بسمه تعالی

دانشکده مهندسی برق و کامپیوتر

دانشگاه صنعتی اصفهان

زبان‌های توصیف سخت‌افزار و مدارات – نیمسال دوم ۹۹-۱۳۹۸

تکلیف شماره یک – تحویل یکشنبه ۱۳۹۹/۱/۱۰

مریم سعید مهر

ش.د. : ۹۶۲۹۳۷۳

فهرست

- تصاویر
- کدها
- فایل‌های ضمیمه شده

تصاویر

سوال اول:

در این سوال می‌توانید از هر یک از سطوح Gate، Data flow و یا Behavioral استفاده کنید. در نهایت لازم است برای هر یک از ماژول‌ها یک test bench به گونه‌ای نوشته شود که حالات ممکن برای آن ماژول را دربر بگیرد. هر یک از این تست بنچ‌ها باید دربردارنده‌ی حد اقل سه جفت عدد ورودی متفاوت برای ماژول باشند.

الف)

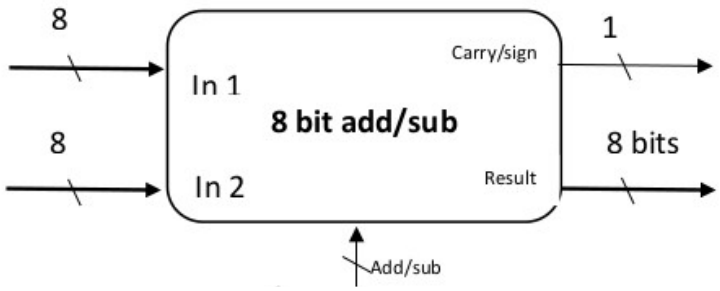
یک full adder تک بیتی طراحی نمایید.

ب)

با استفاده از full adder 1 بیتی‌ای که در قسمت قبل طراحی کردید، یک full adder 8 بیتی بسازید. در این قسمت فقط مجاز هستید که از instant ماژول طراحی شده در قسمت قبل استفاده کنید.

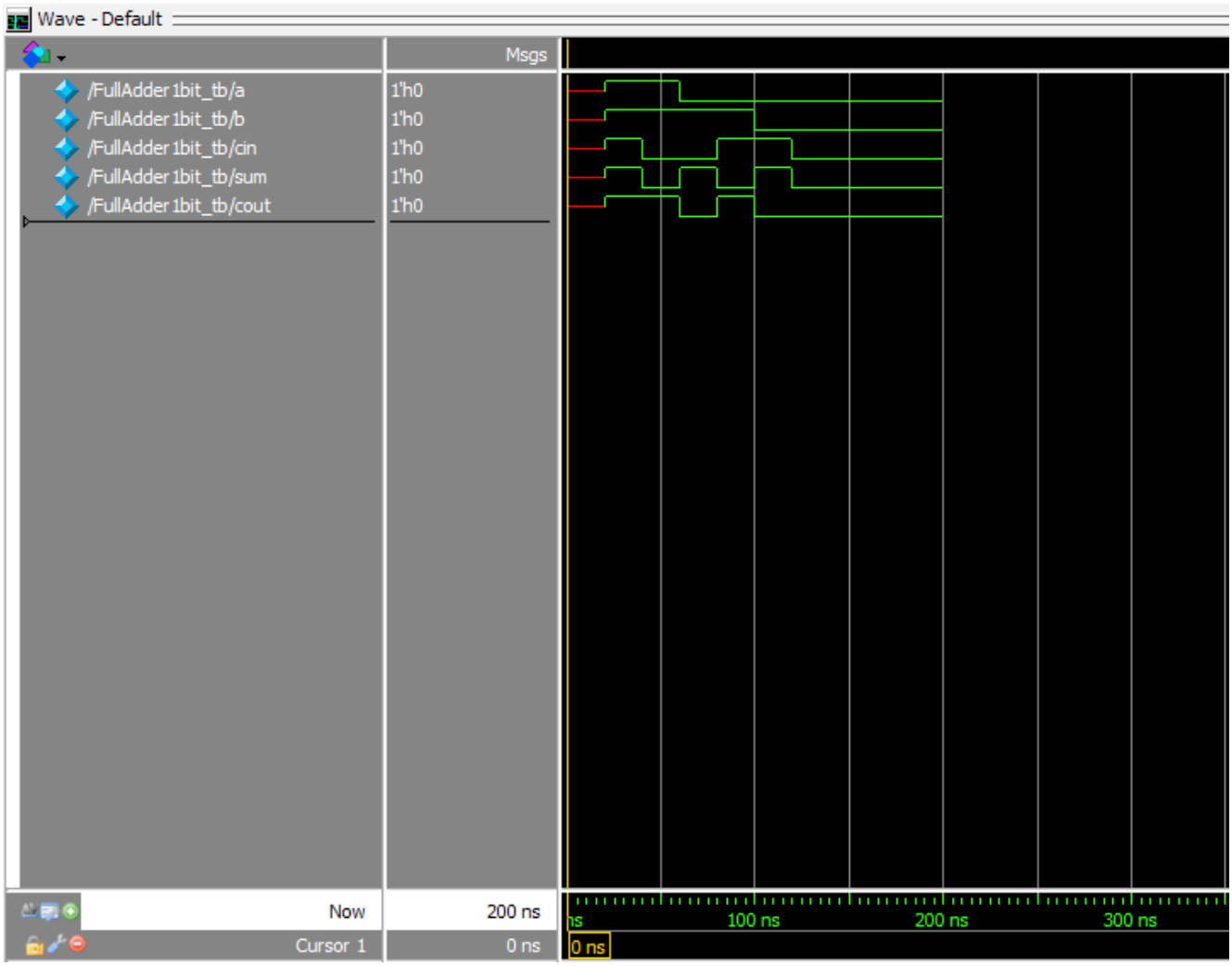
ج)

کد قسمت (ب) را به گونه ای تغییر دهید که تبدیل به یک جمع و تفریق کننده‌ی 8 بیتی شود. یک جمع و تفریق کننده‌ی هشت بیتی ماژولی است که دارای دو ورودی 8 بیتی برای اعداد ورودی و یک ورودی یک بیتی برای تعیین جمع یا تفریق کردن اعداد ورودی است. ماژول با یک شدن پایه کنترلی عمل جمع، و با صفر شدن عمل تفریق را انجام می دهد.



** اسکرین شات های مربوط به اجرای هر یک از testbench ها را در ادامه آورده‌ام...

الف) Full Adder 1 bit

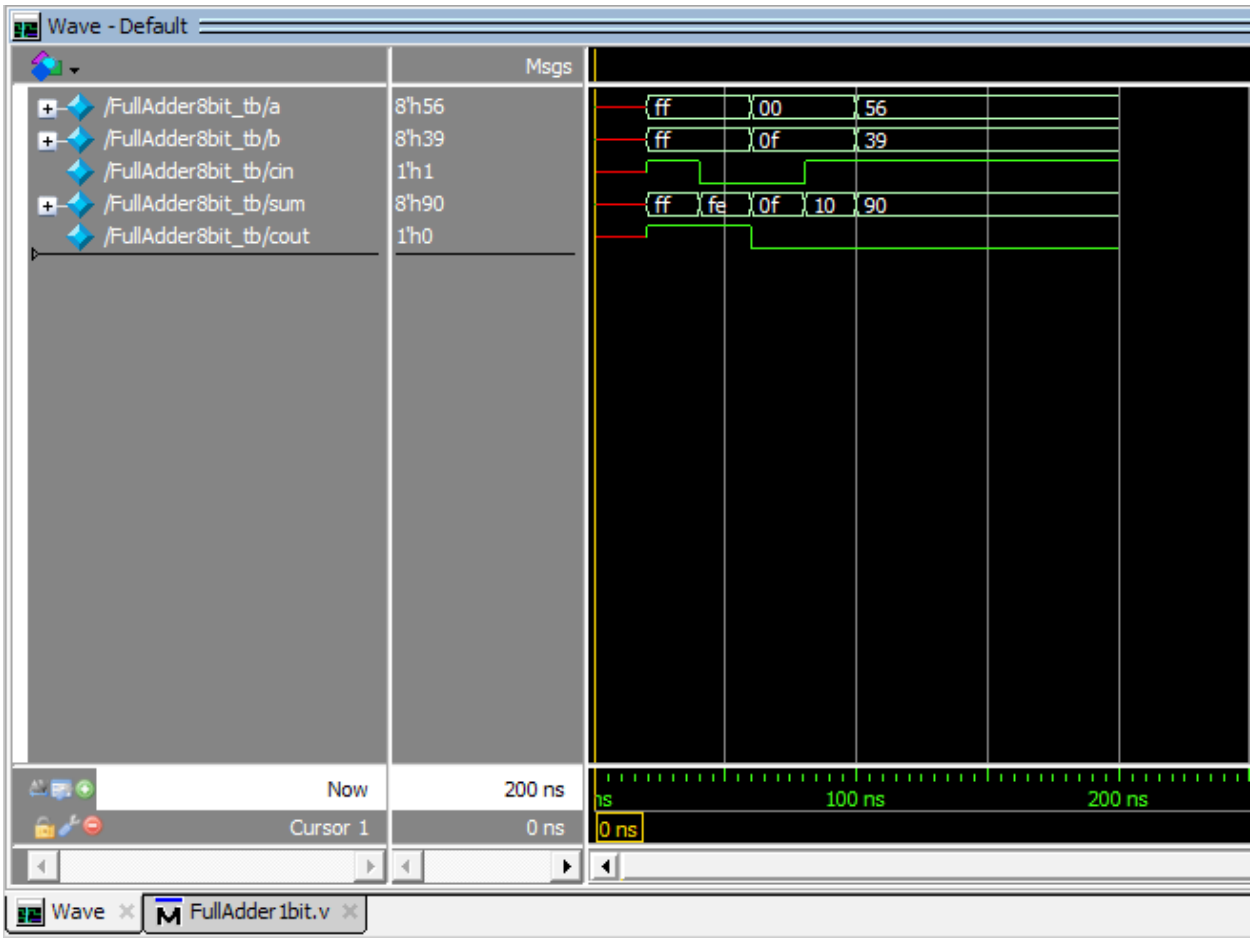


Transcript

```
ModelSim> vsim -voptargs=+acc work.FullAdder1bit_tb
# vsim -voptargs="+acc" work.FullAdder1bit_tb
# Start time: 01:59:02 on Mar 25,2020
# ** Note: (vsim-8009) Loading existing optimized design _opt1
# Loading work.FullAdder1bit_tb(fast)
# Loading work.FullAdder1bit(fast)
add wave -position end  sim:/FullAdder1bit_tb/a
add wave -position end  sim:/FullAdder1bit_tb/b
add wave -position end  sim:/FullAdder1bit_tb/cin
add wave -position end  sim:/FullAdder1bit_tb/sum
add wave -position end  sim:/FullAdder1bit_tb/cout
VSIM 8> run
#                    0 a=x , b=x , cin=x , sum=x , cout=x
#                   20 a=1 , b=1 , cin=1 , sum=1 , cout=1
#                   40 a=1 , b=1 , cin=0 , sum=0 , cout=1
#                   60 a=0 , b=1 , cin=0 , sum=1 , cout=0
#                   80 a=0 , b=1 , cin=1 , sum=0 , cout=1
#                  100 a=0 , b=0 , cin=1 , sum=1 , cout=0
#                  120 a=0 , b=0 , cin=0 , sum=0 , cout=0

VSIM 9>
```

