

به نام خدا



تکلیف سری دوم fpga

محدثه غفوری (9632133)

سوال 1) مطابق سوال اگر $\text{ctrl}=1$ باشد یعنی شیفت به سمت راست باشد و بیت های خارج شده بصورت چرخشی به ابتدای ورودی بروند داریم

11010010 shift_mag=3 >> out=01011010

در ابتدا اگر به اندازه ی تعداد شیفت مورد نظر یعنی **shift_mag** ورودی را به سمت راست شیفت بدھیم داریم

00011010

حال اگر ورودی را به اندازه ی **8-shift_mag** به سمت چپ شیفت بدھیم داریم

01000000

حال اگر نتیجه دو مرحله ی بالا را باهم **Or** کنیم خروجی مورد نظر بدست می اید

(00011010) | (01000000) = 01011010

اگر $\text{ctrl}=0$ باشد یعنی شیفت به سمت چپ باشد و بیت های خارج شده از سمت راست کنار عدد قرار بگیرند داریم

11101001 shift_mag=2 >> out=10100111

در ابتدا اگر به اندازه ی تعداد شیفت مورد نظر یعنی **shift_mag** ورودی را به سمت چپ شیفت بدھیم داریم

10100100

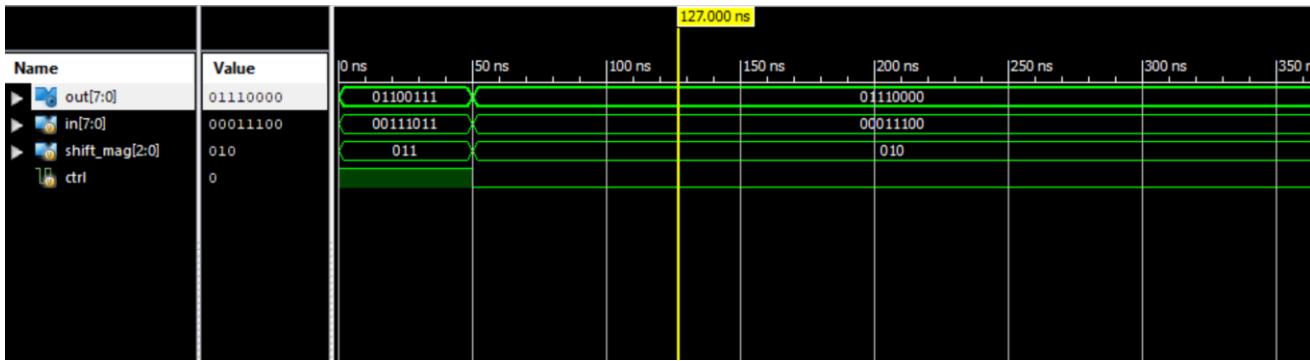
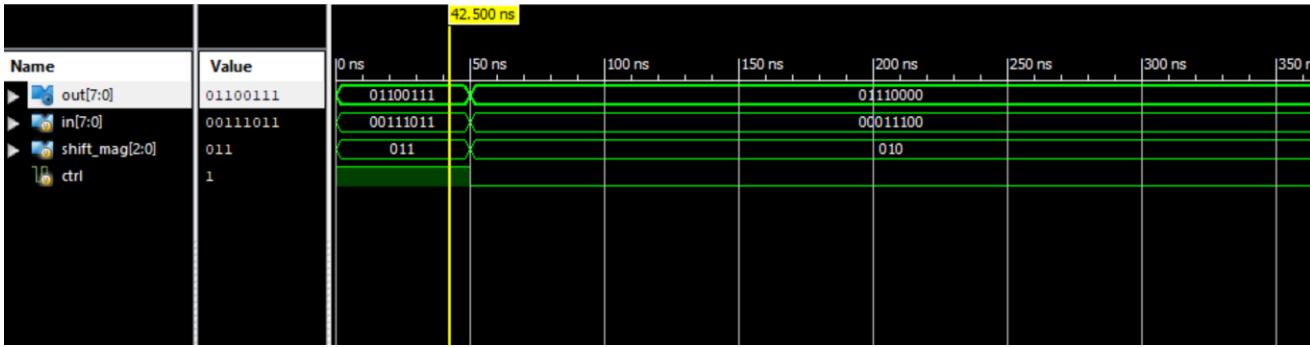
حال اگر ورودی را به اندازه ی **8-shift_mag** به سمت راست شیفت بدھیم داریم

00000011

حال اگر نتیجه دو مرحله ی بالا را باهم **Or** کنیم خروجی مورد نظر بدست می اید

(10100100) | (00000011) = 10100111

نتایج شبیه سازی و waveform ها

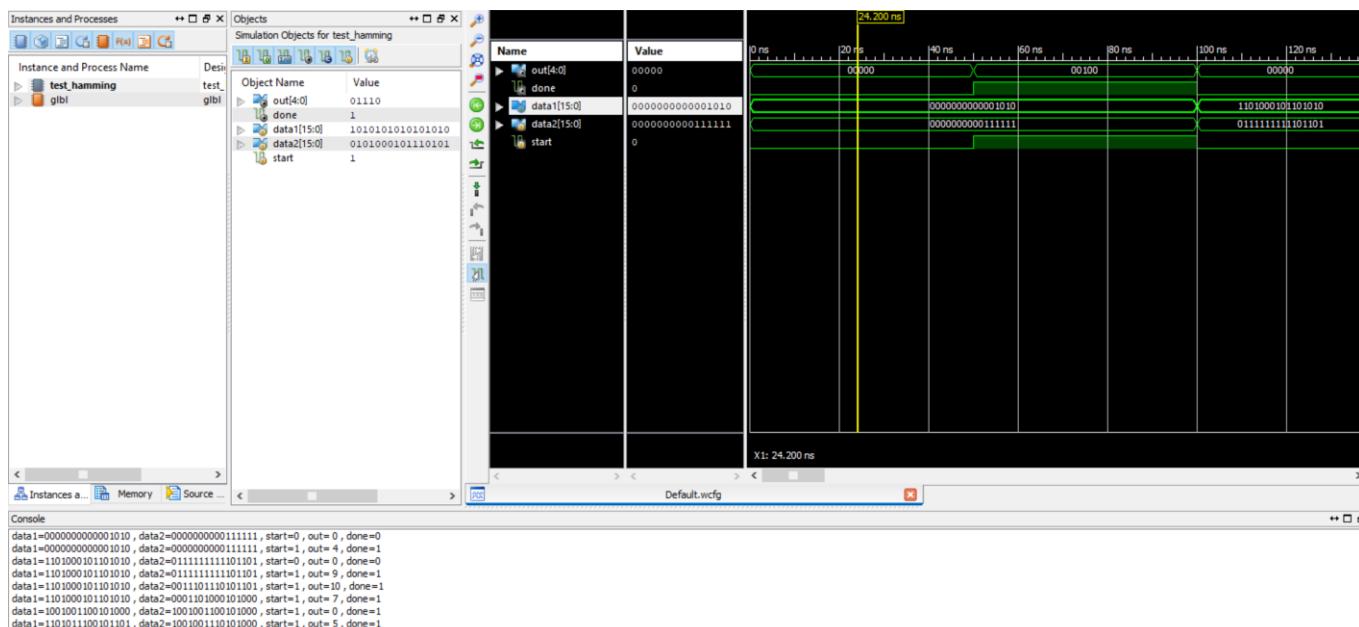


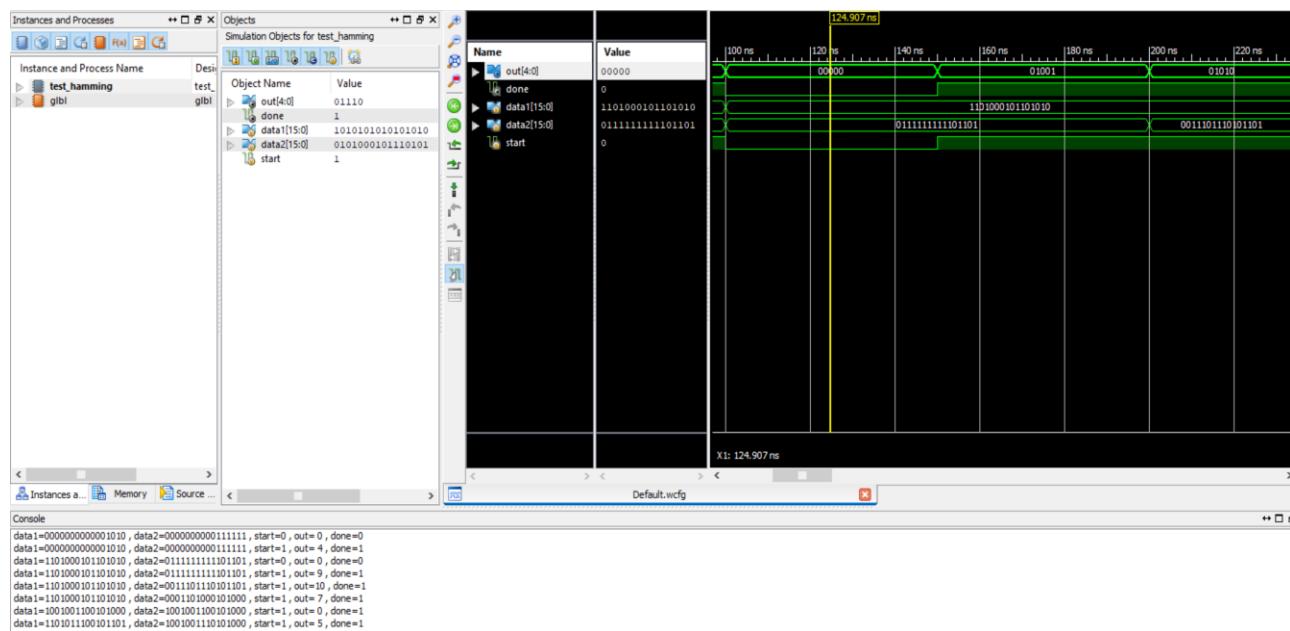
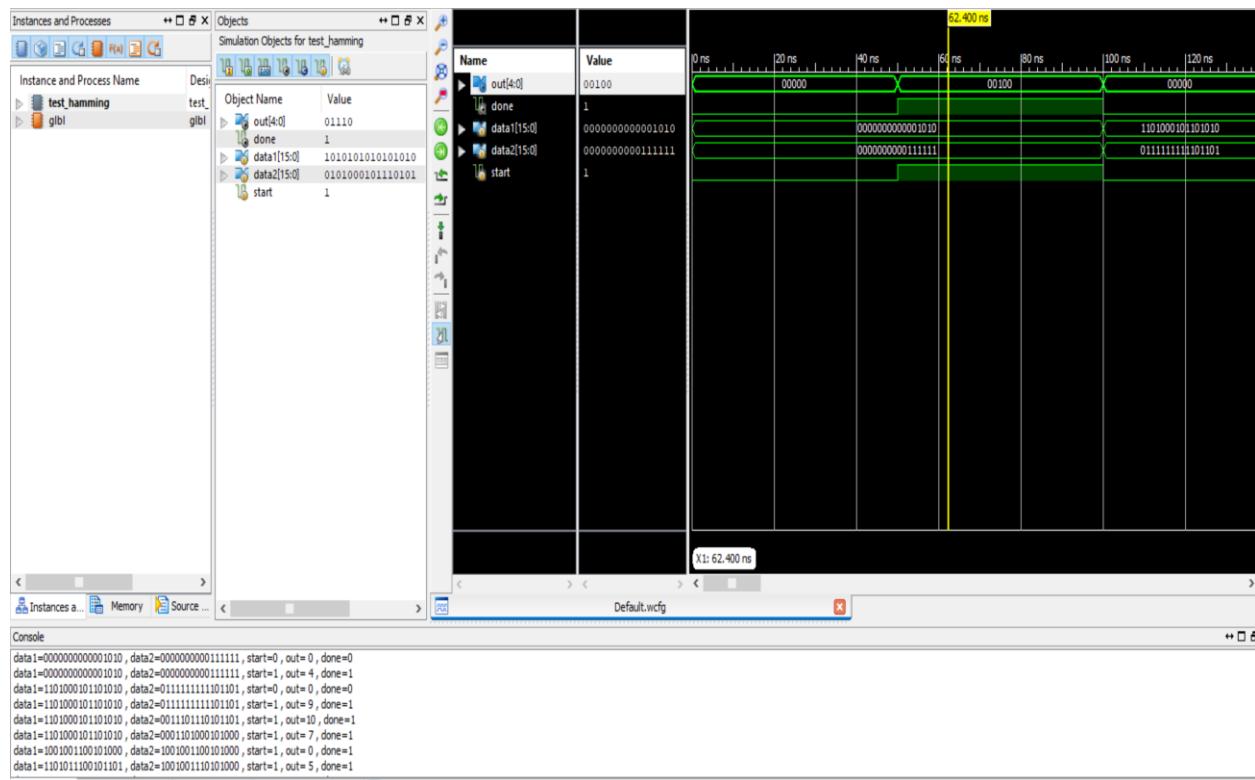
سؤال 2) منظور از فاصله همینگ 2 تعداد بیت های متفاوت دو عدد در مبنا دو است . بدین منظور دو ورودی **data1** ، **data2** را بصورت متناظر بیت هایشان را **xor** میکنیم و در یک متغیر 16 بیتی بنام **w1** میریزیم تا بصورت متناظر تعداد بیت های متفاوت را بدانیم (بیت های یک متغیر **w1** نشان دهنده متفاوت بودن بیت های متناظر **data1,data2** است و بیت های صفر ان نشان دهنده بیت های یکسان ورودی است) حال اگر تعداد بیت های یک شده ی متغیر **w1** را بشماریم در واقع تعداد بیت های متفاوت ورودی ها و فاصله ی همینگ را بدست اورده ایم اما با توجه به سوال خروجی در صورت یک بودن ورودی **start** معتبر است در غیر این صورت صفر است بنابراین با شرط یک شدن **start** حاصل عملیات را در خروجی **out** قرار میدهیم

