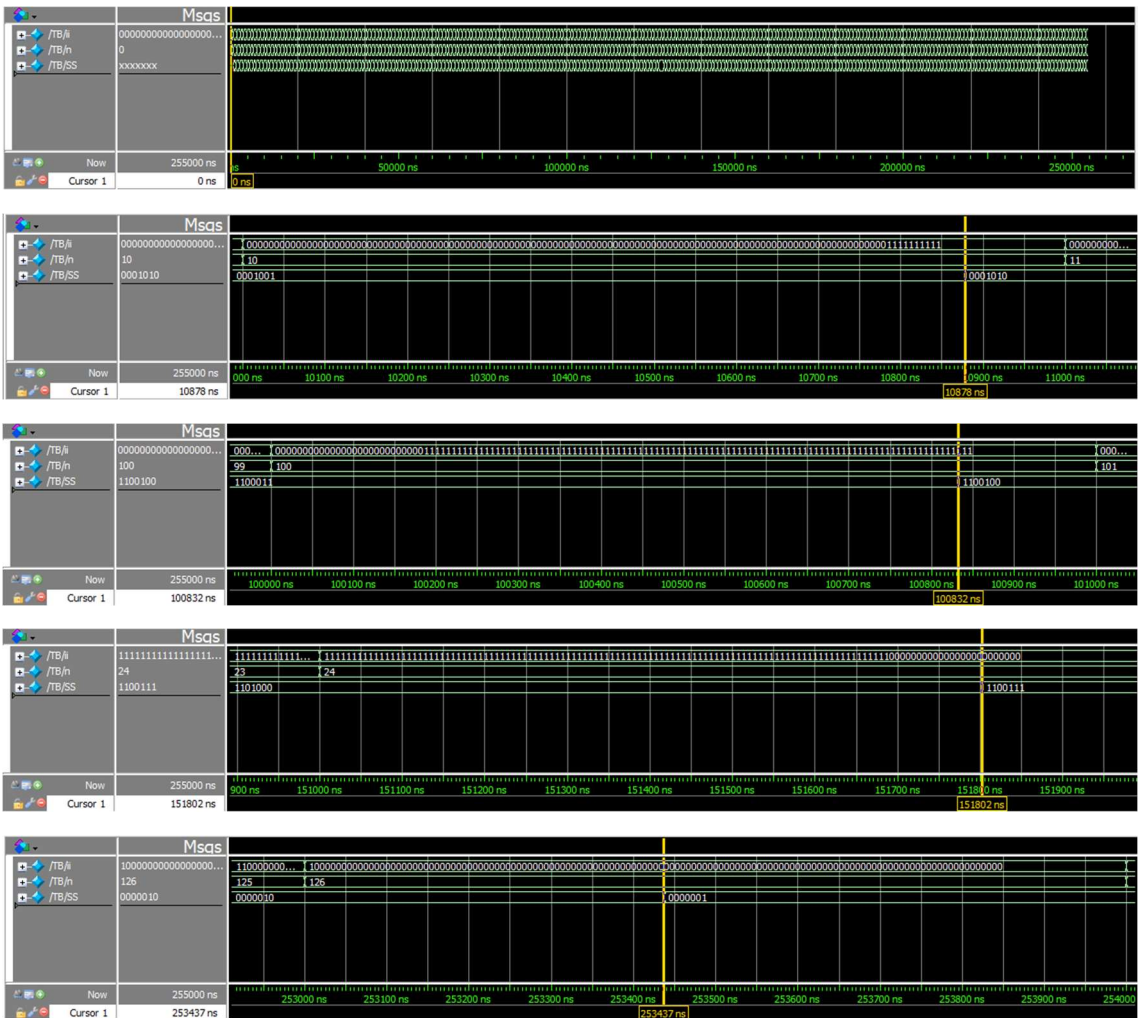
تست‌بنچ این سوال با توجه به خواسته سوال (Marching-1) نوشته شده است. کد آن به شکل زیر است.

```

1  module TB ();
2      reg [126:0] ii=1'd0;
3      int n;
4      wire [6:0] SS;
5      ones_counter my_ic(ii, SS);
6      initial begin
7          for (n=0;n<127;n=n+1)begin
8              #1000 ii=(ii+1)*2-1;
9          end
10         for (n=0;n<127;n=n+1)begin
11             #1000 ii=ii*2;
12         end
13         #1000 $stop;
14     end
15 endmodule

```

این تست‌بنچ در مجموع ۲۵۴ حالت را تست و بررسی می‌کند و عملاً همه‌ی این حالات قابل نمایش نیست. بنابراین ابتدا یک تصویر کلی و سپس چندین تصویر جزئی از آن در ادامه آورده شده است.