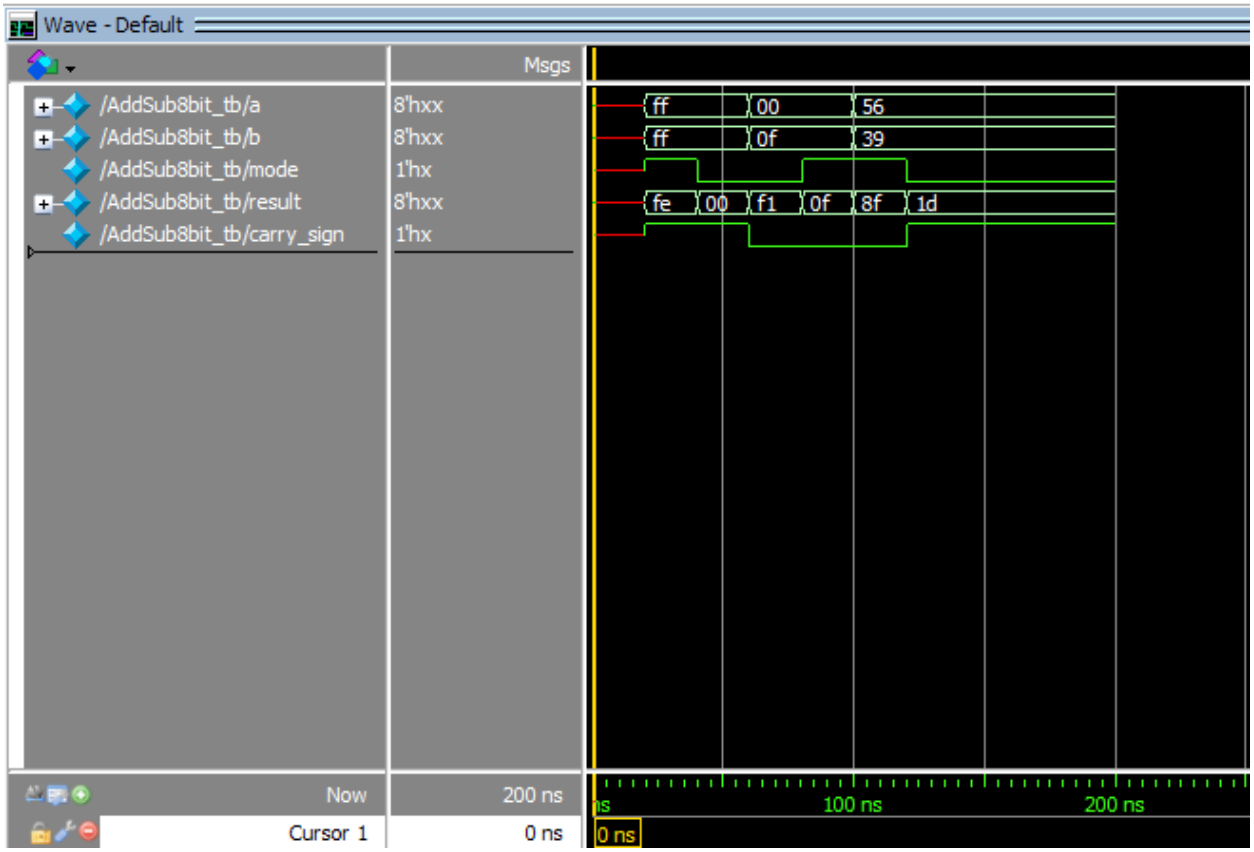
ب) Full Adder 8 bit




```
Transcript
ModelSim> vsim -voptargs=+acc work.FullAdder8bit_tb
# vsim -voptargs="+acc" work.FullAdder8bit_tb
# Start time: 02:10:47 on Mar 25,2020
# ** Note: (vsim-8009) Loading existing optimized design _opt
# Loading work.FullAdder8bit_tb(fast)
# Loading work.FullAdder8bit(fast)
# Loading work.FullAdder1bit(fast)
add wave -position end  sim:/FullAdder8bit_tb/a
add wave -position end  sim:/FullAdder8bit_tb/b
add wave -position end  sim:/FullAdder8bit_tb/cin
add wave -position end  sim:/FullAdder8bit_tb/sum
add wave -position end  sim:/FullAdder8bit_tb/cout
VSIM 16> run
#                    0 a=xxxxxxxx , b=xxxxxxxx , cin=x , sum=xxxxxxxx , cout=x
#                   20 a=11111111 , b=11111111 , cin=1 , sum=11111111 , cout=1
#                   40 a=11111111 , b=11111111 , cin=0 , sum=11111110 , cout=1
#                   60 a=00000000 , b=00001111 , cin=0 , sum=00001111 , cout=0
#                   80 a=00000000 , b=00001111 , cin=1 , sum=00010000 , cout=0
#                  100 a=01010110 , b=00111001 , cin=1 , sum=10010000 , cout=0

VSIM 17>
```

Add / Sub 8 bit (ج



Transcript

```
ModelSim> vsim -voptargs=+acc work.AddSub8bit_tb
# vsim -voptargs="+acc" work.AddSub8bit_tb
# Start time: 02:14:17 on Mar 25,2020
# ** Note: (vsim-8009) Loading existing optimized design _opt2
# Loading work.AddSub8bit_tb(fast)
# Loading work.AddSub8bit(fast)
# Loading work.FullAdderlbit(fast)
add wave -position end  sim:/AddSub8bit_tb/a
add wave -position end  sim:/AddSub8bit_tb/b
add wave -position end  sim:/AddSub8bit_tb/mode
add wave -position end  sim:/AddSub8bit_tb/result
add wave -position end  sim:/AddSub8bit_tb/carry_sign
VSIM 24> run
#                    0 Input1=xxxxxxxx , Input2=xxxxxxxx , Mode=x , Result=xxxxxxxx , Carry/Sign=x
#                   20 Input1=11111111 , Input2=11111111 , Mode=1 , Result=11111110 , Carry/Sign=1
#                   40 Input1=11111111 , Input2=11111111 , Mode=0 , Result=00000000 , Carry/Sign=1
#                   60 Input1=00000000 , Input2=00001111 , Mode=0 , Result=11110001 , Carry/Sign=0
#                   80 Input1=00000000 , Input2=00001111 , Mode=1 , Result=00001111 , Carry/Sign=0
#                  100 Input1=01010110 , Input2=00111001 , Mode=1 , Result=10001111 , Carry/Sign=0
#                  120 Input1=01010110 , Input2=00111001 , Mode=0 , Result=00011101 , Carry/Sign=1

VSIM 25>
```

سوال دوم:

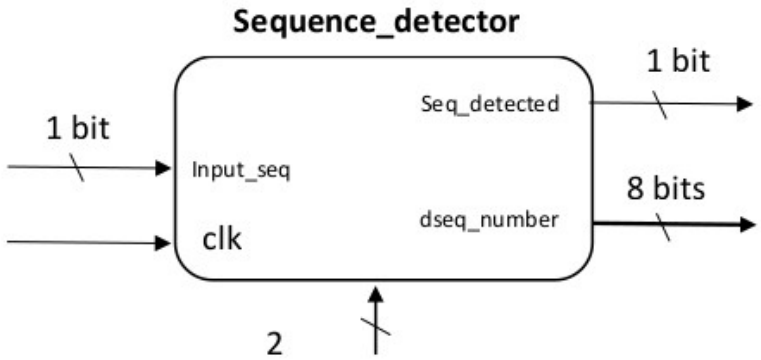
مطابق شکل زیر یک ماژول sequence detector درسطح Behavioral تعریف کنید. عملکرد ورودی ها و خروجی های این ماژول به شرح زیر است:

Input_seq: این ورودی تک بیتی در واقع رشته ای از 0 و1 ها است که به صورت سریال وارد ماژول میشود.

Seq_select: این ورودی 2 بیتی یکی از رشته های 1001، 0110، 1010 و یا 1100 را برای جستجو انتخاب می‌کند به این معنی که مثلا اگر seq_select=2 باشد، ماژول به دنبال رشته 0110 در رشته اعداد ورودی میگردد.

Seq_detected: این پایه یک بیتی پس از هر بار شناسایی رشته مورد نظر یک پالس ساعت یک میشود و سپس مجددا صفر میشود.