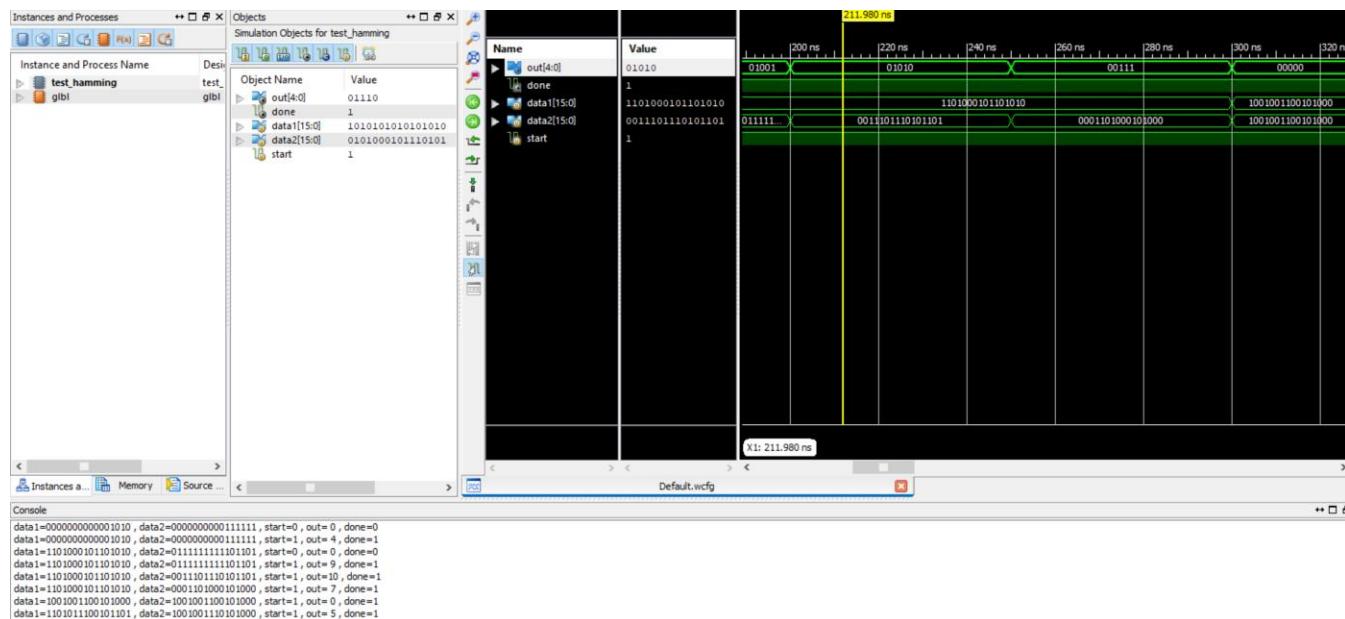
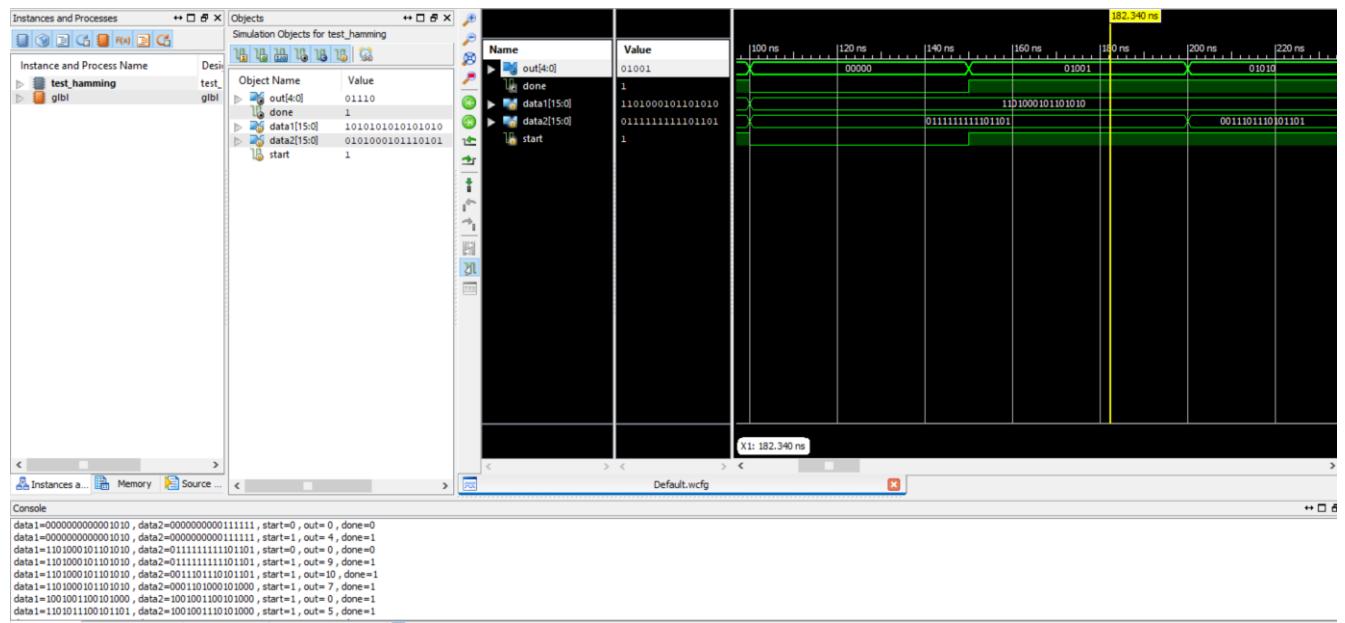
همچنین خروجی **done** در صورت یک بودن **start** و در نیجه انجام عملیات یک است در غیر این صورت با صفر بودن **start** عملیات معتبر انجام نشده و خروجی **done** صفر است بنابراین **start** به **done** متصل میکنیم چون رفتاری همانند آن دارد

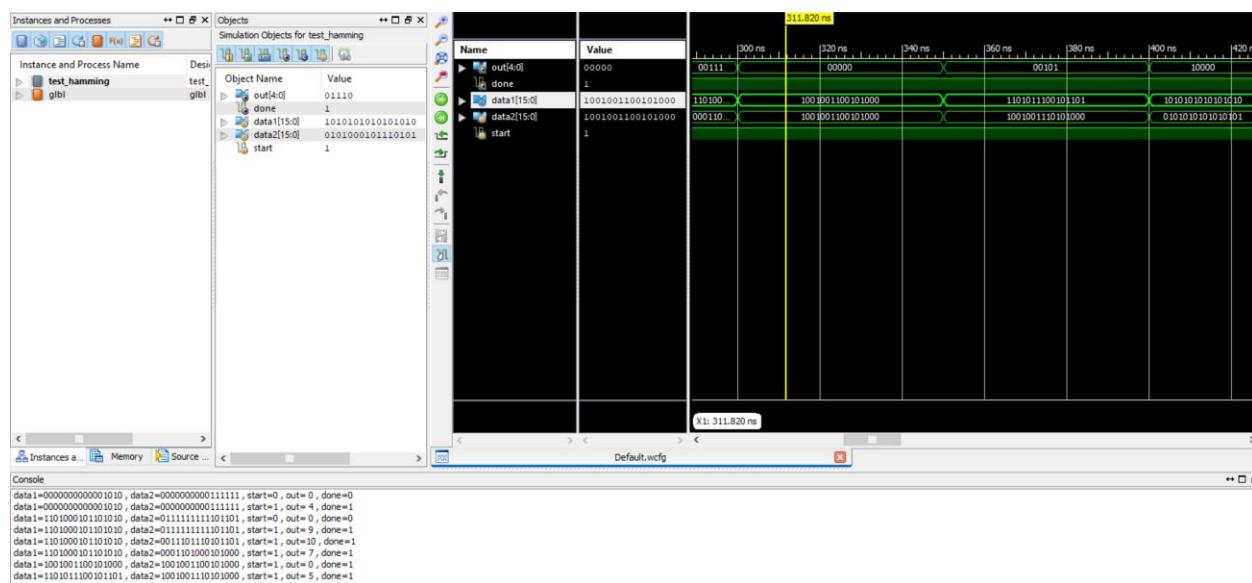
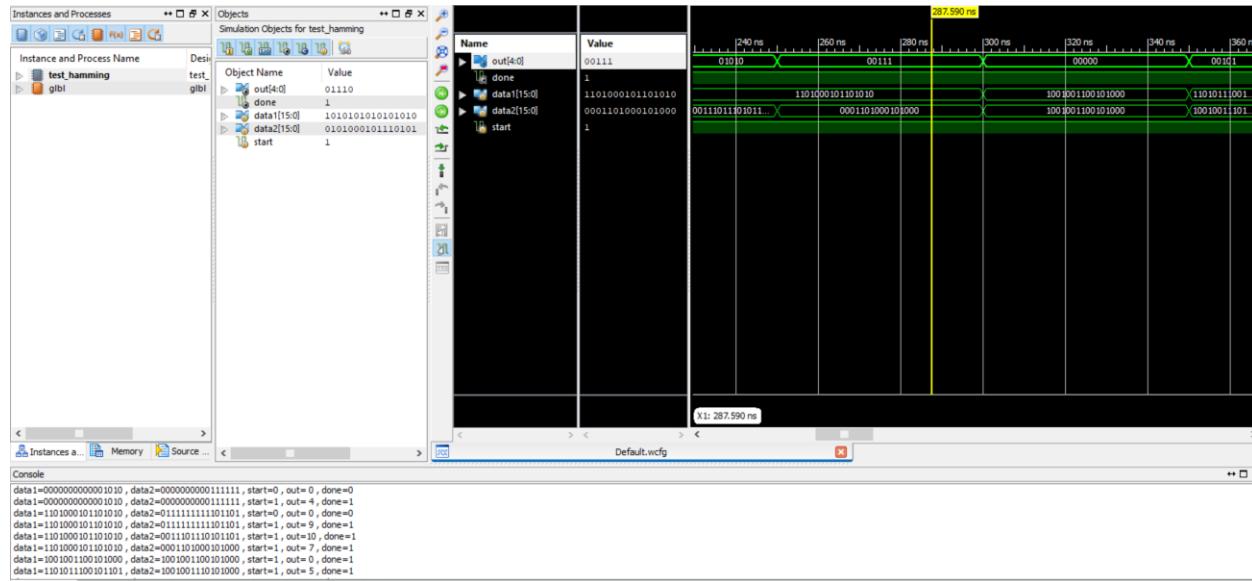
روشی که در این سوال برای کاهش سخت افزار موردنیاز اتخاذ شده است کمبود سخت را مدیریت میکند زیرا روش دیگر یعنی تعریف یک شمارنده و چک کردن دانه به دانه بیت های ورودی ها و در صورت تفاوت شان زیاد شدن شمارنده ! به 16 بار استفاده از یک جمع کننده برای افزایش شمارنده نیاز بود که میدانیم هر جمع کننده فول ادر از دو گیت xor دو گیت and و یک گیت or تشکیل شده است که سخت افزار زیادی را مصرف میکند اما در این روش کل ورودی ها xor شده و سپس با یک بار جمع کردن تمامی بیت ها نیاز ما را به چندین بار استفاده از جمع کننده برطرف کرده است