<div>شماره</div> <div>سوال</div>	<div>محمد مهدی معینی منش</div> <div>CA3</div> <div>سیستم‌های دیجیتال</div> <div>تاریخ تحویل:</div>	<div>۸۱۰۱۹۸۴۷۵</div> <div>۱۴۰۰/۲/۲۶</div> <div>۹/۱۴</div>	<div>۶</div> <div>کد نوشته شده به صورت زیر است. تأخیرهای نوشته شده بر حسب بدترین تأخیرهای به دست آمده در قسمت‌های قبل است.</div> <div> <pre> 1 module ones_counter(input [126:0] i, output reg [6:0] s); 2     always @(i) begin 3         int n; 4         reg [95:0] input_1bit; 5         reg [79:0] input_2bit; 6         reg [55:0] input_3bit; 7         reg [35:0] input_4bit; 8         reg [21:0] input_5bit; 9         reg [12:0] input_6bit; 10 11         for (n=0;n&lt;96;n=n+1) input_1bit[n]=i[n]; 12         for (n=64;n&lt;80;n=n+1) input_2bit[n]=i[n+32]; 13         for (n=48;n&lt;56;n=n+1) input_3bit[n]=i[n+64]; 14         for (n=32;n&lt;36;n=n+1) input_4bit[n]=i[n+88]; 15         for (n=20;n&lt;22;n=n+1) input_5bit[n]=i[n+104]; 16         input_6bit [12] = i[126]; 17 18         for (n=0;n&lt;32;n=n+1) begin 19             {input_2bit[2*n+1],input_2bit[2*n]} = 20                 input_1bit[3*n]+ 21                 input_1bit[3*n+1]+ 22                 input_1bit[3*n+2]; 23         end 24         #42 25 26         for (n=0;n&lt;16;n=n+1) begin 27             {input_3bit[3*n+2],input_3bit[3*n+1],input_3bit[3*n]} = 28                 input_2bit[4*n]+ 29                 input_2bit[4*n+1]*2+ 30                 input_2bit[4*n+2]+ 31                 input_2bit[4*n+3]*2+ 32                 input_2bit[n+64]; 33         end 34         #84 35 36         for (n=0;n&lt;8;n=n+1) begin 37             {input_4bit[4*n+3],input_4bit[4*n+2],input_4bit[4*n+1],input_4bit[4*n]} = 38                 input_3bit[6*n]+ 39                 input_3bit[6*n+1]*2+ 40                 input_3bit[6*n+2]*4+ 41                 input_3bit[6*n+3]+ 42                 input_3bit[6*n+4]*2+ 43                 input_3bit[6*n+5]*4+ 44                 input_3bit[n+48]; 45         end 46         #126 47 48         for (n=0;n&lt;4;n=n+1) begin 49             {input_5bit[5*n+4],input_5bit[5*n+3],input_5bit[5*n+2],input_5bit[5*n+1],input_5bit[5*n]} = 50                 input_4bit[8*n]+ 51                 input_4bit[8*n+1]*2+ 52                 input_4bit[8*n+2]*4+ 53                 input_4bit[8*n+3]*8+ 54                 input_4bit[8*n+4]+ 55                 input_4bit[8*n+5]*2+ 56                 input_4bit[8*n+6]*4+ 57                 input_4bit[8*n+7]*8+ 58                 input_4bit[n+32]; 59         end 60         #168 61 62         for (n=0;n&lt;2;n=n+1) begin 63             {input_6bit[6*n+5],input_6bit[6*n+4],input_6bit[6*n+3],input_6bit[6*n+2],input_6bit[6*n+1],input_6bit[6*n]} = 64                 input_5bit[10*n]+ 65                 input_5bit[10*n+1]*2+ 66                 input_5bit[10*n+2]*4+ 67                 input_5bit[10*n+3]*8+ 68                 input_5bit[10*n+4]*16+ 69                 input_5bit[10*n+5]+ 70                 input_5bit[10*n+6]*2+ 71                 input_5bit[10*n+7]*4+ 72                 input_5bit[10*n+8]*8+ 73                 input_5bit[10*n+9]*16+ 74                 input_5bit[n+20]; 75         end 76         #210 77 78         #252{s[6],s[5],s[4],s[3],s[2],s[1],s[0]} = 79             input_6bit[0]+ 80             input_6bit[1]*2+ 81             input_6bit[2]*4+ 82             input_6bit[3]*8+ 83             input_6bit[4]*16+ 84             input_6bit[5]*32+ 85             input_6bit[6]+ 86             input_6bit[7]*2+ 87             input_6bit[8]*4+ 88             input_6bit[9]*8+ 89             input_6bit[10]*16+ 90             input_6bit[11]*32+ 91             input_6bit[12]; 92     end 93 endmodule 94 </pre> </div>
----------------------------------	--	---	---