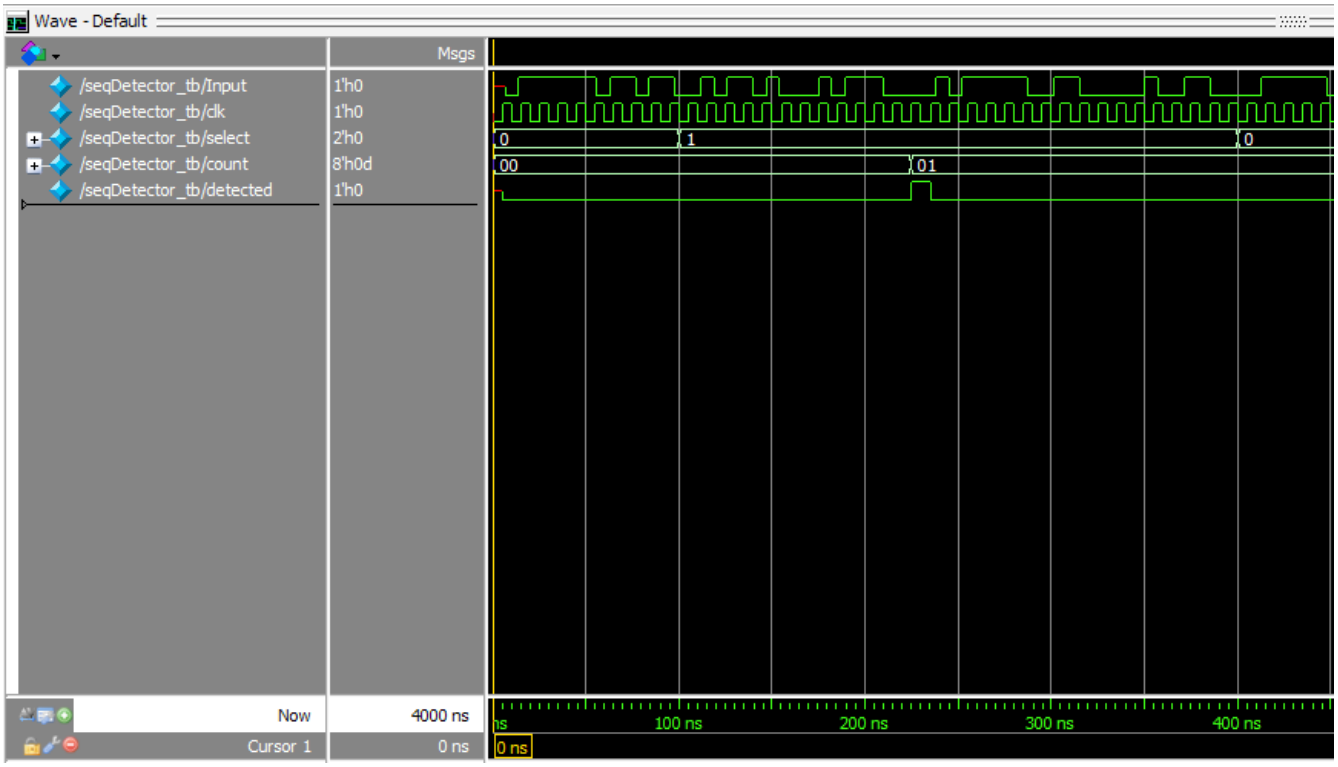
Dseq_number: این خروجی 8 بیتی تعداد رشته های تشخیص داده شده را نشان میدهد.



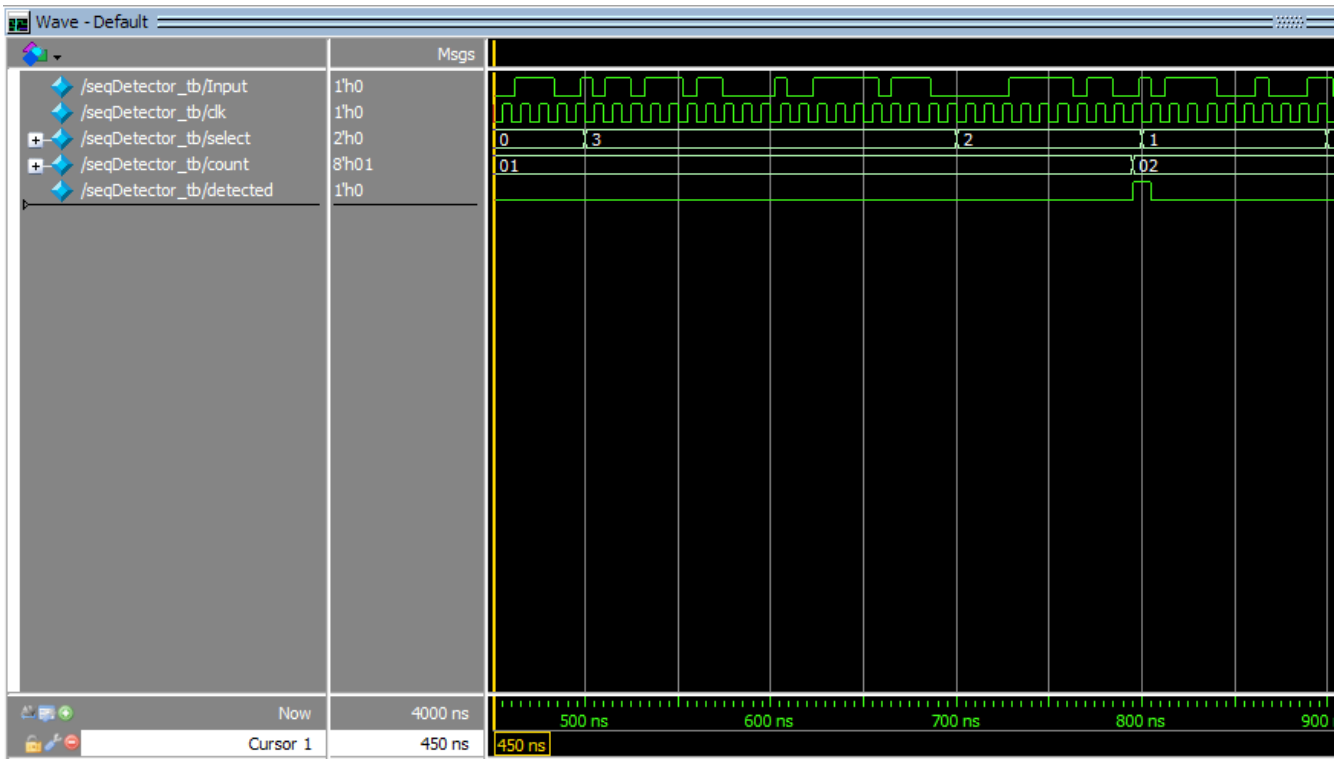
** اسکرین شات های مربوط به اجرای هر یک از testbench ها را در ادامه آورده‌ام...

*توجه : در این سؤال فرض کرده‌ام که dseq_number قرار است تمام رشته‌های یافت شده را بشمارد (ازهر نمونه‌ای که بود) به این معنا که با هر بار تعویض seq_select (یا به عبارتی با عوض شدن sample) ریست نمی شود!

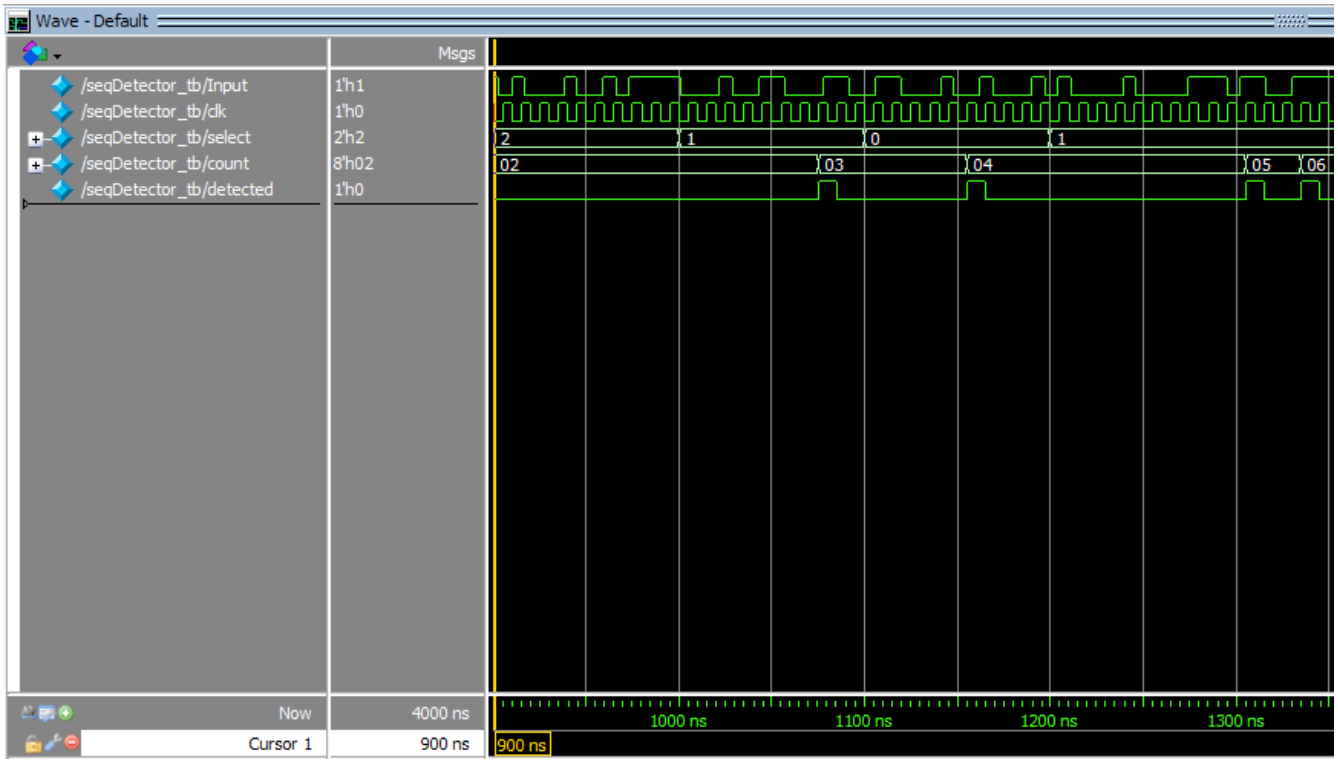
**توجه : در testbench ، ورودی ها را از نوع رندوم آورده‌ام تا حتی الامکان تمام حالات بررسی شود ، لذا برای مشاهده ی تمام کارکرد های ماژول ، به اجرای طولانی‌تر simulation نیاز بود و همچنین برای وضوح بیشتر عکس‌ها ، تصاویر با بازه ی ۴۵۰ ns یکبار گرفته شده اند.