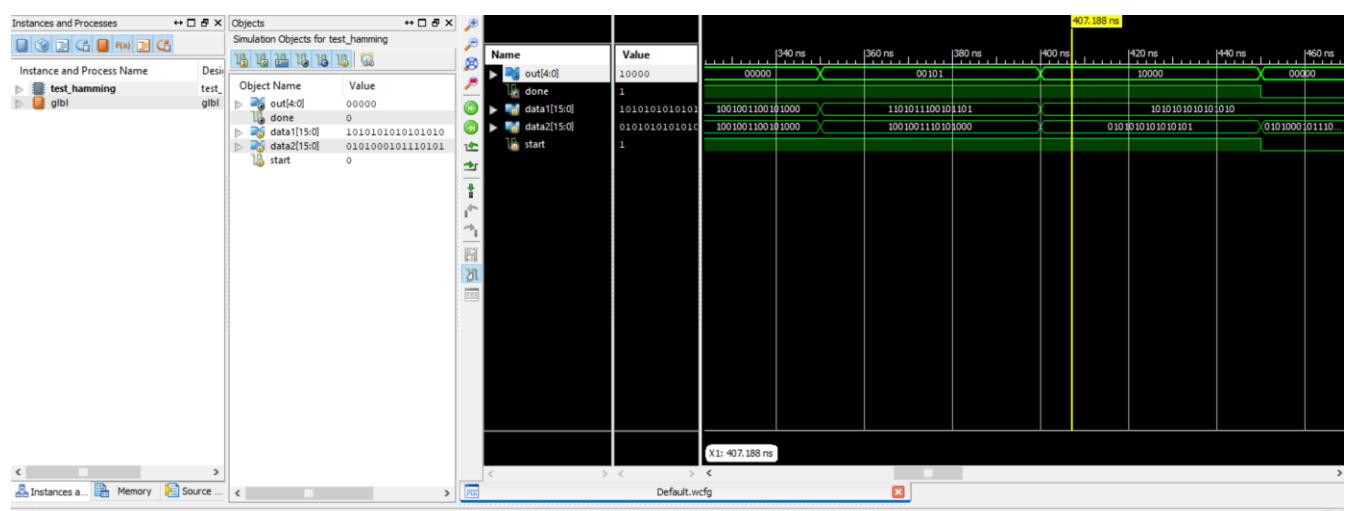
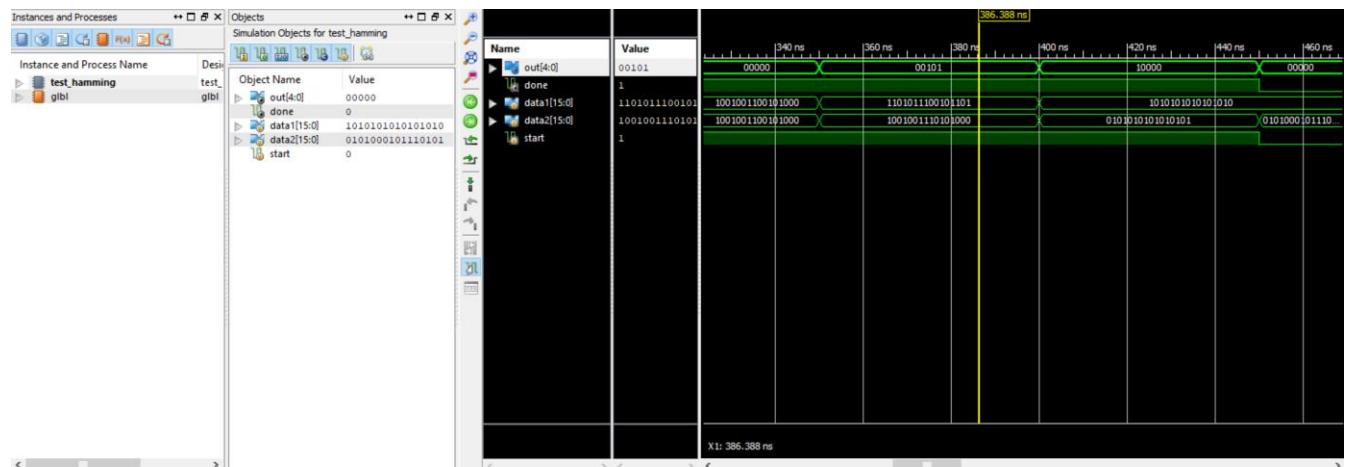
نتایج شبیه سازی و waveform ها









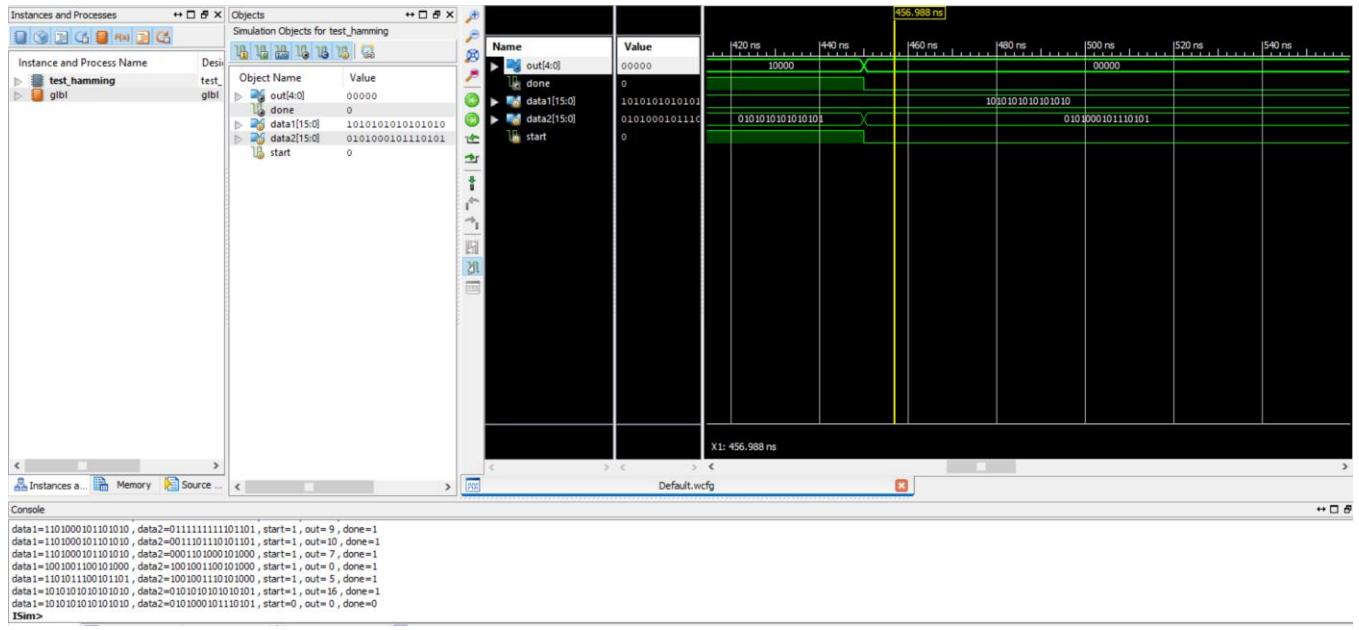


```

data1=1101000101101010 , data2=01111111101101 , start=1 , out=9 , done=1
data1=1101000101101010 , data2=001101101010101 , start=1 , out=10 , done=1
data1=1101000101101010 , data2=0001101000101000 , start=1 , out=7 , done=1
data1=1001001100100000 , data2=1001001100100000 , start=1 , out=1 , done=1
data1=1101011100101010 , data2=1001001100101000 , start=1 , out=5 , done=1
data1=1010101010101010 , data2=1001001100101000 , start=1 , out=16 , done=1
data1=1010101010101010 , data2=0101000101101011 , start=0 , out=0 , done=0

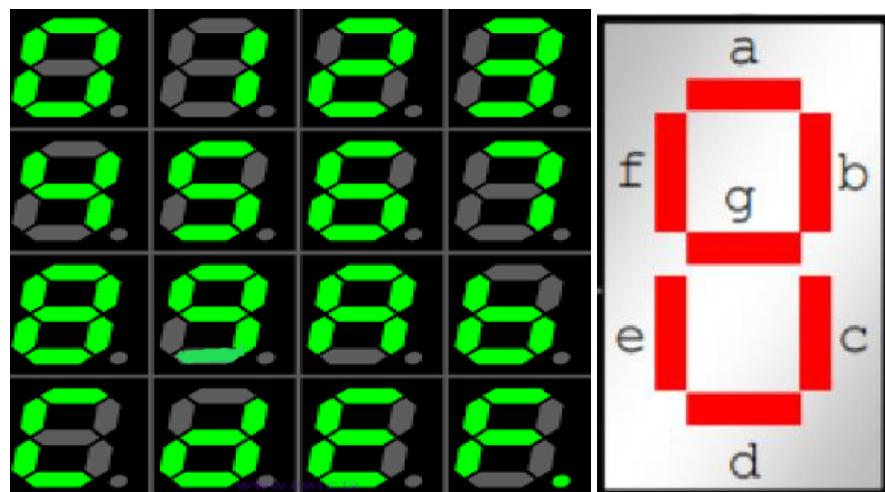
```

I\$im>



سوال 3) این واحد عملیات دارای 4 قسمت است که خروجی با توجه به مقدار متغیر sel تعیین میشود بنابراین بوسیله یک conditional operator چهار شرطی میتوان اثرا پیاده سازی کرد که در حالت اول یعنی $sel=0$ باز با یک conditional operator برای ثبت شدن تفاضل ورودی روبرو هستیم تا بتوانیم تعیین کنیم که همواره ورودی بزرگتر از کوچکتر کم شود

حال برای پیاده سازی دیکودر 7-seg مطابق شکل زیر سعی میکنیم با توجه به 4 بیت ورودی دیکودر تعیین کدام بیت ها و خروجی دیکودر برای نمایش عدد مورد نظر باید یک شوند

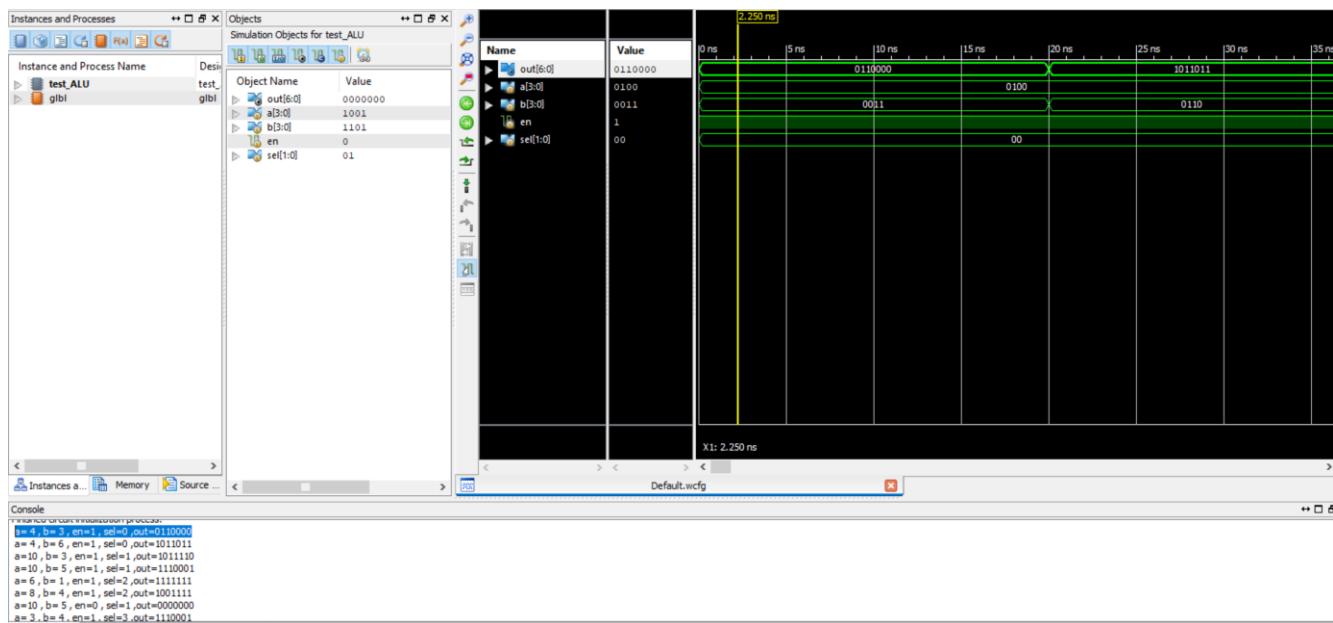


	<u>g</u>	<u>f</u>	<u>e</u>	<u>d</u>	<u>c</u>	<u>b</u>	<u>a</u>
<u>0</u>	<u>0</u>	<u>1</u>	<u>1</u>	<u>1</u>	<u>1</u>	<u>1</u>	<u>1</u>
<u>1</u>	<u>0</u>	<u>1</u>	<u>1</u>	<u>0</u>	<u>0</u>	<u>0</u>	<u>0</u>
<u>2</u>	<u>1</u>	<u>0</u>	<u>1</u>	<u>1</u>	<u>0</u>	<u>1</u>	<u>1</u>
<u>3</u>	<u>1</u>	<u>0</u>	<u>0</u>	<u>1</u>	<u>1</u>	<u>1</u>	<u>1</u>
<u>4</u>	<u>1</u>	<u>1</u>	<u>0</u>	<u>0</u>	<u>1</u>	<u>1</u>	<u>0</u>
<u>5</u>	<u>1</u>	<u>1</u>	<u>0</u>	<u>1</u>	<u>1</u>	<u>0</u>	<u>1</u>
<u>6</u>	<u>1</u>	<u>1</u>	<u>1</u>	<u>1</u>	<u>1</u>	<u>0</u>	<u>1</u>
<u>7</u>	<u>0</u>	<u>0</u>	<u>0</u>	<u>0</u>	<u>1</u>	<u>1</u>	<u>1</u>
<u>8</u>	<u>1</u>						
<u>9</u>	<u>1</u>	<u>1</u>	<u>0</u>	<u>1</u>	<u>1</u>	<u>1</u>	<u>1</u>
<u>10=A</u>	<u>1</u>	<u>1</u>	<u>1</u>	<u>0</u>	<u>1</u>	<u>1</u>	<u>1</u>
<u>11=b</u>	<u>1</u>	<u>1</u>	<u>1</u>	<u>1</u>	<u>1</u>	<u>0</u>	<u>0</u>
<u>12=c</u>	<u>0</u>	<u>1</u>	<u>1</u>	<u>1</u>	<u>0</u>	<u>0</u>	<u>1</u>
<u>13=d</u>	<u>1</u>	<u>0</u>	<u>1</u>	<u>1</u>	<u>1</u>	<u>1</u>	<u>0</u>
<u>14=E</u>	<u>1</u>	<u>1</u>	<u>1</u>	<u>1</u>	<u>0</u>	<u>0</u>	<u>1</u>
<u>15=F</u>	<u>1</u>	<u>1</u>	<u>1</u>	<u>0</u>	<u>0</u>	<u>0</u>	<u>1</u>