شماره سوال	محمد مهدی معینی منش	سیستم‌های دیجیتال	تاریخ تحویل:	۱۰/۱۴	۱۴۰۰/۲/۲۶
	۸۱۰۱۹۸۴۷۵	CA3			
۶	<p>اما کد اخیر عملاً فرقی با کد نوشته شده در قسمت ۴ ندارد. بنابراین کد دیگری نوشته می‌شود و در ادامه هم این کد جدید مورد استفاده قرار خواهد گرفت. در این کد یک تأخیر که بدترین تأخیر کل می‌باشد برای همه حالات در نظر گرفته می‌شود.</p> <pre> 1  `timescale 1ns/1ns 2 3  module ones_counter(input [126:0] i, output reg [6:0] S); 4      int n; 5      always @(i) begin 6          #630 S=7'b0; 7          for (n=0;n&lt;126;n=n+1)begin 8              if(i[n]) S=S+1; 9          end 10         end 11     endmodule </pre>				

شماره سوال	محمد مهدی معینی منش ۸۱۰۱۹۸۴۷۵	سیستم‌های دیجیتال CA3	تاریخ تحویل: ۱۴۰۰/۲/۲۶	۱۱/۱۴
------------	----------------------------------	--------------------------	---------------------------	-------

حالا می‌خواهیم به کمک ابزار yosys، عملیات سنتزکردن را برای بخش‌های ۴ و ۶ انجام دهیم. لازم به ذکر است برای تبدیل فایل‌های با پسوند sv به فایل‌های به پسوند v (برای شناسایی نرم‌افزار yosys) تغییراتی صورت گرفته است.

برای بخش ۴ نتایج به شکل زیر است.

برای full adder یک بیتی:

```

D:\University\Term4\DigitalSystems\yosys\src\yosys.exe
ABC: + strash
ABC: + dc2
ABC: + scorr
ABC: Warning: The network is combinational (run "fraig" or "fraig_sweep").
ABC: + ifraig
ABC: + retime -o
ABC: + strash
ABC: + dch -f
ABC: + map
ABC: + write_blif <abc-temp-dir>/output.blif

4.2.3. Re-integrating ABC results
ABC RESULTS:      NAND cells:      5
ABC RESULTS:      NOR cells:       6
ABC RESULTS:      NOT cells:       3
ABC RESULTS:      internal signals: 3
ABC RESULTS:      input signals:   3
ABC RESULTS:      output signals:  2
Removing temp directory.

4.2. Extracting gate netlist of module '\full_adder_2' to '<abc-temp-dir>/input.blif'..
Extracted 8 gates and 13 wires to a netlist network with 5 inputs and 3 outputs.

4.2.1. Executing ABC.
Running ABC command: <yosys-exe-dir>/yosys-abc -s -f <abc-temp-dir>/abc.script 2>&1
ABC: ABC command line: "source <abc-temp-dir>/abc.script".
ABC:
ABC: + read_blif <abc-temp-dir>/input.blif
ABC: + read_lib -w D:\University\Term4\DigitalSystems\yosys\src\mycells.lib
ABC: Parsing finished successfully. Parsing time = 0.00 sec

```

برای full adder دو بیتی:

```

D:\University\Term4\DigitalSystems\yosys\src\yosys.exe
ABC: Memory = 0.00 MB. Time = 0.00 sec
ABC: + strash
ABC: + dc2
ABC: + scorr
ABC: Warning: The network is combinational (run "fraig" or "fraig_sweep").
ABC: + ifraig
ABC: + retime -o
ABC: + strash
ABC: + dch -f
ABC: + map
ABC: + write_blif <abc-temp-dir>/output.blif

4.2.3. Re-integrating ABC results
ABC RESULTS:      NAND cells:      8
ABC RESULTS:      NOR cells:      12
ABC RESULTS:      NOT cells:       5
ABC RESULTS:      internal signals: 5
ABC RESULTS:      input signals:   5
ABC RESULTS:      output signals:  3
Removing temp directory.

4.3. Extracting gate netlist of module '\full_adder_3' to '<abc-temp-dir>/input.blif'..
Extracted 13 gates and 20 wires to a netlist network with 7 inputs and 4 outputs.

4.3.1. Executing ABC.
Running ABC command: <yosys-exe-dir>/yosys-abc -s -f <abc-temp-dir>/abc.script 2>&1
ABC: ABC command line: "source <abc-temp-dir>/abc.script".
ABC:
ABC: + read_blif <abc-temp-dir>/input.blif
ABC: + read_lib -w D:\University\Term4\DigitalSystems\yosys\src\mycells.lib