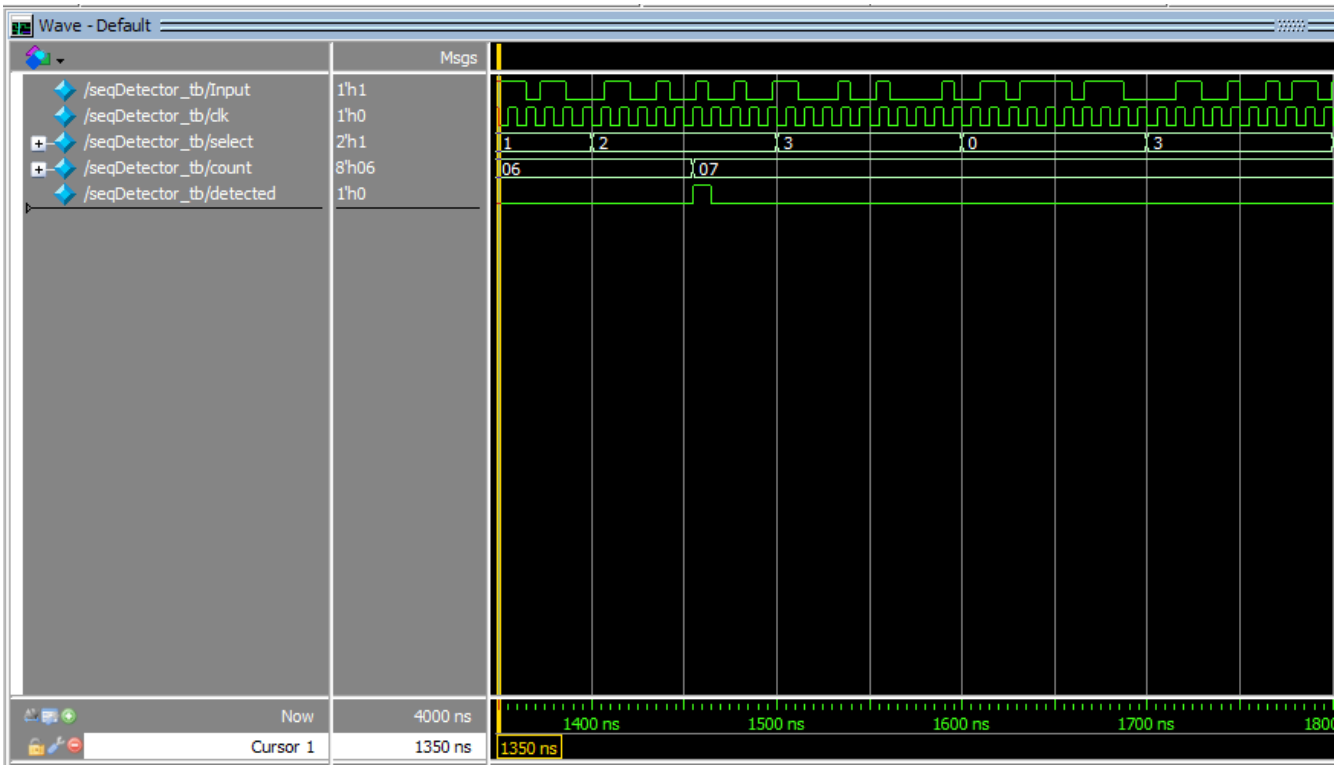
Pic 1 : First 450 ns = [0 , 450 ns]



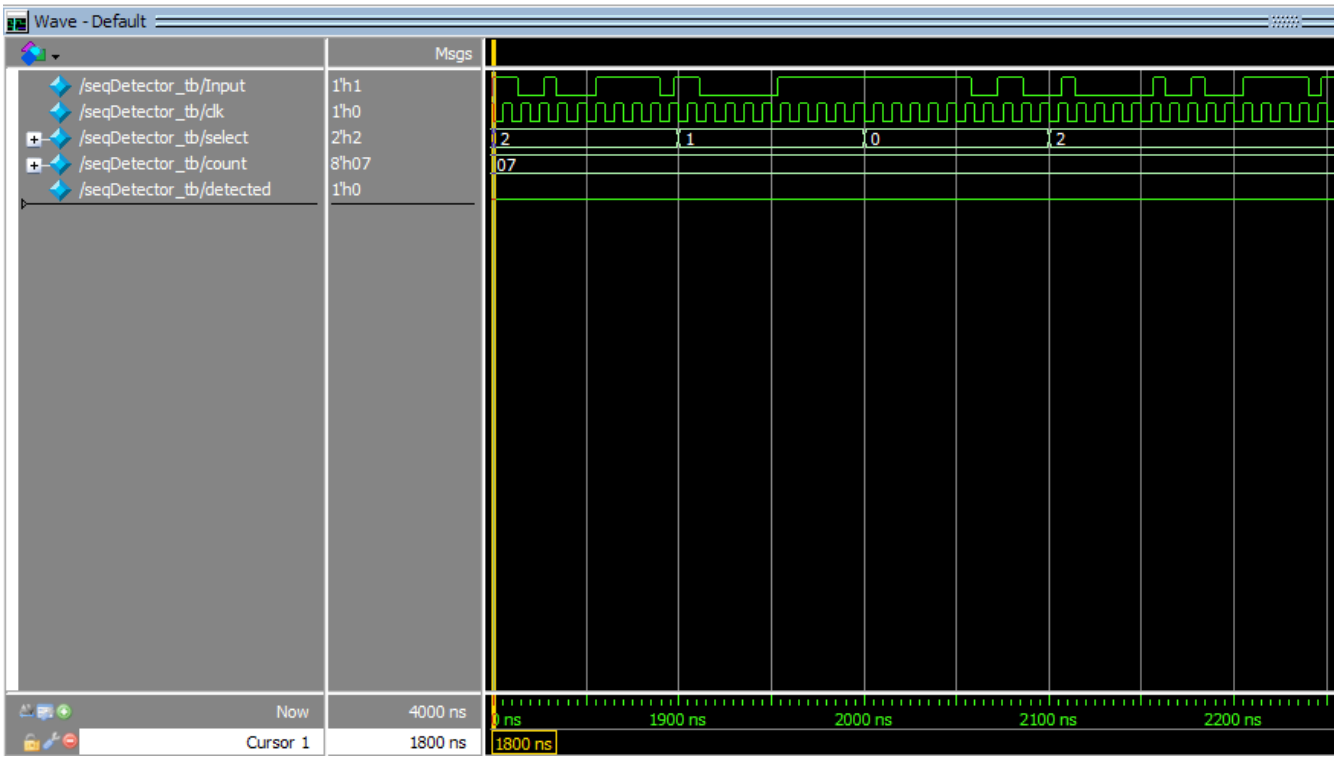
Pic 2 : Second 450 ns = [450 ns , 900 ns]



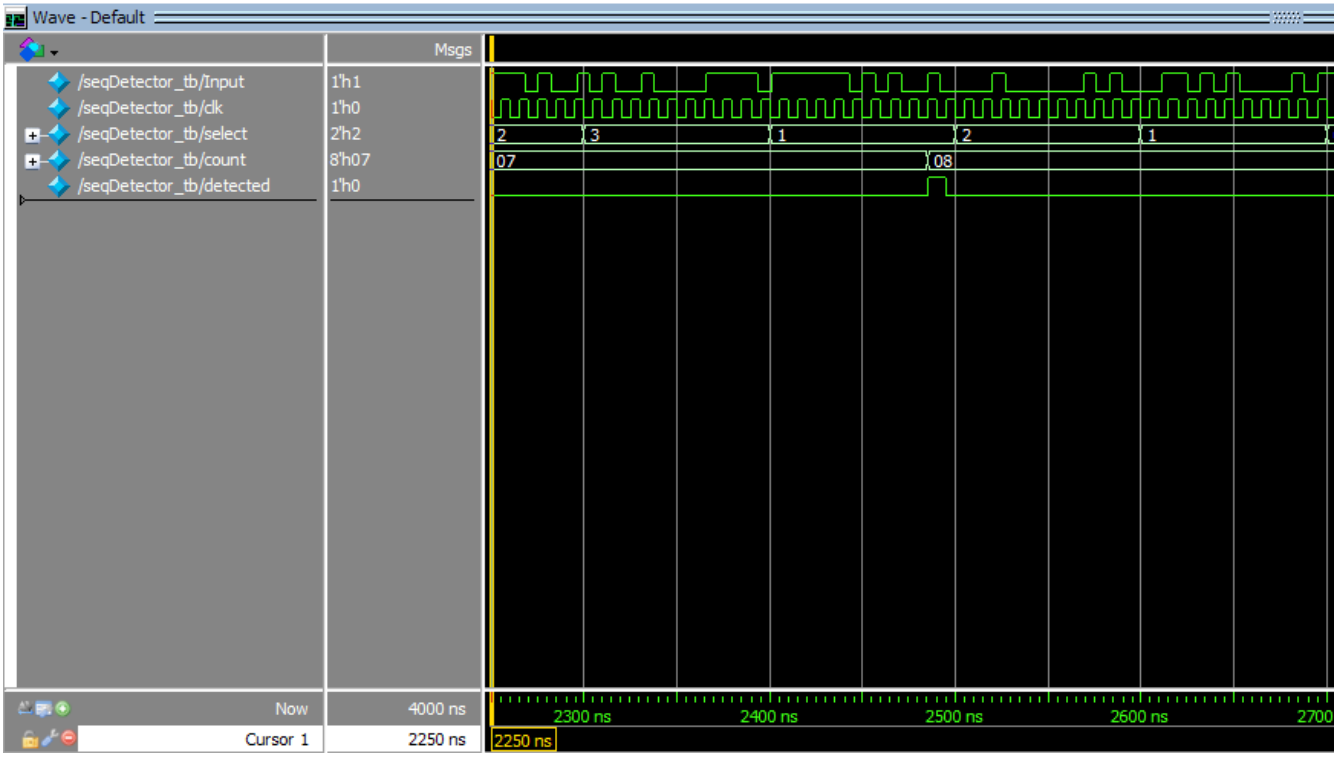
Pic 3 : Third 450 ns = [900 ns , 1350 ns]