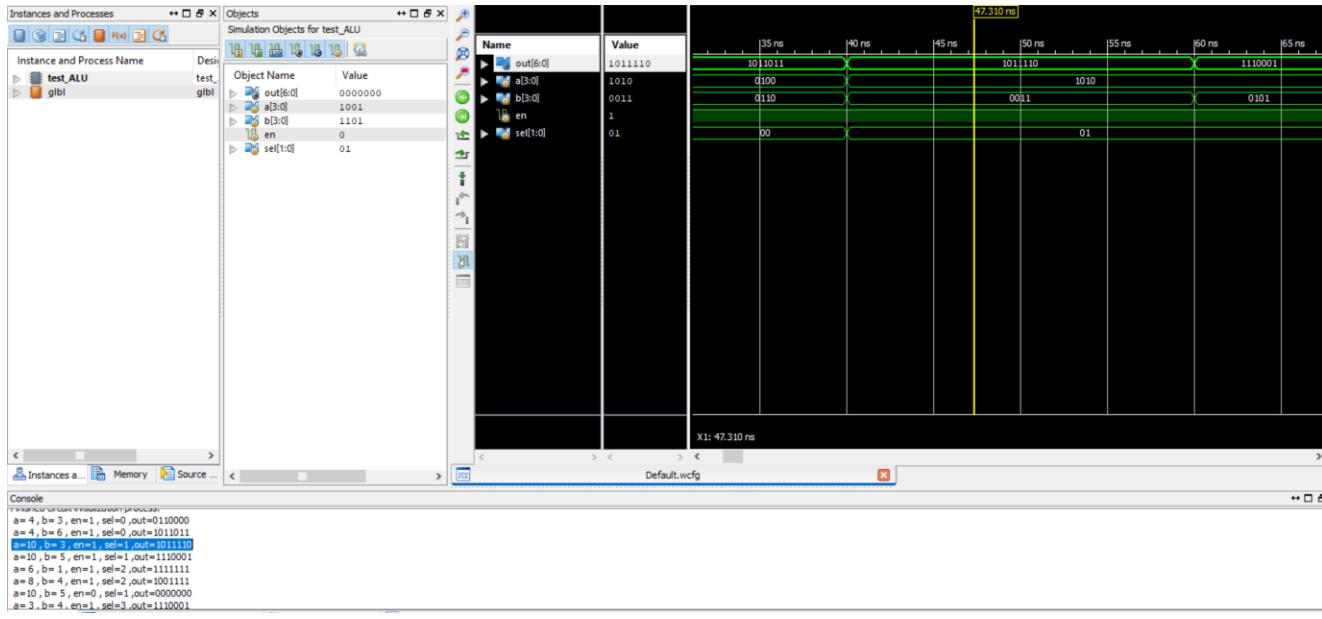
حال با توجه به جدول بالا مژول sevenseg_decoder را چنان طراحی میکنیم که به ازای هر ورودی از جدول بالا بیت های متناظر خروجی را یک کند تا عدد مبنا شانزده آن روی سون سگمنت نمایش داده شود اما باید دقت داشت تنها زمانی که پایه فعال ساز یعنی EN فعال و یک باشد خروجی خواهیم داشت در غیر این صورت خروجی مژول دیکودر صفر و هیچ سون سگمنت روش نخواهد بود که به وسیله‌ی یک conditional operator LED صورت برقرار بودن شرط یک بودن پایه EN خروجی را غیر صفر نمایش خواهد داد

نتایج شبیه سازی و waveform ها

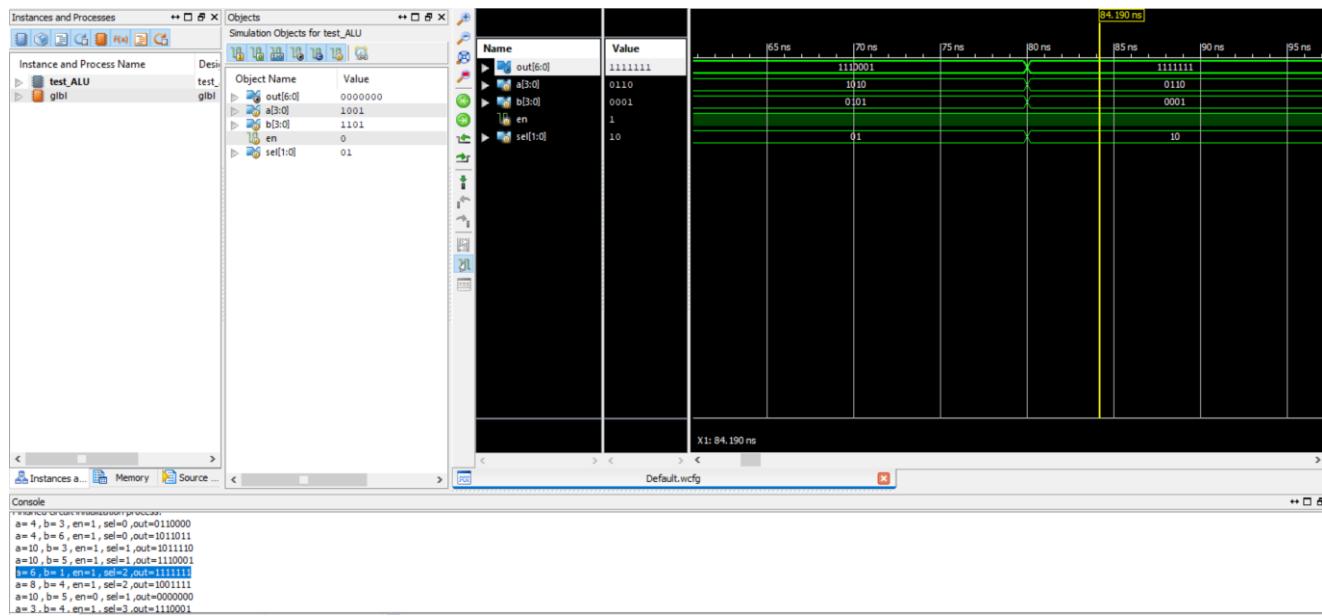
$a = 4, b = 3, en = 1, sel = 0, out = 0110000$



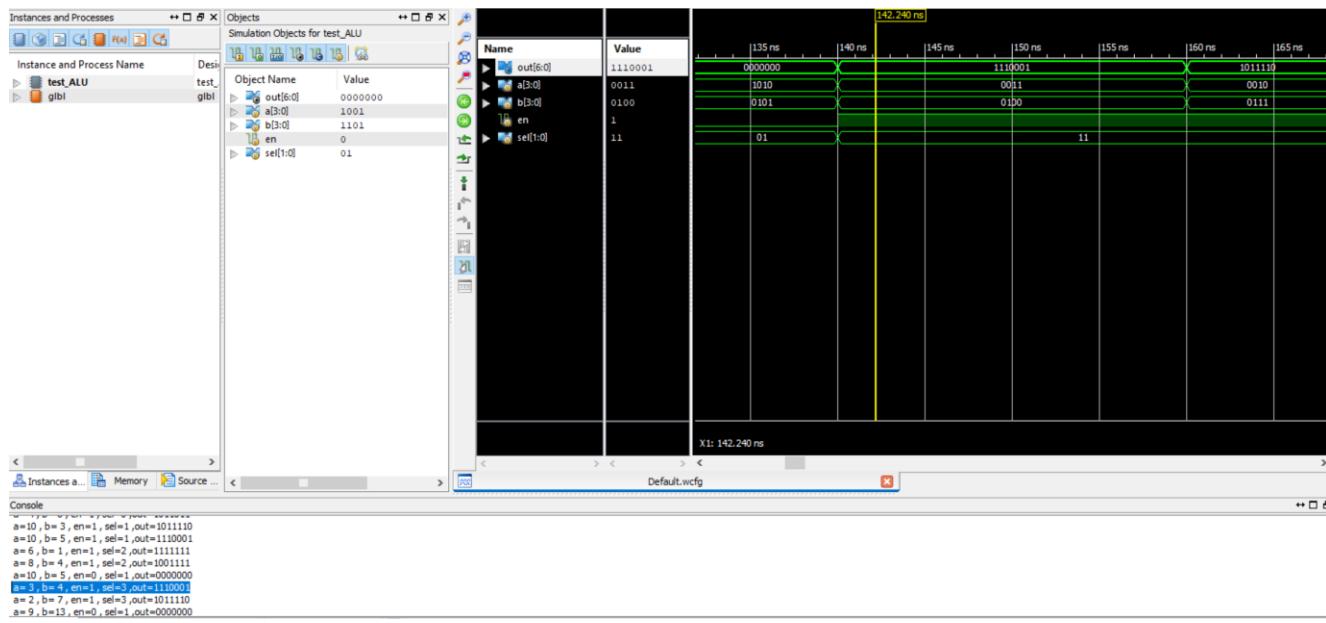
$a = 10, b = 3, en = 1, sel = 1, out = 1011110$



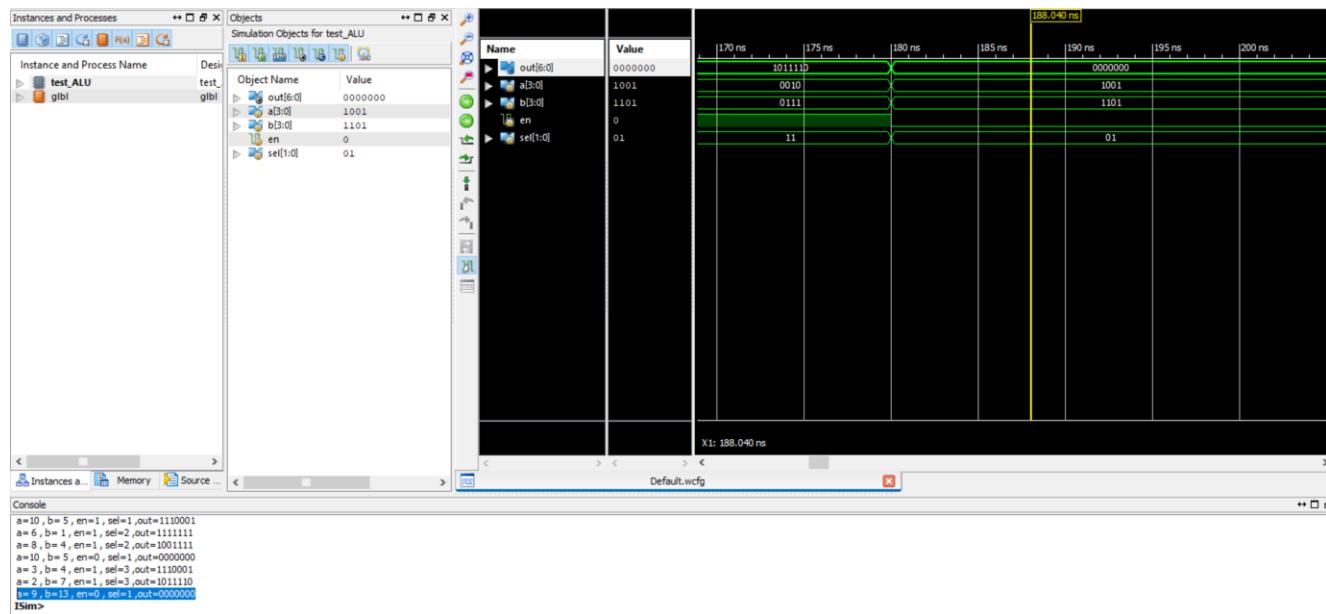
$a = 6, b = 1, en = 1, sel = 2, out = 1111111$