```

۷

برای full adder سه بیتی:

```
D:\University\Term4\DigitalSystems\yosys\src\yosys.exe
-state; 0 no func). Time = 0.00 sec
ABC: Memory = 0.00 MB. Time = 0.00 sec
ABC: + strash
ABC: + dc2
ABC: + scorr
ABC: Warning: The network is combinational (run "fraig" or "fraig_sweep").
ABC: + ifraig
ABC: + retime -o
ABC: + strash
ABC: + dch -f
ABC: + map
ABC: + write_blif <abc-temp-dir>/output.blif
ABC RESULTS: NAND cells: 10
ABC RESULTS: NOR cells: 19
ABC RESULTS: NOT cells: 9
ABC RESULTS: internal signals: 9
ABC RESULTS: input signals: 7
ABC RESULTS: output signals: 4
4.4. Extracting gate netlist of module '\full_adder_4' to '<abc-temp-dir>/input.blif'..
Extracted 20 gates and 29 wires to a netlist network with 9 inputs and 5 outputs.
4.4.1. Executing ABC.
Running ABC command: <yosys-exe-dir>/yosys-abc -s -f <abc-temp-dir>/abc.script 2>&1
ABC: ABC command line: "source <abc-temp-dir>/abc.script".
ABC:
ABC: + read_blif <abc-temp-dir>/input.blif
```

برای full adder چهار بیتی:

```
D:\University\Term4\DigitalSystems\yosys\src\yosys.exe
ABC: Library "demo" from "D:\University\Term4\DigitalSystems\yosys\src\mycells.lib" has 4 cells (1 skipped: 1 seq; 0 tri
-state; 0 no func). Time = 0.00 sec
ABC: Memory = 0.00 MB. Time = 0.00 sec
ABC: + strash
ABC: + dc2
ABC: + scorr
ABC: Warning: The network is combinational (run "fraig" or "fraig_sweep").
ABC: + ifraig
ABC: + retime -o
ABC: + strash
ABC: + dch -f
ABC: + map
ABC: + write_blif <abc-temp-dir>/output.blif
ABC RESULTS: NAND cells: 13
ABC RESULTS: NOR cells: 25
ABC RESULTS: NOT cells: 11
ABC RESULTS: internal signals: 15
ABC RESULTS: input signals: 9
ABC RESULTS: output signals: 5
4.5. Extracting gate netlist of module '\full_adder_5' to '<abc-temp-dir>/input.blif'..
Extracted 24 gates and 35 wires to a netlist network with 11 inputs and 6 outputs.
4.5.1. Executing ABC.
Running ABC command: <yosys-exe-dir>/yosys-abc -s -f <abc-temp-dir>/abc.script 2>&1
ABC: ABC command line: "source <abc-temp-dir>/abc.script".
ABC:
```

۷

برای full adder پنج بیتی:

```

D:\University\Term4\DigitalSystems\yosys\src\yosys.exe
ABC: + scorr
ABC: Warning: The network is combinational (run "fraig" or "fraig_sweep").
ABC: + ifraig
ABC: + retime -o
ABC: + strash
ABC: + dch -f
ABC: + map
ABC: + write_blif <abc-temp-dir>/output.blif
4.6.3. Integrating ABC results:
ABC RESULTS:      NAND cells:      18
ABC RESULTS:      NOR cells:       30
ABC RESULTS:      NOT cells:       11
ABC RESULTS:      internal signals: 18
ABC RESULTS:      input signals:   11
ABC RESULTS:      output signals:   6
Removing temp directory.

4.6. Extracting gate netlist of module '\full_adder_6' to '<abc-temp-dir>/input.blif'..
Extracted 32 gates and 45 wires to a netlist network with 13 inputs and 7 outputs.

4.6.1. Executing ABC.
Running ABC command: <yosys-exe-dir>/yosys-abc -s -f <abc-temp-dir>/abc.script 2>&1
ABC: ABC command line: "source <abc-temp-dir>/abc.script".
ABC:
ABC: + read_blif <abc-temp-dir>/input.blif
ABC: + read_lib -w D:\University\Term4\DigitalSystems\yosys\src\mycells.lib
ABC: Parsing finished successfully. Parsing time = 0.00 sec
ABC: Warning: Templates are not defined.
ABC: Libery parser cannot read "time unit". Assuming time unit : "1ns".