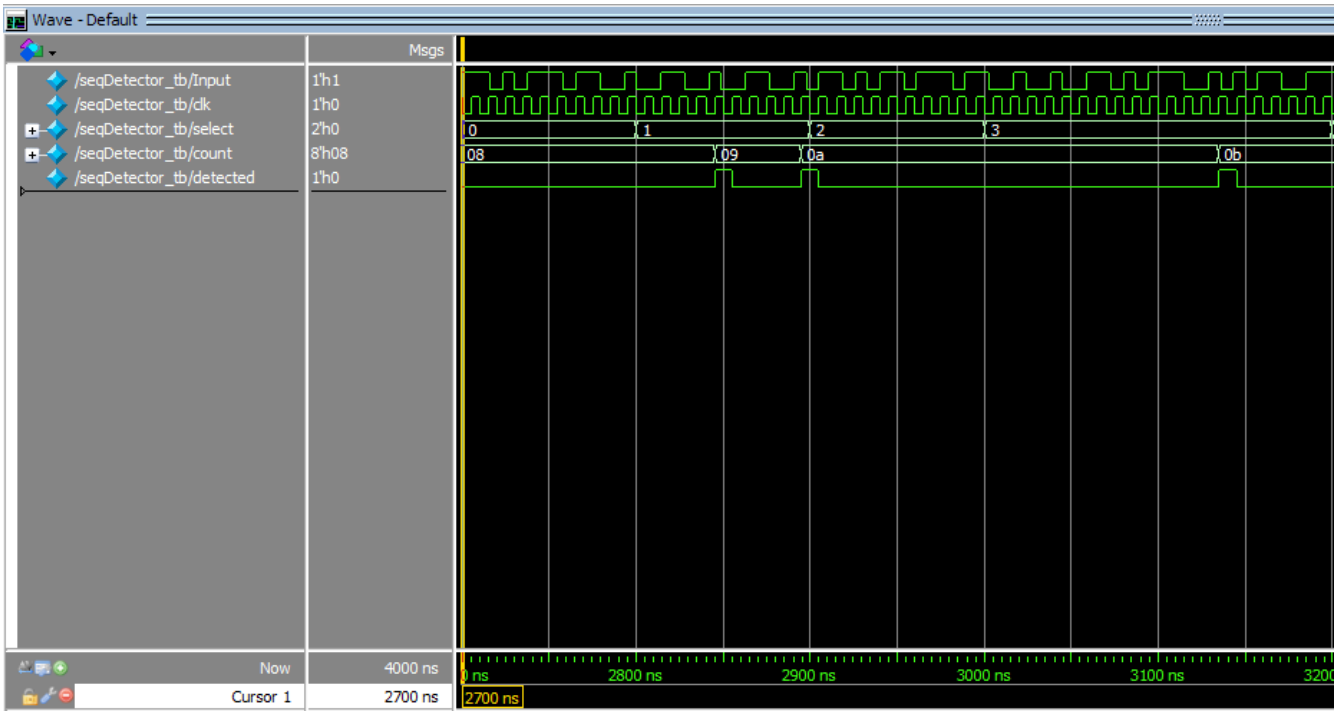
Pic 4 : Fourth 450 ns = [1350 ns , 1800 ns]



Pic 5 : Fifth 450 ns = [1800 ns , 2250 ns]



Pic 6 : Sixth 450 ns = [2250 ns , 2700 ns]



Pic 7 : Seventh 450 ns = [2700 ns , 3200 ns]

به صورت خلاصه :

- شبیه سازی ماژول در مجموع 3700ns به طول انجامید.
- اولین رشته ی دیتکت شده مربوط به سمپل 0110 و در بازه ی [200,250] بوده است (Pic 1)
- دومین رشته ی دیتکت شده مربوط به سمپل 1010 و در بازه ی [750,800] بوده است (Pic 2)
- سومین رشته ی دیتکت شده مربوط به سمپل 0110 و در بازه ی [1050,1100] بوده است (Pic 3)
- چهارمین رشته ی دیتکت شده مربوط به سمپل 1001 و در بازه ی [1150,1200] بوده است (Pic 3)
- پنجمین رشته ی دیتکت شده مربوط به سمپل 0110 و در بازه ی [1300,1350] بوده است (Pic 3)
- ششمین رشته ی دیتکت شده مربوط به سمپل 0110 و در بازه ی [1300,1350] بوده است (Pic 3)
- هفتمین رشته ی دیتکت شده مربوط به سمپل 1010 و در بازه ی [1450,1500] بوده است (Pic 4)
- هشتمین رشته ی دیتکت شده مربوط به سمپل 0110 و در بازه ی [2450,2500] بوده است (Pic 6)
- نهمین رشته ی دیتکت شده مربوط به سمپل 0110 و در بازه ی [2800,2850] بوده است (Pic 7)
- دهمین رشته ی دیتکت شده مربوط به سمپل 0110 و در بازه ی [2850,2900] بوده است (Pic 7)
- یازدهمین رشته ی دیتکت شده مربوط به سمپل 1100 و در بازه ی [3100,3150] بوده است (Pic 7)

سوال سوم:

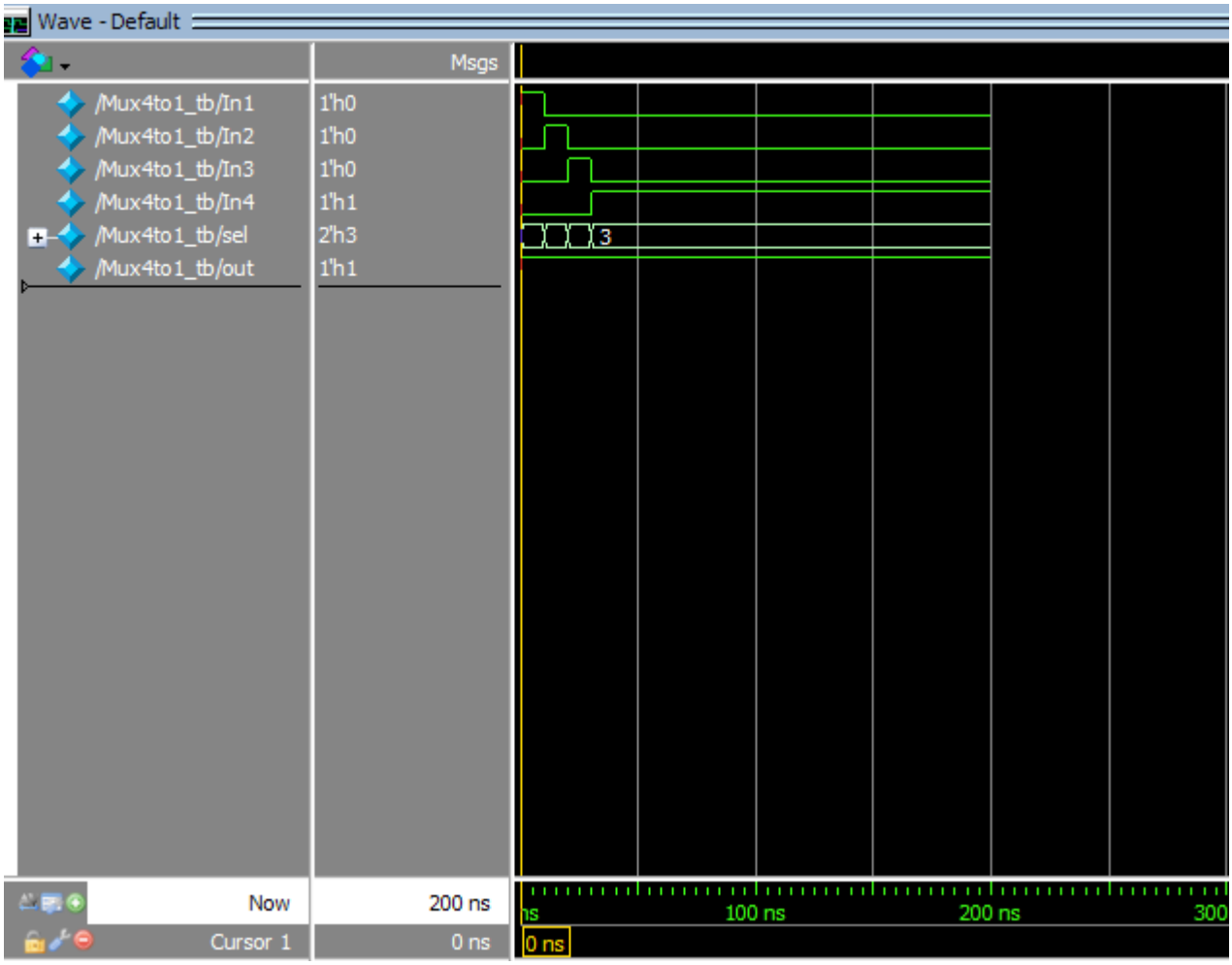
در این سوال میخواهیم یک مالتی پلکسر 16 به 1 با پهنای بیت ورودی 8 بیت طراحی کنیم:

الف) یک مالتی پلکسر 4 به 1 با پهنای بیت ورودی یک بیت طراحی کنید.

ب) با استفاده از مالتی پلکسر های 4 به 1 پیاده سازی شده در قسمت الف، یک مالتی پلکسر 16 به 1 با عرض بیت ورودی 8 بیت طراحی کنید.

**** اسکرین شات های مربوط به اجرای هر یک از testbench ها را در ادامه آورده‌ام...**

الف)



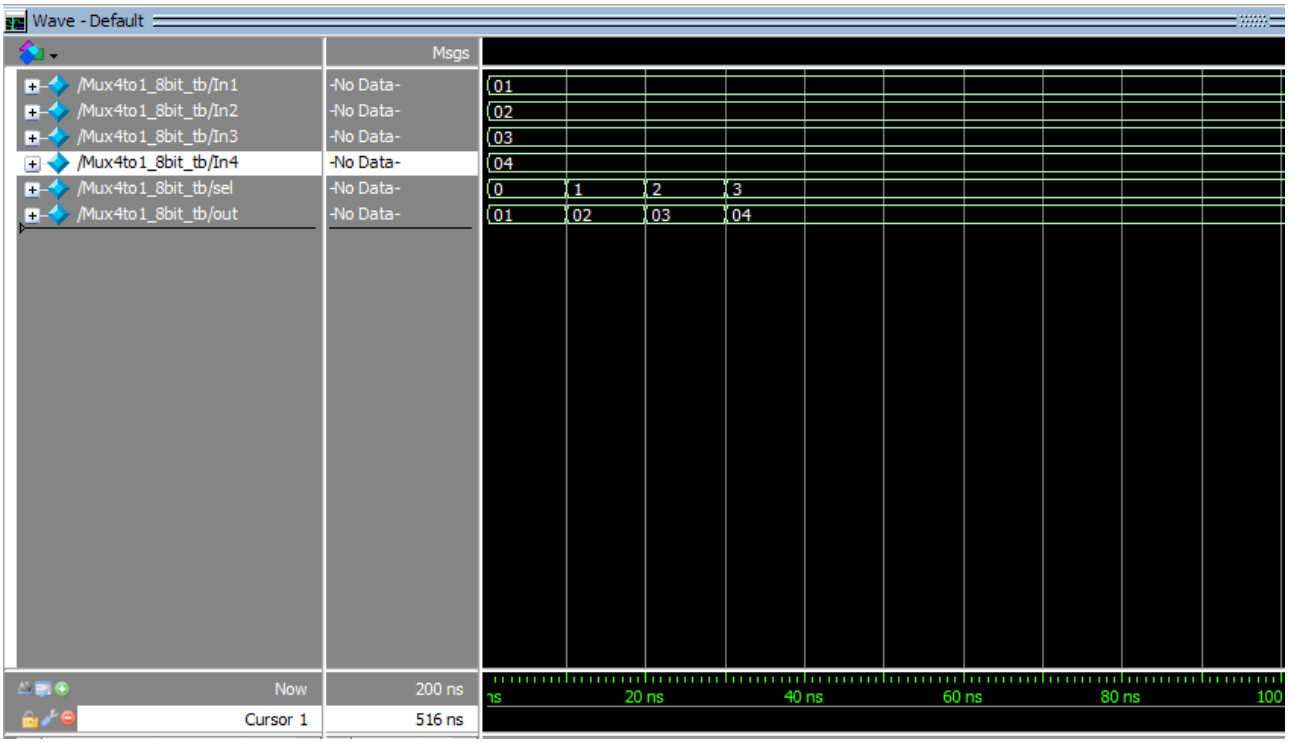
```
Transcript
ModelSim> vsim -voptargs=+acc work.Mux4to1_tb
# vsim -voptargs="+acc" work.Mux4to1_tb
# Start time: 02:36:38 on Mar 25,2020
# ** Note: (vsim-3812) Design is being optimized...
# Loading work.Mux4to1_tb(fast)
# Loading work.Mux4to1_1bit(fast)
add wave -position end  sim:/Mux4to1_tb/In1
add wave -position end  sim:/Mux4to1_tb/In2
add wave -position end  sim:/Mux4to1_tb/In3
add wave -position end  sim:/Mux4to1_tb/In4
add wave -position end  sim:/Mux4to1_tb/sel
add wave -position end  sim:/Mux4to1_tb/out
VSIM 9> run
#                    0 a=1 , b=0 , c=0 , d=0 , select=00 output=1
#                   10 a=0 , b=1 , c=0 , d=0 , select=01 output=1
#                   20 a=0 , b=0 , c=1 , d=0 , select=10 output=1
#                   30 a=0 , b=0 , c=0 , d=1 , select=11 output=1

VSIM 10>
```