$a = 3, b = 4, en = 1, sel = 3, out = 1110001$



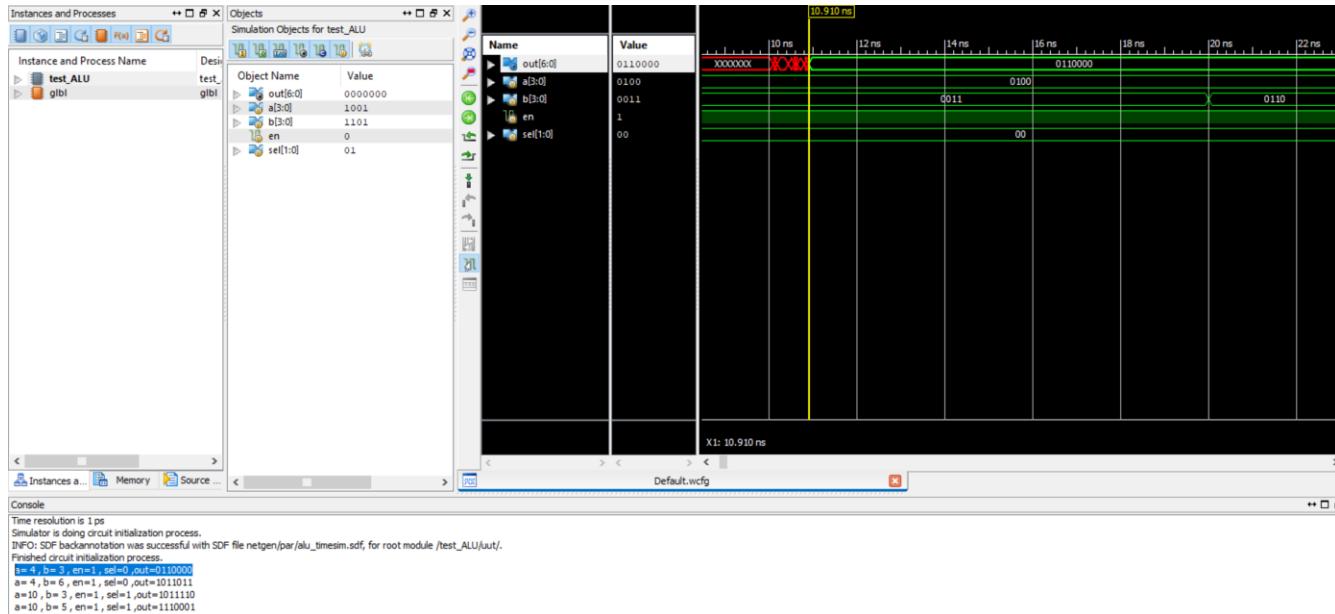
$a = 9, b = 13, en = 0, sel = 1, out = 0000000$



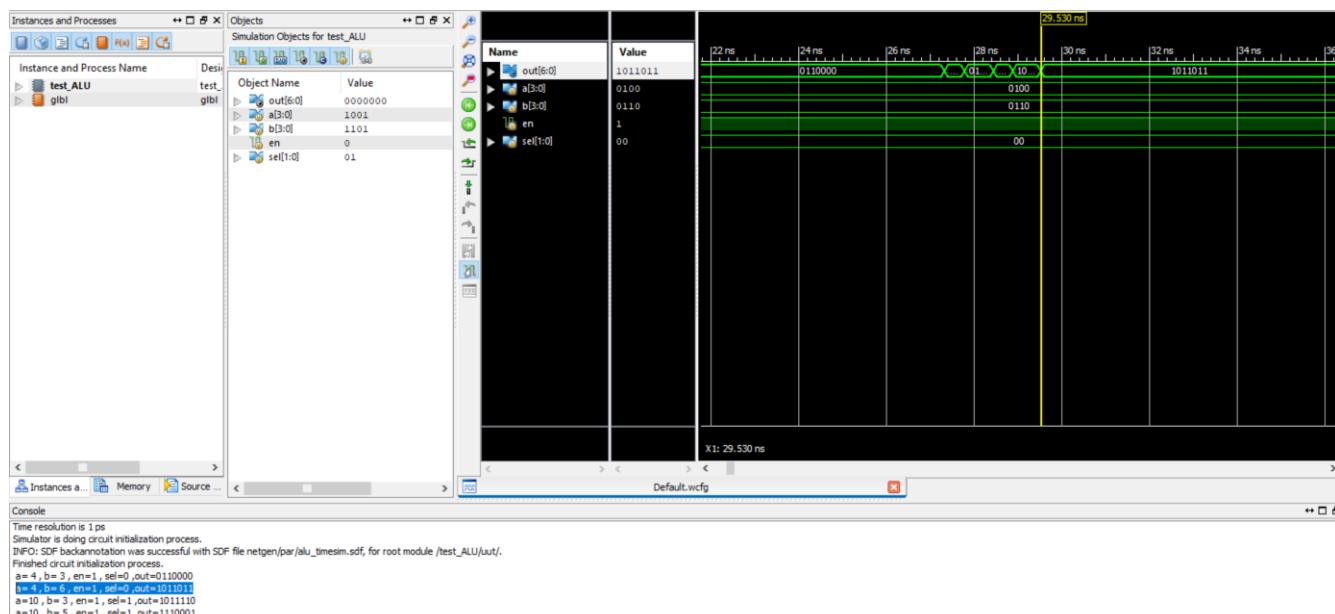
با توجه به جدول بالا در هر تصویر نتیجه شبیه سازی تطابق پاسخ ALU را با اعداد مربوط به نمایش ان میتوان مشاهده کرد در واقع خروجی OUT عددی که باید به سون سگمنت اعمال بشود تا نتیجه ی محاسبات نمایش داده شود را نشان میدهد نه پاسخ واقعی عملیات ALU

قسمت ب) در حالتی که فاصلهٔ اعمال ورودی‌های بعدی 20 نانو ثانیه است

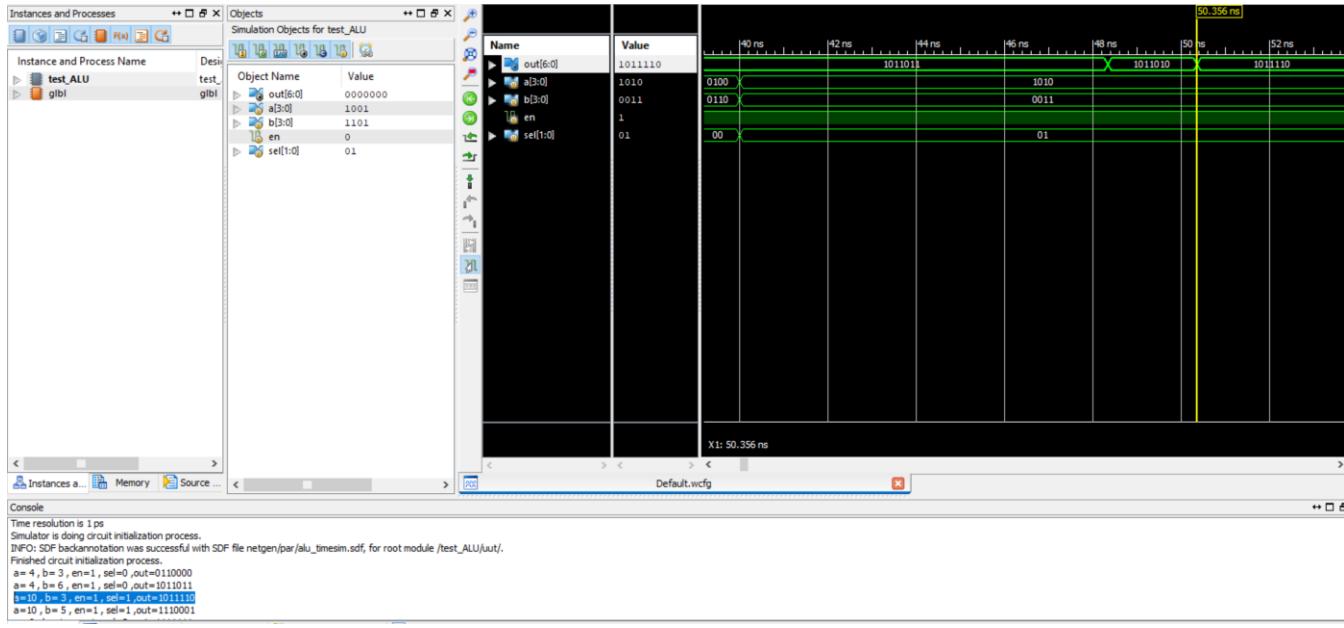
$a = 4, b = 3, en = 1, sel = 0, out = 0110000, delay = 10.91\text{ns}$



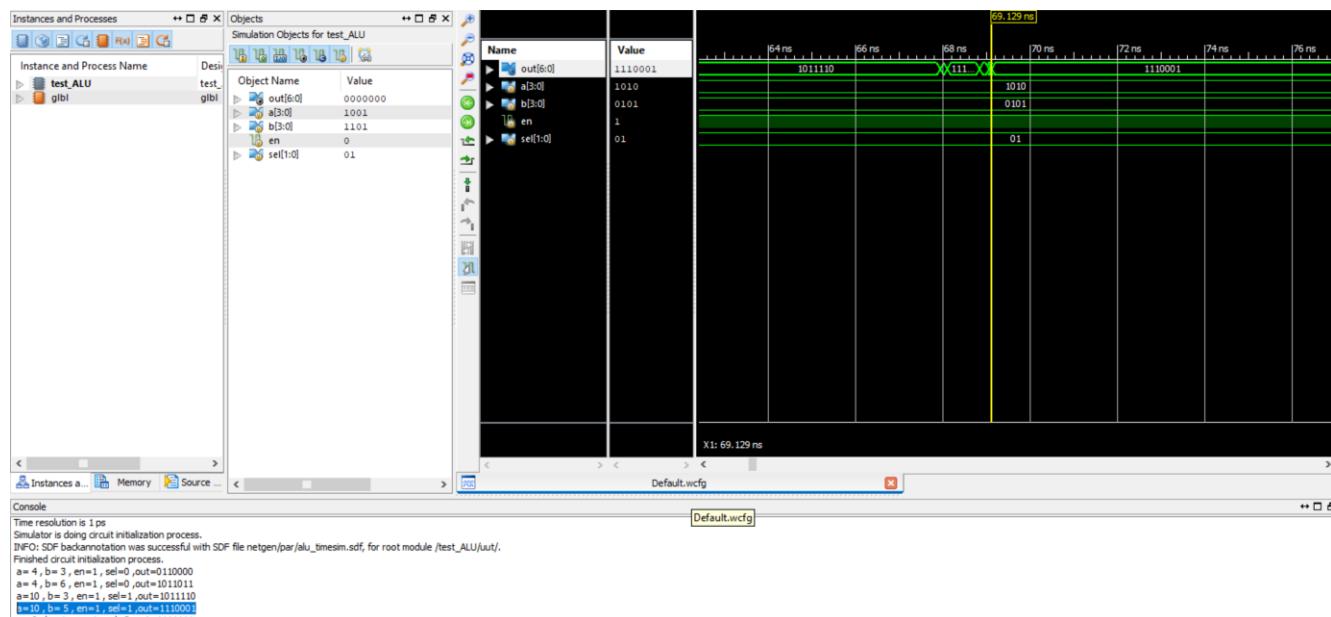
$a = 4, b = 6, en = 1, sel = 0, out = 1011011, delay = 9.53\text{ns}$



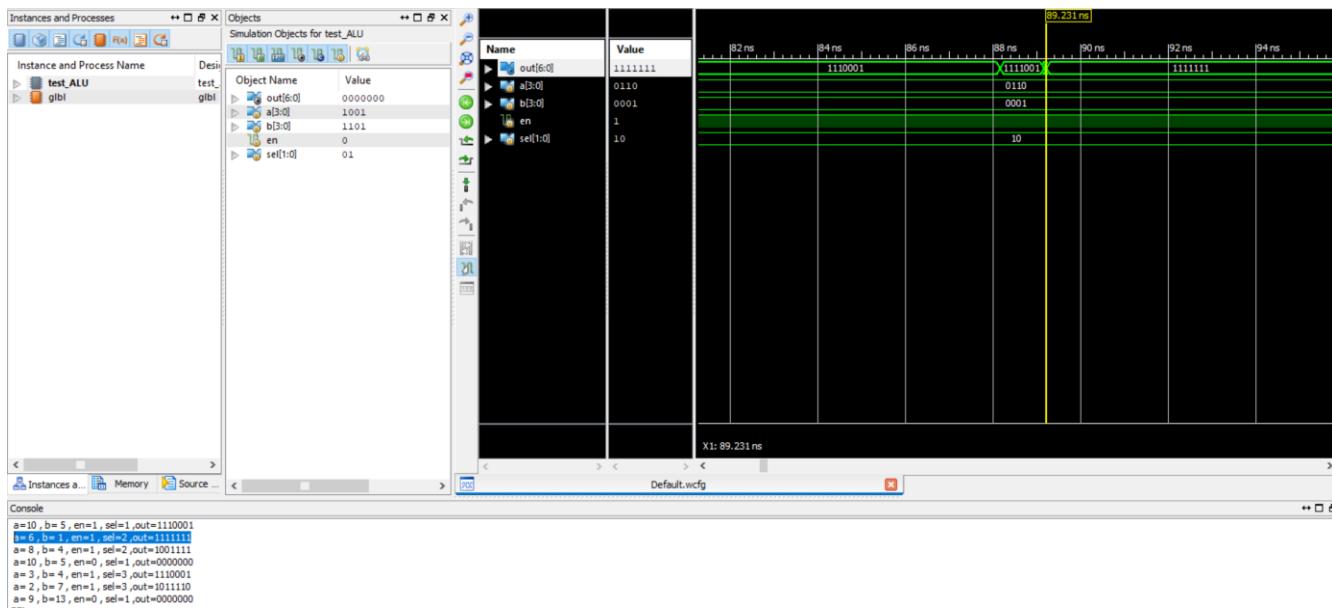
$a = 10, b = 3, en = 1, sel = 1, out = 1011110, delay = 10.356\text{ns}$