```

برای full adder شش بیتی:

```

D:\University\Term4\DigitalSystems\yosys\src\yosys.exe
ABC: + dc2
ABC: + scorr
ABC: Warning: The network is combinational (run "fraig" or "fraig_sweep").
ABC: + ifraig
ABC: + retime -o
ABC: + strash
ABC: + dch -f
ABC: + map
ABC: + write_blif <abc-temp-dir>/output.blif
4.7.1. Integrating ABC results:
ABC RESULTS:      NAND cells:      18
ABC RESULTS:      NOR cells:       39
ABC RESULTS:      NOT cells:       19
ABC RESULTS:      internal signals: 25
ABC RESULTS:      input signals:   13
ABC RESULTS:      output signals:   7
Removing temp directory.

4.7. Extracting gate netlist of module '\ones_counter' to '<abc-temp-dir>/input.blif'..
Extracted 0 gates and 0 wires to a netlist network with 0 inputs and 0 outputs.
Don't call ABC as there is nothing to map.
Removing temp directory.

yosys> write_verilog -noattr Q4_s.v

5. Executing Verilog backend.
Dumping module '\full_adder_1'.
Dumping module '\full_adder_2'.
Dumping module '\full_adder_3'.

```

با توجه به اینکه در این بخش تشکیل شده است از ۳۲ تا full adder یک بیتی، ۱۶ تا full adder دو بیتی، ۸ تا full adder سه بیتی، ۴ تا full adder چهار بیتی، ۲ تا full adder پنج بیتی و یک عدد full adder شش بیتی، می‌توان گفت کل این مجموعه تشکیل شده است از گیت‌های nand، nor و not که تعدادشان در این قسمت آورده شده است.

$$\text{NAND cells: } 32 \times 5 + 16 \times 8 + 8 \times 10 + 4 \times 13 + 2 \times 18 + 1 \times 18 = 474$$

$$\text{NOR cells: } 32 \times 6 + 16 \times 12 + 8 \times 19 + 4 \times 25 + 2 \times 30 + 1 \times 39 = 735$$

$$\text{NOT cells: } 32 \times 3 + 16 \times 5 + 8 \times 9 + 4 \times 11 + 2 \times 11 + 1 \times 19 = 333$$

۷

برای بخش ۶ نتایج به شکل زیر است.

```

D:\University\Term4\DigitalSystems\yosys\src\yosys.exe
ABC: + read_lib -w D:\University\Term4\DigitalSystems\yosys\src\mycells.lib
ABC: Parsing finished successfully. Parsing time = 0.00 sec
ABC: Warning: Templates are not defined.
ABC: Libery parser cannot read "time_unit". Assuming time_unit : "1ns".
ABC: Libery parser cannot read "capacitive_load_unit". Assuming capacitive_load_unit(1, pf).
ABC: Scl_LibertyReadGenlib() skipped sequential cell "DFF".
ABC: Library "demo" from "D:\University\Term4\DigitalSystems\yosys\src\mycells.lib" has 4 cells (1 skipped: 1 seq; 0 tri
-state; 0 no func). Time = 0.00 sec
ABC: Memory = 0.00 MB. Time = 0.00 sec
ABC: + strash
ABC: + dc2
ABC: + scorr
ABC: Warning: The network is combinational (run "fraig" or "fraig_sweep").
ABC: + ifraig
ABC: + retime -o
ABC: + strash
ABC: + dch -f
ABC: + map
ABC: + write_blif <abc-temp-dir>/output.blif
ABC: + no integrating abc results.
ABC RESULTS:      NAND cells:      1413
ABC RESULTS:      NOR cells:      1743
ABC RESULTS:      NOT cells:       512
ABC RESULTS:      internal signals: 3161
ABC RESULTS:      input signals:   126
ABC RESULTS:      output signals:   7
Removing temp directory.
yosys>
  
```

در مقایسه این دو ماژول طبق جدول زیر کاملاً واضح است که ماژولی که در بخش چهار طراحی شد، بسیار بهتر سنتز شد چرا که تا حد خوبی خودمان راه را نشان ابزار yosys دادیم و بهینه‌سازی بیشتری حاصل شد. اما برای بخش شش هیچ طراحی خاصی انجام ندادیم. به هر حال می‌دانیم که **There is no free lunch**!

	Part 4	Part 6
NAND	474	1413
NOR	735	1743
NOT	333	512