ب)

*** توجه : قبل از طراحی و پیاده‌سازی ماژول Mux16×1 ، یک ماژول واسطه تحت عنوان Mux4×1 با عرض بیت 8bit ایجاد کردم که فقط از ماژول Mux4×1 با عرض بیت 1bit ، درحقیقت instant میگیرد و هیچ چیز اضافه تری ندارد(صرفاً برای خواناتر شدن کد ماژول اصلی و تمیزتر شدن کد است!)**

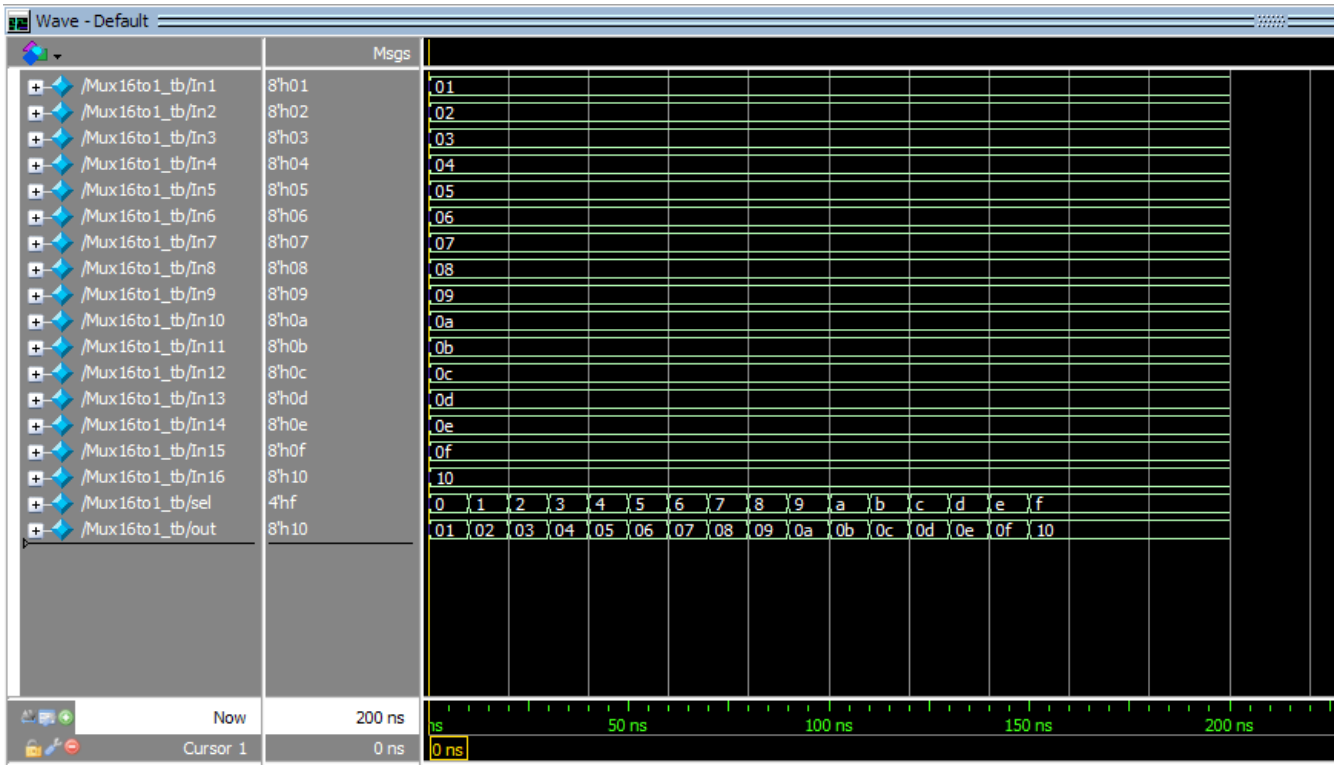
این تصاویر هم مربوط به تست همین ماژول واسط است :



Transcript

```
ModelSim> vsim -voptargs=+acc work.Mux4to1_8bit_tb
# vsim -voptargs="+acc" work.Mux4to1_8bit_tb
# Start time: 02:39:38 on Mar 25,2020
# ** Note: (vsim-8009) Loading existing optimized design _opt1
# Loading work.Mux4to1_8bit_tb(fast)
# Loading work.Mux4to1_8bit(fast)
# Loading work.Mux4to1_1bit(fast)
add wave -position end  sim:/Mux4to1_8bit_tb/In1
add wave -position end  sim:/Mux4to1_8bit_tb/In2
add wave -position end  sim:/Mux4to1_8bit_tb/In3
add wave -position end  sim:/Mux4to1_8bit_tb/In4
add wave -position end  sim:/Mux4to1_8bit_tb/sel
add wave -position end  sim:/Mux4to1_8bit_tb/out
VSIM 18> run
#                    0 a=00000001 , b=00000010 , c=00000011 , d=00000100 , select=00 output=00000001
#                   10 a=00000001 , b=00000010 , c=00000011 , d=00000100 , select=01 output=00000010
#                   20 a=00000001 , b=00000010 , c=00000011 , d=00000100 , select=10 output=00000011
#                   30 a=00000001 , b=00000010 , c=00000011 , d=00000100 , select=11 output=00000100

VSIM 19>
```



Transcript

```
VSIM 61> run
#                    0 a=00000001 , b=00000010 , c=00000011 , d=00000100 , e=00000101 , f=00000110 , g=00000111 , h=00001000 , i=00001001 , j=00001010 , k=00001011 , l=00001100 , m=00001101 , n=00001110 , o=00001111 , p=00010000 , select=0000 , output=00000001
#                   10 a=00000001 , b=00000010 , c=00000011 , d=00000100 , e=00000101 , f=00000110 , g=00000111 , h=00001000 , i=00001001 , j=00001010 , k=00001011 , l=00001100 , m=00001101 , n=00001110 , o=00001111 , p=00010000 , select=0001 , output=00000010
#                   20 a=00000001 , b=00000010 , c=00000011 , d=00000100 , e=00000101 , f=00000110 , g=00000111 , h=00001000 , i=00001001 , j=00001010 , k=00001011 , l=00001100 , m=00001101 , n=00001110 , o=00001111 , p=00010000 , select=0010 , output=00000011
#                   30 a=00000001 , b=00000010 , c=00000011 , d=00000100 , e=00000101 , f=00000110 , g=00000111 , h=00001000 , i=00001001 , j=00001010 , k=00001011 , l=00001100 , m=00001101 , n=00001110 , o=00001111 , p=00010000 , select=0011 , output=00000100
#                   40 a=00000001 , b=00000010 , c=00000011 , d=00000100 , e=00000101 , f=00000110 , g=00000111 , h=00001000 , i=00001001 , j=00001010 , k=00001011 , l=00001100 , m=00001101 , n=00001110 , o=00001111 , p=00010000 , select=0100 , output=00000101
#                   50 a=00000001 , b=00000010 , c=00000011 , d=00000100 , e=00000101 , f=00000110 , g=00000111 , h=00001000 , i=00001001 , j=00001010 , k=00001011 , l=00001100 , m=00001101 , n=00001110 , o=00001111 , p=00010000 , select=0101 , output=00000110
#                   60 a=00000001 , b=00000010 , c=00000011 , d=00000100 , e=00000101 , f=00000110 , g=00000111 , h=00001000 , i=00001001 , j=00001010 , k=00001011 , l=00001100 , m=00001101 , n=00001110 , o=00001111 , p=00010000 , select=0110 , output=00000111
#                   70 a=00000001 , b=00000010 , c=00000011 , d=00000100 , e=00000101 , f=00000110 , g=00000111 , h=00001000 , i=00001001 , j=00001010 , k=00001011 , l=00001100 , m=00001101 , n=00001110 , o=00001111 , p=00010000 , select=0111 , output=00001000
#                   80 a=00000001 , b=00000010 , c=00000011 , d=00000100 , e=00000101 , f=00000110 , g=00000111 , h=00001000 , i=00001001 , j=00001010 , k=00001011 , l=00001100 , m=00001101 , n=00001110 , o=00001111 , p=00010000 , select=1000 , output=00001001
#                   90 a=00000001 , b=00000010 , c=00000011 , d=00000100 , e=00000101 , f=00000110 , g=00000111 , h=00001000 , i=00001001 , j=00001010 , k=00001011 , l=00001100 , m=00001101 , n=00001110 , o=00001111 , p=00010000 , select=1001 , output=00001010
#                  100 a=00000001 , b=00000010 , c=00000011 , d=00000100 , e=00000101 , f=00000110 , g=00000111 , h=00001000 , i=00001001 , j=00001010 , k=00001011 , l=00001100 , m=00001101 , n=00001110 , o=00001111 , p=00010000 , select=1010 , output=00001011
#                  110 a=00000001 , b=00000010 , c=00000011 , d=00000100 , e=00000101 , f=00000110 , g=00000111 , h=00001000 , i=00001001 , j=00001010 , k=00001011 , l=00001100 , m=00001101 , n=00001110 , o=00001111 , p=00010000 , select=1011 , output=00001100
#                  120 a=00000001 , b=00000010 , c=00000011 , d=00000100 , e=00000101 , f=00000110 , g=00000111 , h=00001000 , i=00001001 , j=00001010 , k=00001011 , l=00001100 , m=00001101 , n=00001110 , o=00001111 , p=00010000 , select=1100 , output=00001101
#                  130 a=00000001 , b=00000010 , c=00000011 , d=00000100 , e=00000101 , f=00000110 , g=00000111 , h=00001000 , i=00001001 , j=00001010 , k=00001011 , l=00001100 , m=00001101 , n=00001110 , o=00001111 , p=00010000 , select=1101 , output=00001110
#                  140 a=00000001 , b=00000010 , c=00000011 , d=00000100 , e=00000101 , f=00000110 , g=00000111 , h=00001000 , i=00001001 , j=00001010 , k=00001011 , l=00001100 , m=00001101 , n=00001110 , o=00001111 , p=00010000 , select=1110 , output=00001111
#                  150 a=00000001 , b=00000010 , c=00000011 , d=00000100 , e=00000101 , f=00000110 , g=00000111 , h=00001000 , i=00001001 , j=00001010 , k=00001011 , l=00001100 , m=00001101 , n=00001110 , o=00001111 , p=00010000 , select=1111 , output=00010000

VSIM 62>
```