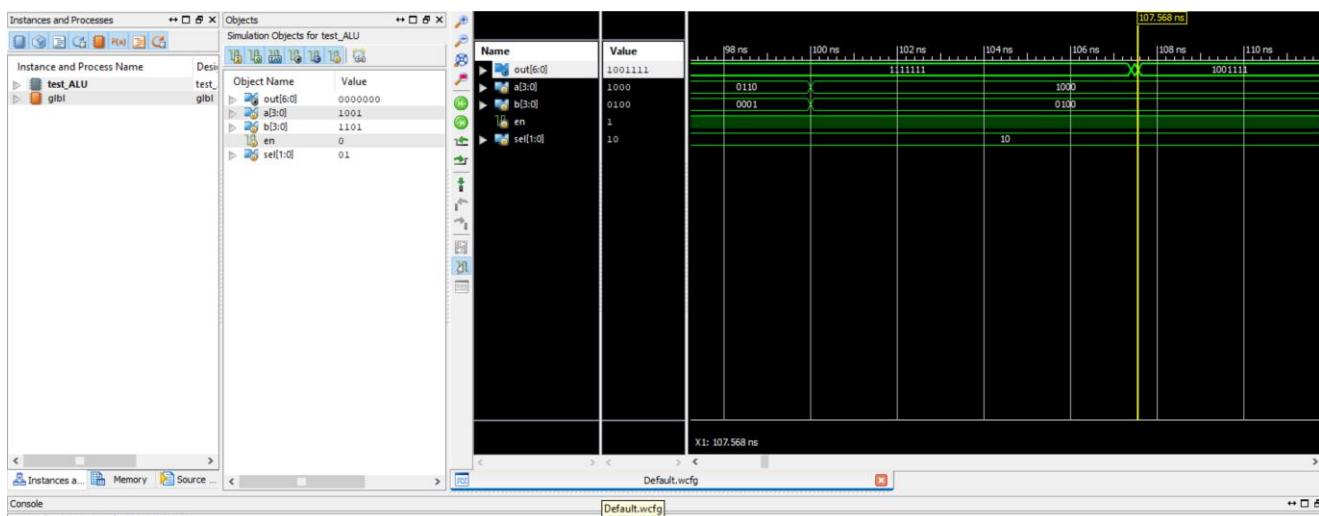
$a=10, b=5, en=1, sel=1, out=1110001, delay=9.129\text{ns}$



$a=6, b=1, en=1, sel=2, out=1111111, delay=9.231\text{ns}$



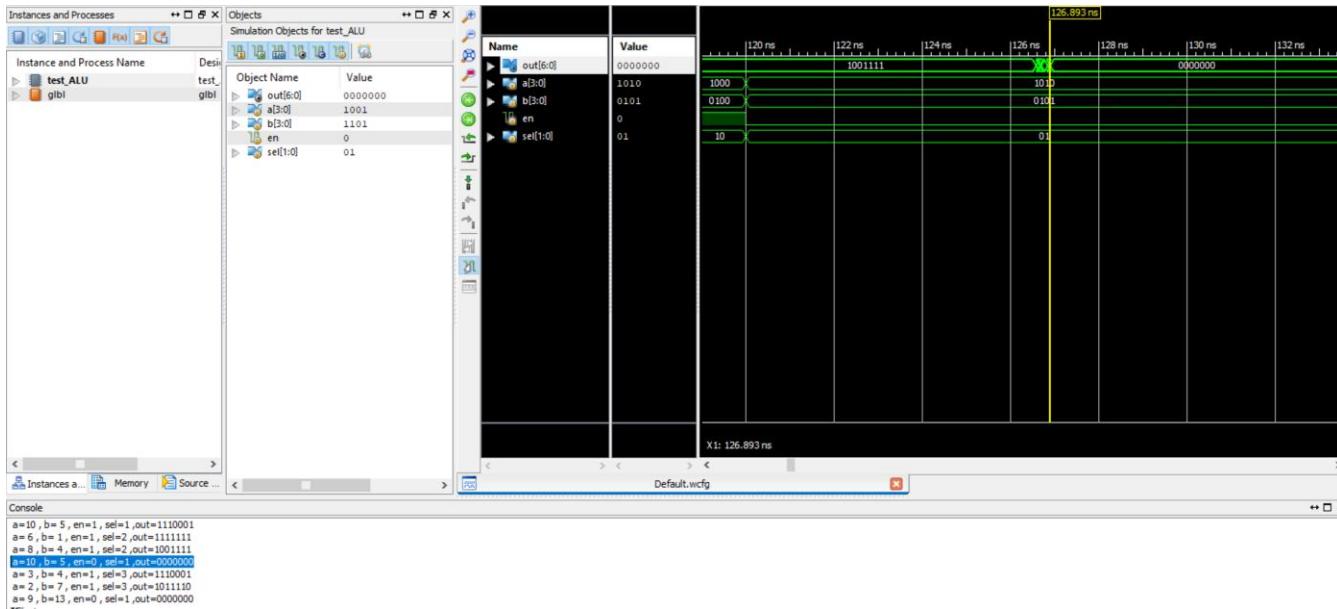
a= 8 , b= 4 , en=1 , sel=2 ,out=1001111, delay=7.568



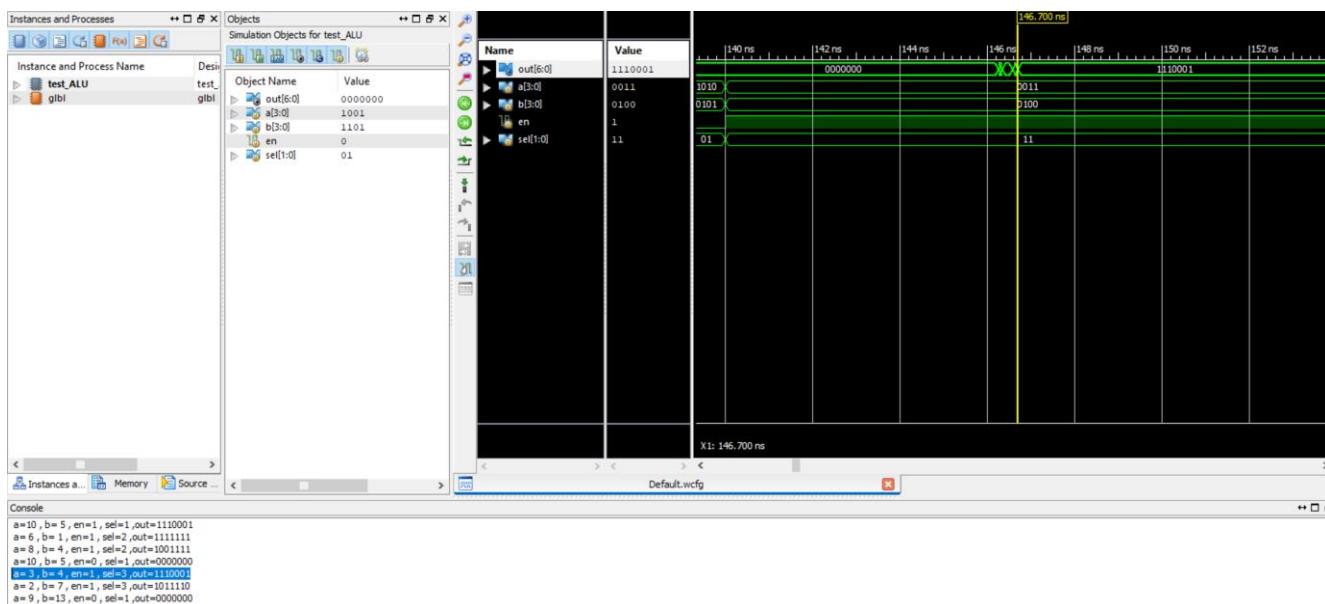
a= 8 , b= 4 , en=1 , sel=2 ,out=1001111, delay=7.568



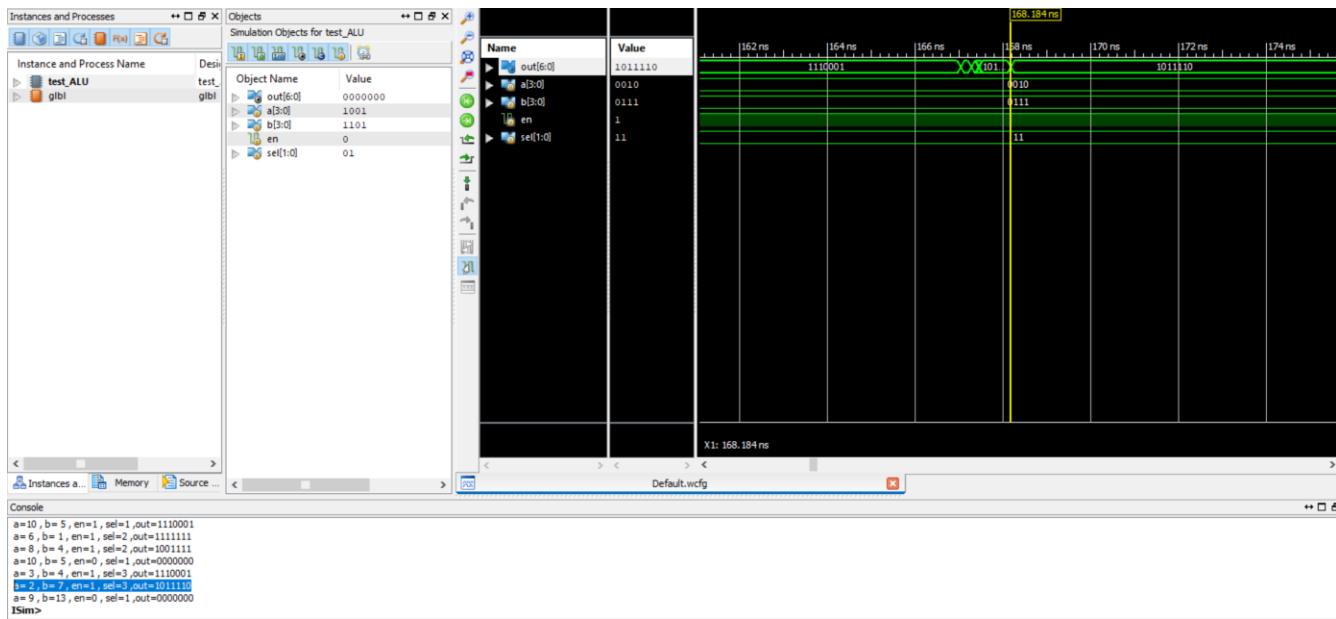
a=10 , b= 5 , en=0 , sel=1 ,out=0000000, delay=6.893