کدها

سوال اول:

در این سوال می‌توانید از هر یک از سطوح Gate، Data flow و یا Behavioral استفاده کنید. در نهایت لازم است برای هر یک از ماژول‌ها یک test bench به گونه‌ای نوشته شود که حالات ممکن برای آن ماژول را دربر بگیرد. هر یک از این تست بنچ‌ها باید دربردارنده‌ی حد اقل سه جفت عدد ورودی متفاوت برای ماژول باشند.

الف)

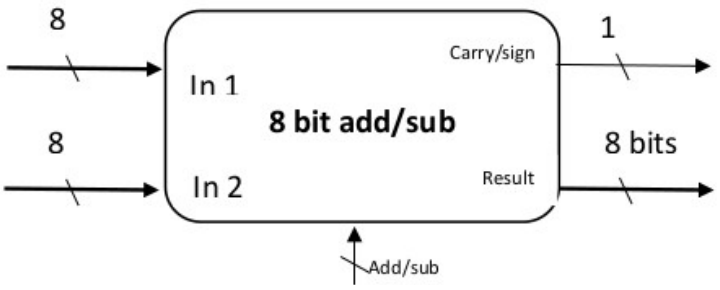
یک full adder تک بیتی طراحی نمایید.

ب)

با استفاده از full adder 1 بیتی‌ای که در قسمت قبل طراحی کردید، یک full adder 8 بیتی بسازید. در این قسمت فقط مجاز هستید که از instant ماژول طراحی شده در قسمت قبل استفاده کنید.

ج)

کد قسمت (ب) را به گونه ای تغییر دهید که تبدیل به یک جمع و تفریق کننده‌ی 8 بیتی شود. یک جمع و تفریق کننده‌ی هشت بیتی ماژولی است که دارای دو ورودی 8 بیتی برای اعداد ورودی و یک ورودی یک بیتی برای تعیین جمع یا تفریق کردن اعداد ورودی است. ماژول با یک شدن پایه کنترلی عمل جمع، و با صفر شدن عمل تفریق را انجام می دهد.



پاسخ :

الف)

```
module FullAdder1bit (input a,b,cin, output sum , cout);
	assign {cout,sum} = a + b + cin;
endmodule
```

```
module FullAdder1bit_tb;
	//variables
	reg a,b;
	reg cin;
	wire sum;
	wire cout;

	//call comparetor
	FullAdder1bit FA_Obj(.a(a),.b(b),.cin(cin),.sum(sum),.cout(cout));

	//Test inputs
	initial begin
	$monitor("%d a=%b , b=%b , cin=%b , sum=%b , cout=%b ",
		$time,a,b,cin,sum,cout);
	#20
	a = 1'b1;
	b = 1'b1;
	cin = 1'b1;
	#20
	a = 1'b1;
	b = 1'b1;
	cin = 1'b0;
	#20
	a = 1'b0;
	b = 1'b1;
	cin = 1'b0;
	#20
	a = 1'b0;
	b = 1'b1;
	cin = 1'b1;
	#20
	a = 1'b0;
	b = 1'b0;
	cin = 1'b1;
	#20
	a = 1'b0;
	b = 1'b0;
	cin = 1'b0;
	end
endmodule
```

(ب

```
module FullAdder8bit (input [7:0] a, b , input cin , output [7:0] sum, output cout);
	wire cout1,cout2,cout3,cout4,cout5,cout6,cout7;
	FullAdder1bit U1 (.a(a[0]),.b(b[0]),.cin(cin),.sum(sum[0]),.cout(cout1));
	FullAdder1bit U2 (.a(a[1]),.b(b[1]),.cin(cout1),.sum(sum[1]),.cout(cout2));
	FullAdder1bit U3 (.a(a[2]),.b(b[2]),.cin(cout2),.sum(sum[2]),.cout(cout3));
	FullAdder1bit U4 (.a(a[3]),.b(b[3]),.cin(cout3),.sum(sum[3]),.cout(cout4));
	FullAdder1bit U5 (.a(a[4]),.b(b[4]),.cin(cout4),.sum(sum[4]),.cout(cout5));
	FullAdder1bit U6 (.a(a[5]),.b(b[5]),.cin(cout5),.sum(sum[5]),.cout(cout6));
	FullAdder1bit U7 (.a(a[6]),.b(b[6]),.cin(cout6),.sum(sum[6]),.cout(cout7));
	FullAdder1bit U8 (.a(a[7]),.b(b[7]),.cin(cout7),.sum(sum[7]),.cout(cout));
endmodule
```

```
module FullAdder8bit_tb;
	//variables
	reg [7:0] a,b;
	reg cin;
	wire [7:0] sum;
	wire cout;

	//call comparetor
	FullAdder8bit FA_Obj(.a(a),.b(b),.cin(cin),.sum(sum),.cout(cout));

	//Test inputs
	initial begin
	$monitor("%d a=%b , b=%b , cin=%b , sum=%b , cout=%b ",
		$time,a,b,cin,sum,cout);
	#20
	a = 8'hff;
	b = 8'hff;
	cin = 1'b1;
	#20
	a = 8'hff;
	b = 8'hff;
	cin = 1'b0;
	#20
	a = 8'h00;
	b = 8'h0f;
	cin = 1'b0;
	#20
	a = 8'h00;
	b = 8'h0f;
	cin = 1'b1;
	#20
	a = 8'h56;
	b = 8'h39;
	cin = 1'b1;
	end
endmodule
```

ج)

```
module AddSub8bit (input [7:0] In1, In2 , input mode , output [7:0] result, output carry_sign);
// mode : 1 for adder , 0 for subtractor
	wire cout1,cout2,cout3,cout4,cout5,cout6,cout7;
	wire [7:0] a,b;
	assign a = In1;
	assign b = In2^(mode?8'h00:8'hff);
	FullAdder1bit U1 (.a(a[0]),.b(b[0]),.cin(~mode),.sum(result[0]),.cout(cout1));
	FullAdder1bit U2 (.a(a[1]),.b(b[1]),.cin(cout1),.sum(result[1]),.cout(cout2));
	FullAdder1bit U3 (.a(a[2]),.b(b[2]),.cin(cout2),.sum(result[2]),.cout(cout3));
	FullAdder1bit U4 (.a(a[3]),.b(b[3]),.cin(cout3),.sum(result[3]),.cout(cout4));
	FullAdder1bit U5 (.a(a[4]),.b(b[4]),.cin(cout4),.sum(result[4]),.cout(cout5));
	FullAdder1bit U6 (.a(a[5]),.b(b[5]),.cin(cout5),.sum(result[5]),.cout(cout6));
	FullAdder1bit U7 (.a(a[6]),.b(b[6]),.cin(cout6),.sum(result[6]),.cout(cout7));
	FullAdder1bit U8 (.a(a[7]),.b(b[7]),.cin(cout7),.sum(result[7]),.cout(carry_sign));
endmodule
```

```
module AddSub8bit_tb;
	//variables
	reg [7:0] a,b;
	reg mode;
	wire [7:0] result;
	wire carry_sign;

	//call comparetor
	AddSub8bit AS_Obj(.In1(a),.In2(b),.mode(mode),.result(result),.carry_sign(carry_sign));


	//Test inputs
	initial begin

	$monitor("%d Input1=%b , Input2=%b , Mode=%b , Result=%b , Carry/Sign=%b",
		$time,a,b,mode,result,carry_sign);
	#20
	a = 8'hff;
	b = 8'hff;
	mode = 1'b1;//Add
	#20
	a = 8'hff;
	b = 8'hff;
	mode = 1'b0;//Sub
	#20
```

```
        a = 8'h00;
        b = 8'h0f;
        mode = 1'b0;//Sub
        #20
        a = 8'h00;
        b = 8'h0f;
        mode = 1'b1;//Add
        #20
        a = 8'h56;
        b = 8'h39;
        mode = 1'b1;//Add
        #20
        a = 8'h56;
        b = 8'h39;
        mode = 1'b0;//Sub
        end
endmodule
```

سوال دوم:

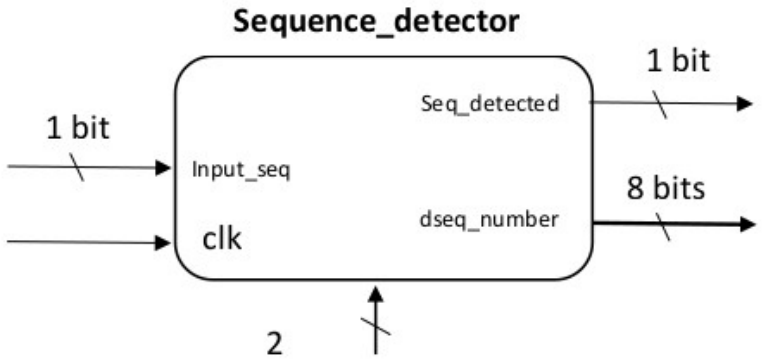
مطابق شکل زیر یک ماژول sequence detector درسطح Behavioral تعریف کنید. عملکرد ورودی ها و خروجی های این ماژول به شرح زیر است:

Input_seq: این ورودی تک بیتی در واقع رشته ای از 0 و1 ها است که به صورت سریال وارد ماژول میشود.