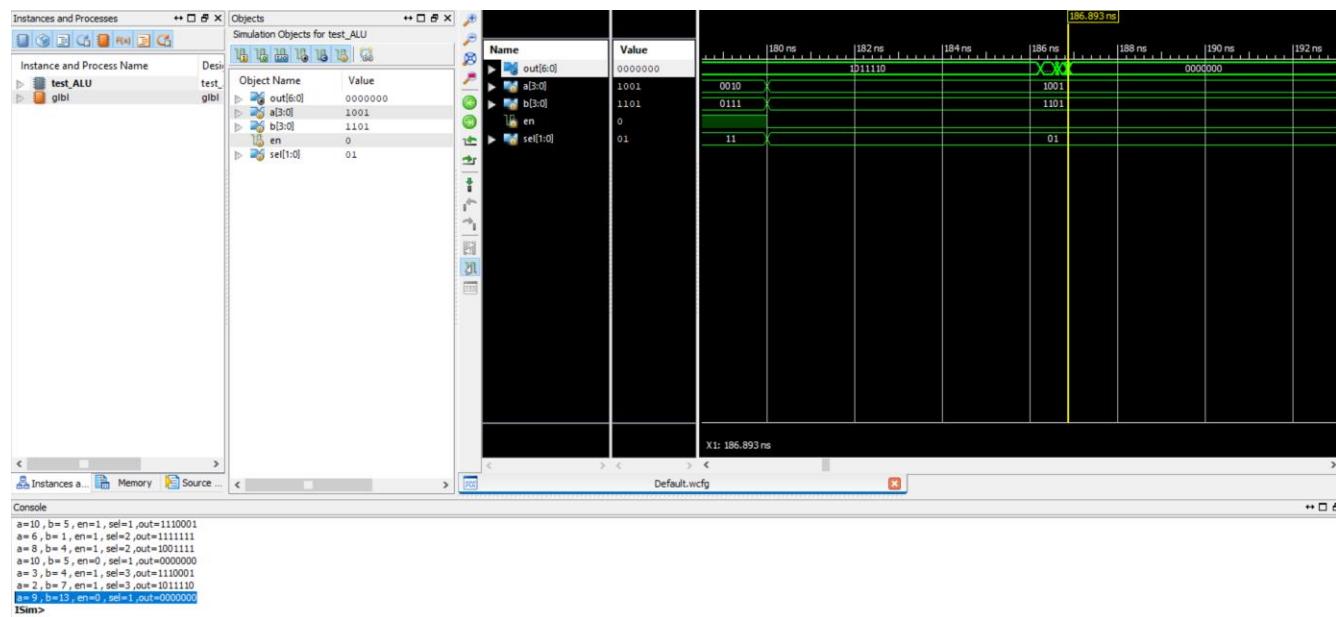
$a = 3, b = 4, en = 1, sel = 3, out = 1110001, \text{delay} = 6.7\text{ns}$



$a = 2, b = 7, en = 1, sel = 3, out = 1011110, \text{delay} = 8.184\text{ns}$



a= 9 , b=13 , en=0 , sel=1 ,out=0000000 , delay=6.893ns

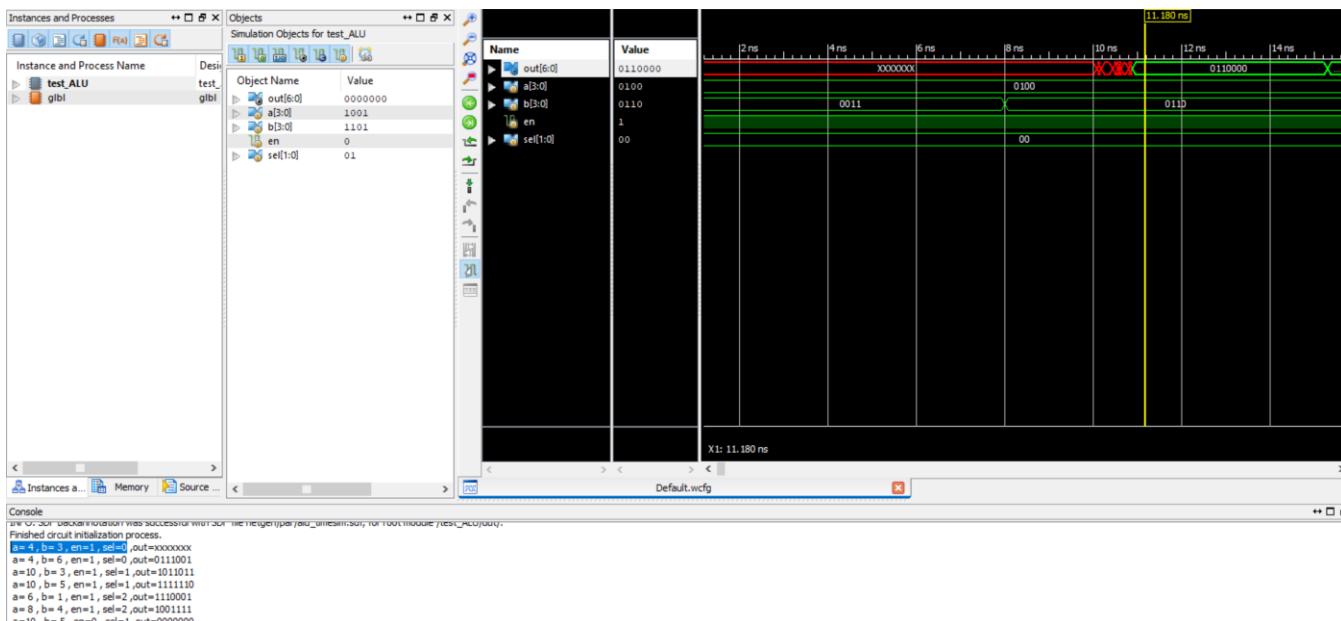


با توجه به تصاویر فوق مشاهده میکنیم که با فاصله زمانی 20 نانو ثانیه قبل از اعمال ورودی بعدی خروجی درست بدست می آید و در این بازه هم ورودی و هم خروجی مربوط به همان ورودی را بطور درست میتوانیم ببینیم و حداکثر تاخیر برابر 10.91 نانو ثانیه است

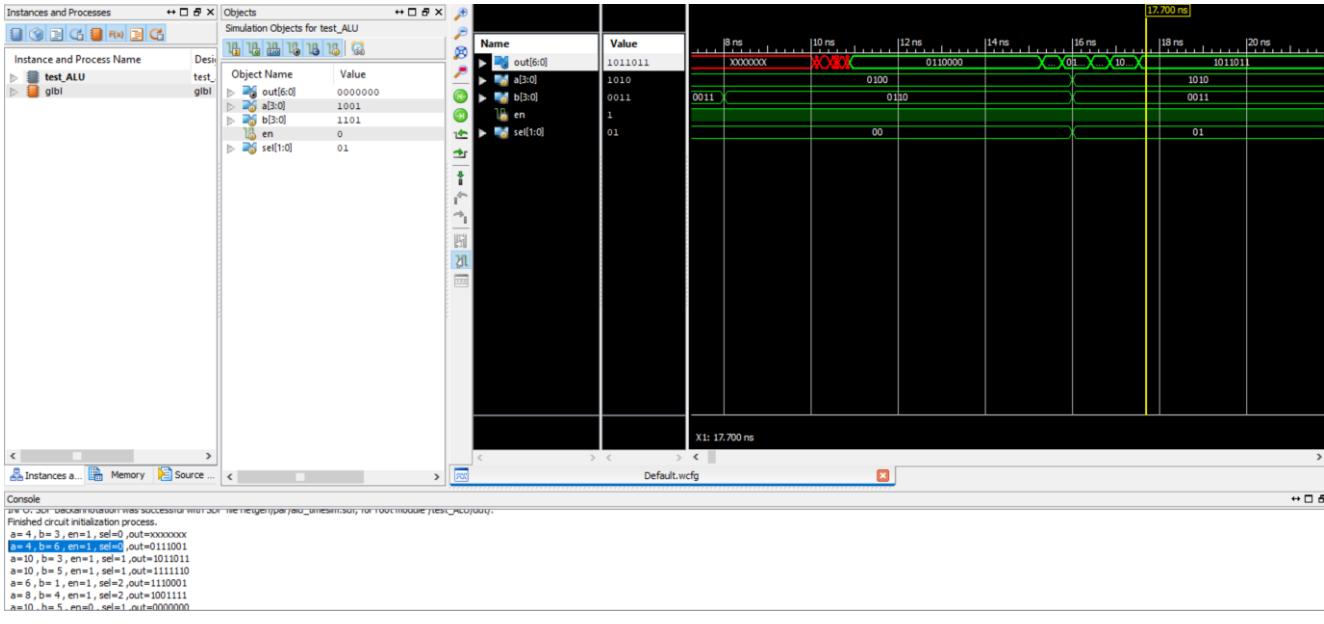
حال فاصله‌ی اعمال ورودی‌ها را کم می‌کنیم

با توجه به اینکه تاخیر‌ها حدوداً در بازه‌ی 5.6 تا 10 نانو‌ثانیه می‌باشند در صورتی که فاصله‌ی اعمال ورودی جدید را 8 نانو‌ثانیه قرار دهیم انتظار داریم محاسباتی که تاخیر کمی در حدود 6 نانو‌ثانیه دارند قبل از اعمال ورودی جدید پاسخ صحیح شان را ببینیم و محاسبات با تاخیر بیشتر در حدود ده نانو‌ثانیه بعد از اعمال ورودی بعدی پاسخ شان اماده شود

در این حالت چون تاخیر محاسبه‌ی این ورودی 10.91 نانو‌ثانیه است بعد از اعمال ورودی بعدی خروجی را مشاهده می‌کنیم یعنی با اعمال ورودی دوم ما خروجی اول را ببینیم

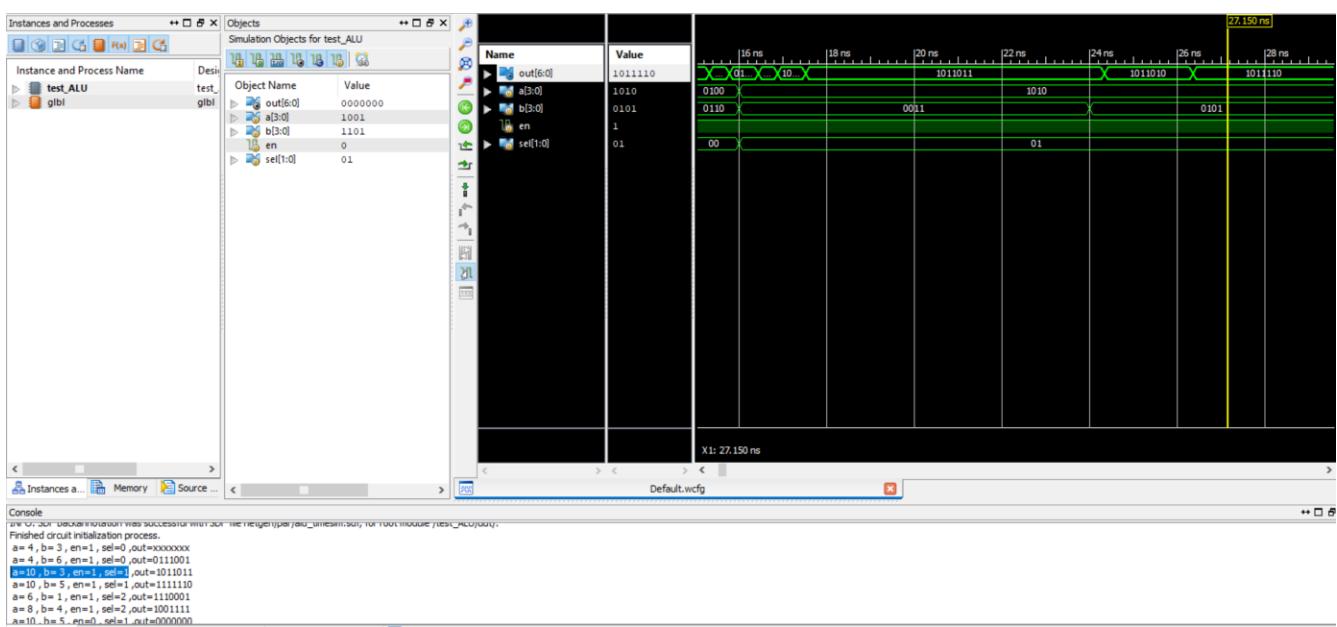


در این حالت نیز خروجی بعد از اعمال ورودی سوم بدست می‌اید



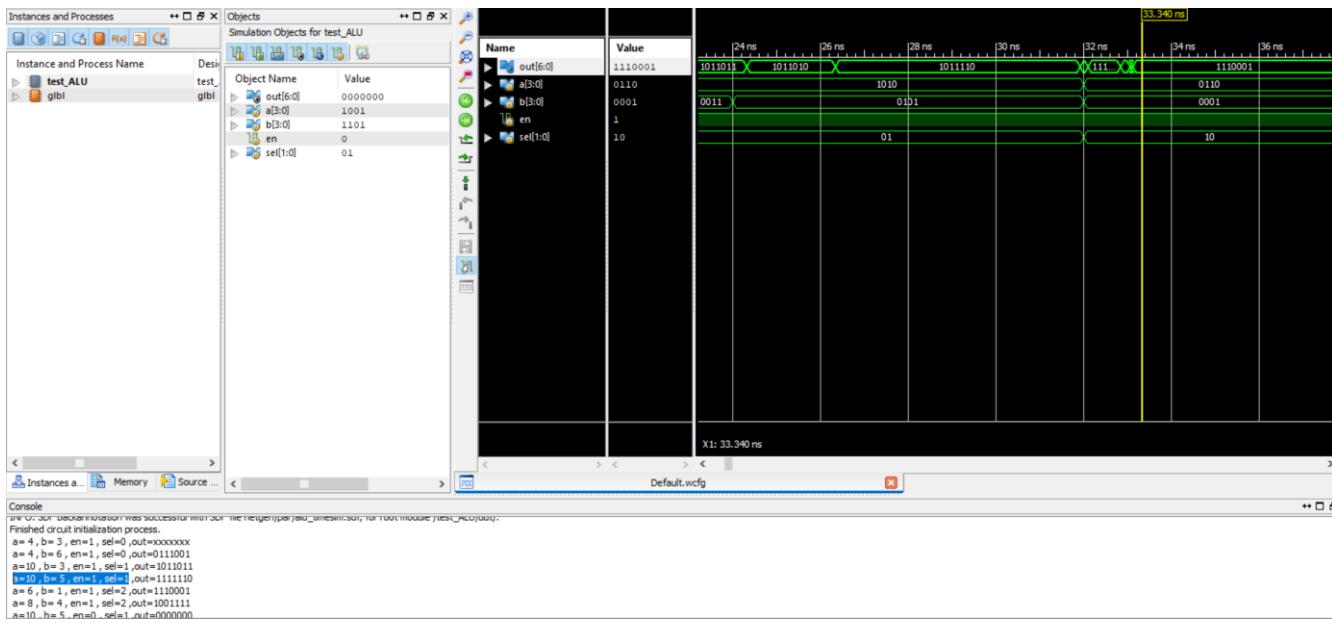
$a=10$, $b=3$, $en=1$, $sel=1$ در این حالت نیز خروجی بعد از اعمال ورودی چهارم بدهست می‌اید یعنی بعد 24