Seq_select: این ورودی 2 بیتی یکی از رشته های 1001، 0110، 1010 و یا 1100 را برای جستجو انتخاب می‌کند به این معنی که مثلا اگر seq_select=2 باشد، ماژول به دنبال رشته 0110 در رشته اعداد ورودی میگردد.

Seq_detected: این پایه یک بیتی پس از هر بار شناسایی رشته مورد نظر یک پالس ساعت یک میشود و سپس مجددا صفر میشود.

Dseq_number: این خروجی 8 بیتی تعداد رشته های تشخیص داده شده را نشان میدهد.



```
module seqDetector(input Input_seq,clk, input [1:0] seq_select,
          output reg seq_detected, output reg [7:0] dseq_number=0);

        parameter [2:0] s0=3'b000,s1=3'b001,s2=3'b010,s3=3'b011,s4=3'b100;
        reg [2:0] current=s0;

        always @(posedge clk) begin
        case(seq_select)
                0:case(current) // 1001
                        s0: if(Input_seq) current<=s1;else current<=s0;
                        s1: if(Input_seq) current<=s1;else current<=s2;
                        s2: if(Input_seq) current<=s1;else current<=s3;
                        s3: if(Input_seq) current<=s4;else current<=s0;
                        s4: if(Input_seq) current<=s1;else current<=s2;
                        endcase
                1:case(current) // 0110
                        s0: if(Input_seq) current<=s0;else current<=s1;
                        s1: if(Input_seq) current<=s2;else current<=s1;
                        s2: if(Input_seq) current<=s3;else current<=s1;
                        s3: if(Input_seq) current<=s0;else current<=s4;
                        s4: if(Input_seq) current<=s2;else current<=s1;
                        endcase
                2:case(current) // 1010
                        s0: if(Input_seq) current<=s1;else current<=s0;
                        s1: if(Input_seq) current<=s1;else current<=s2;
                        s2: if(Input_seq) current<=s3;else current<=s0;
                        s3: if(Input_seq) current<=s1;else current<=s4;
                        s4: if(Input_seq) current<=s3;else current<=s0;
                        endcase
                3:case(current) // 1100
                        s0: if(Input_seq) current<=s1;else current<=s0;
                        s1: if(Input_seq) current<=s2;else current<=s0;
                        s2: if(Input_seq) current<=s2;else current<=s3;
                        s3: if(Input_seq) current<=s1;else current<=s4;
                        s4: if(Input_seq) current<=s1;else current<=s0;
                        endcase
        endcase
        end

        always @(seq_select) begin
        current<=s0;
        end
        always @(seq_detected) begin
        dseq_number<=(seq_detected)?dseq_number+1:dseq_number;
        end
        always @(posedge clk) begin
        seq_detected<=(current==s4)?1:0;
        end
endmodule
```

```
module seqDetector_tb();
	reg Input,clk=0;
	reg [1:0] select=0;
	wire [7:0] count;
	wire detected;

	seqDetector U1
(.Input_seq(Input),.clk(clk),.seq_select(select),.seq_detected(detected),.dseq_number(count));

	always #5 clk=~clk;
	always #7 Input=$random;
	always #100 select=$random;

endmodule
```

سوال سوم:

در این سوال میخواهیم یک مالتی پلکسر 16 به 1 با پهنای بیت ورودی 8 بیت طراحی کنیم:

الف) یک مالتی پلکسر 4 به 1 با پهنای بیت ورودی یک بیت طراحی کنید.

ب) با استفاده از مالتی پلکسر های 4 به 1 پیاده سازی شده در قسمت الف، یک مالتی پلکسر 16 به 1 با عرض بیت ورودی 8 بیت طراحی کنید.

الف)

```
module Mux4to1_1bit(input a,b,c,d, input [1:0] select, output outBus);
	assign outBus = select[1]?(select[0]?d:c):(select[0]?b:a);
endmodule
```

```
module Mux4to1_tb;
	reg In1,In2,In3,In4;
	reg [1:0] sel;
	wire out;
	Mux4to1_1bit U1(.a(In1),
			.b(In2),
			.c(In3),
			.d(In4),
			.select(sel),
			.outBus(out));
	initial begin
	$monitor("%d a=%b , b=%b , c=%b , d=%b , select=%b output=%b ",
			 $time,In1,In2,In3,In4,sel,out);
	In1=1;
	In2=0;
	In3=0;
	In4=0;
	sel=0;
	#10
	In1=0;
	In2=1;
	In3=0;
	In4=0;
	sel=1;
	#10
	In1=0;
	In2=0;
	In3=1;
	In4=0;
	sel=2;
	#10
	In1=0;
	In2=0;
	In3=0;
	In4=1;
	sel=3;
	end
endmodule
```

(ب

```
module Mux4to1_8bit (input [7:0] a,b,c,d, input [1:0] select, output [7:0] outBus);
	Mux4to1_1bit U0 (.a(a[0]),.b(b[0]),.c(c[0]),.d(d[0]),.select(select),.outBus(outBus[0]));
	Mux4to1_1bit U1 (.a(a[1]),.b(b[1]),.c(c[1]),.d(d[1]),.select(select),.outBus(outBus[1]));
	Mux4to1_1bit U2 (.a(a[2]),.b(b[2]),.c(c[2]),.d(d[2]),.select(select),.outBus(outBus[2]));
	Mux4to1_1bit U3 (.a(a[3]),.b(b[3]),.c(c[3]),.d(d[3]),.select(select),.outBus(outBus[3]));
	Mux4to1_1bit U4 (.a(a[4]),.b(b[4]),.c(c[4]),.d(d[4]),.select(select),.outBus(outBus[4]));
	Mux4to1_1bit U5 (.a(a[5]),.b(b[5]),.c(c[5]),.d(d[5]),.select(select),.outBus(outBus[5]));
	Mux4to1_1bit U6 (.a(a[6]),.b(b[6]),.c(c[6]),.d(d[6]),.select(select),.outBus(outBus[6]));
	Mux4to1_1bit U7 (.a(a[7]),.b(b[7]),.c(c[7]),.d(d[7]),.select(select),.outBus(outBus[7]));
endmodule


module Mux16to1_8bit(input [7:0] a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p, input [3:0] select, output [7:0]
outBus);
	wire [7:0] out [3:0];
	Mux4to1_8bit U0 (.a(a),.b(b),.c(c),.d(d),.select(select[1:0]),.outBus(out[0]));
	Mux4to1_8bit U1 (.a(e),.b(f),.c(g),.d(h),.select(select[1:0]),.outBus(out[1]));
	Mux4to1_8bit U2 (.a(i),.b(j),.c(k),.d(l),.select(select[1:0]),.outBus(out[2]));
	Mux4to1_8bit U3 (.a(m),.b(n),.c(o),.d(p),.select(select[1:0]),.outBus(out[3]));
	Mux4to1_8bit U4
(.a(out[0]),.b(out[1]),.c(out[2]),.d(out[3]),.select(select[3:2]),.outBus(outBus));
endmodule
```

```
module Mux4to1_8bit_tb;
	reg [7:0] In1=1,In2=2,In3=3,In4=4;
	reg [1:0] sel;
	wire [7:0] out;
	Mux4to1_8bit U1(.a(In1),
			.b(In2),
			.c(In3),
			.d(In4),
			.select(sel),
			.outBus(out));
	initial begin
	$monitor("%d a=%b , b=%b , c=%b , d=%b , select=%b output=%b ",
			 $time,In1,In2,In3,In4,sel,out);

	sel=0;
	#10
	sel=1;
	#10
	sel=2;
	#10
	sel=3;

	end
endmodule
```

```
module Mux16to1_tb;
        reg [7:0] In1=1,In2=2,In3=3,In4=4,
          In5=5,In6=6,In7=7,In8=8,
          In9=9,In10=10,In11=11,
          In12=12,In13=13,In14=14,
          In15=15,In16=16;
        reg [3:0] sel;
        wire [7:0] out;
        Mux16to1_8bit U1(.a(In1),.b(In2),.c(In3),.d(In4),
                .e(In5),.f(In6),.g(In7),.h(In8),
                .i(In9),.j(In10),.k(In11),.l(In12),
                .m(In13),.n(In14),.o(In15),.p(In16),
                .select(sel),
                .outBus(out));
        initial begin
        $monitor("%d a=%b , b=%b , c=%b , d=%b , e=%b , f=%b , g=%b , h=%b , i=%b , j=%b , k=
%b , l=%b , m=%b , n=%b , o=%b , p=%b , select=%b , output=%b ",
                 $time,In1,In2,In3,In4,In5,In6,In7,In8,In9,In10,In11,In12,In13,In14,In15,In16,sel,out);

        sel=0;
        #10
        sel=1;
        #10
        sel=2;
        #10
        sel=3;
        #10
        sel=4;
        #10
        sel=5;
        #10
        sel=6;
        #10
        sel=7;
        #10
        sel=8;
        #10
        sel=9;
        #10
        sel=10;
        #10
        sel=11;
        #10
        sel=12;
        #10
        sel=13;
        #10
        sel=14;
        #10
        sel=15;
        end
endmodule
```

فایل‌های ضمیمه شده :

1. فایل های مربوط به سؤال اول :
    i. FullAdder1bit.v
    ii. FA1bit_tb.v
    iii. FullAdder8bit.v
    iv. FA8bit_tb.v
    v. AddSub8bit.v
    vi. AddSub8bit_tb.v
    vii. Screenshots
2. فایل های مربوط به سؤال دوم :
    i. seqDetector.v
    ii. seqDetector_tb.v
    iii. Screenshots
3. فایل های مربوط به سؤال سوم :
    i. Mux4to1_1bit.v
    ii. Mux4to1_tb.v
    iii. Mux16to1_8bit.v
    iv. Mux16to1_tb.v
    v. Screenshots