ناموشه

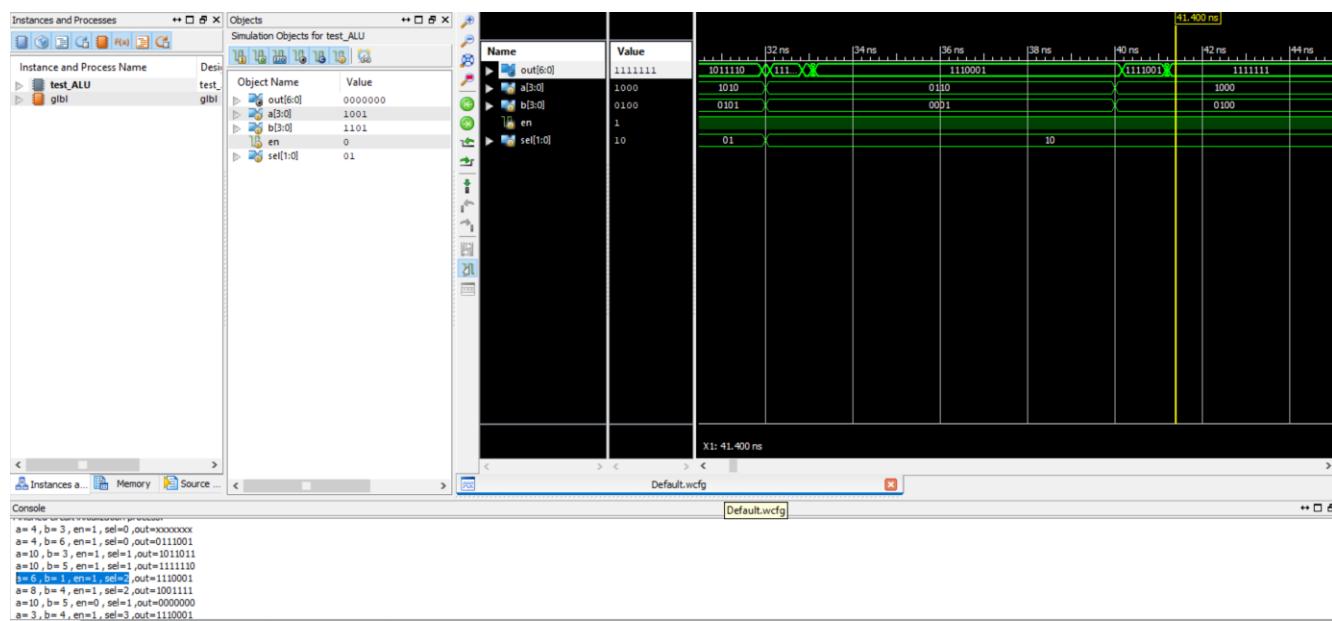


a=10 , b= 5 , en=1 , sel=1 خروجی 15 که همان 1110001 است بعد از اعمال خروجی پنجم یعنی بعد از

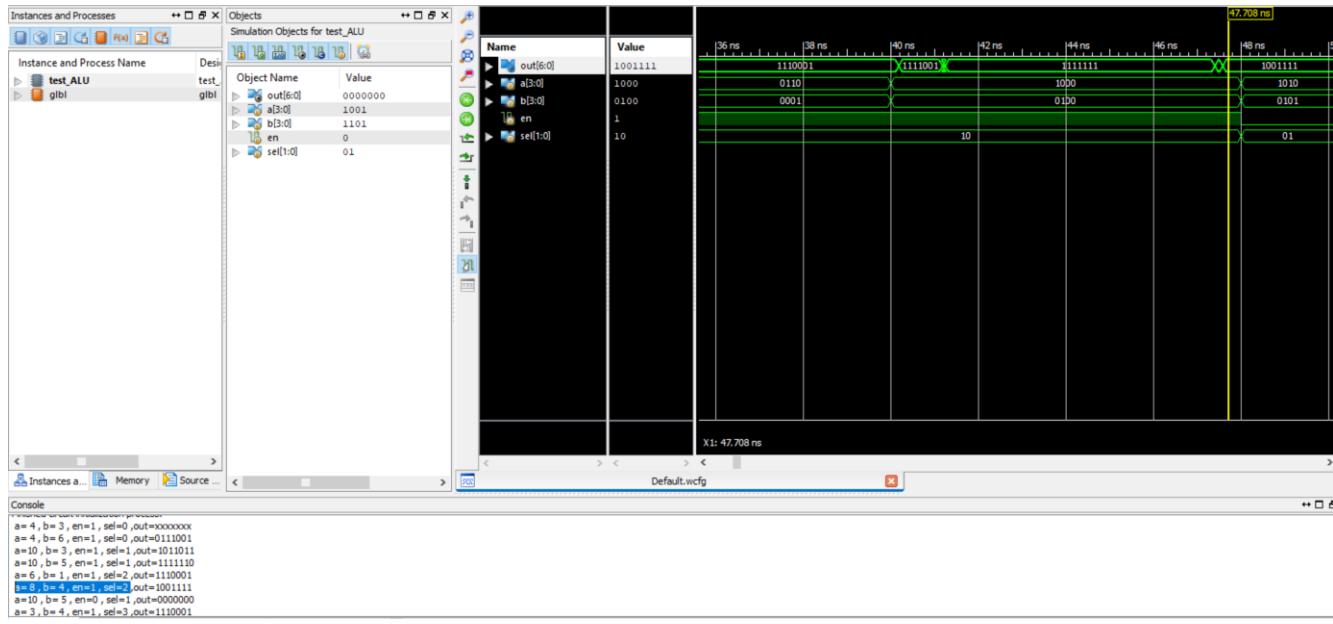
32 نانو ثانیه اماده است



خروجی این حالت یعنی 1111111 که همان 8 است در ورودی بعدی اماده است

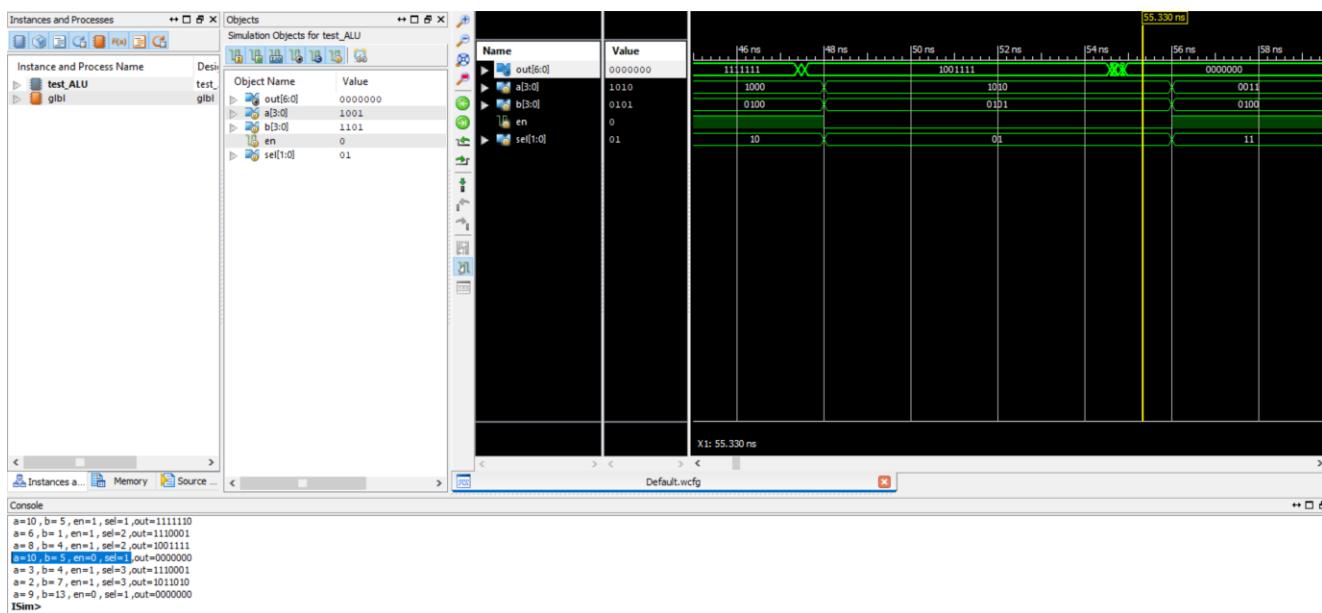


a=8, b=4, en=1, sel=2 در این حالت اما بعلت تاخیر کم انجام محاسبات که حدود 7.57 نانو ثانیه است خروجی یا در این حالت قبل از اعمال ورودی بعدی قابل مشاهده است یعنی در این بازه هم خروجی همین ورودی و هم خروجی ناشی از ورودی مرحله قبل قابل مشاهده است



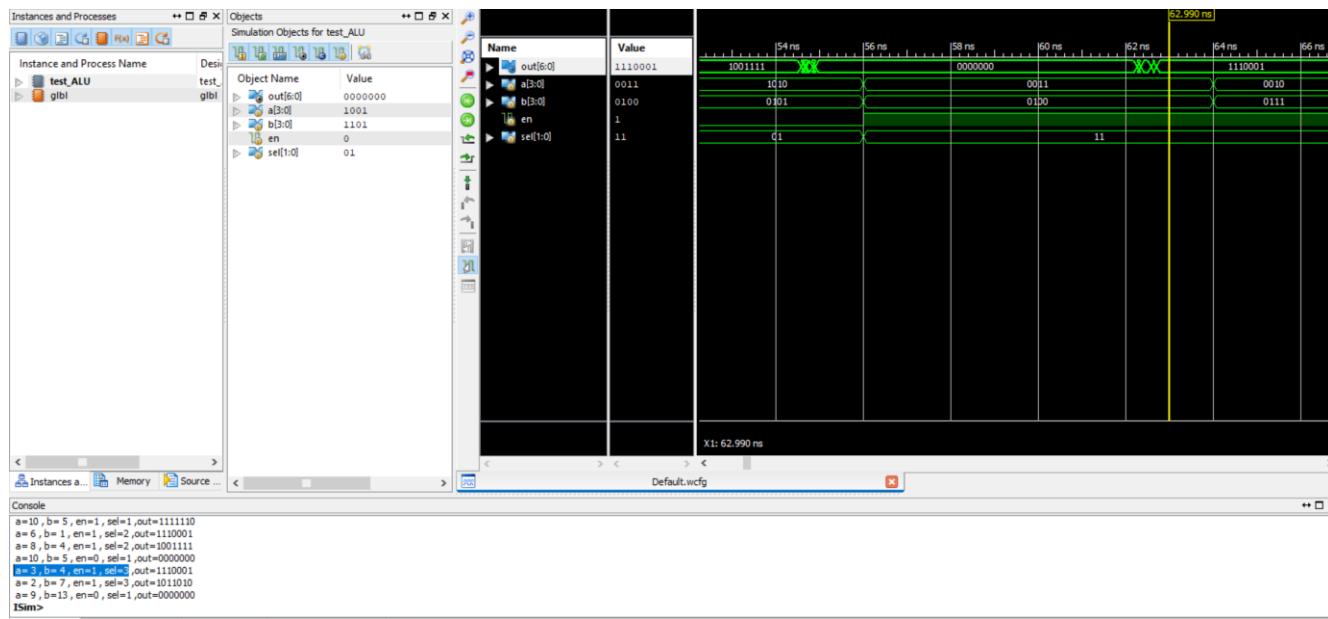
a=4 , b= 3 , en=1 , sel=0 , out=xxxxxxx
a=4 , b= 6 , en=1 , sel=0 , out=0111001
a=10 , b= 3 , en=1 , sel=1 , out=1011011
a=10 , b= 5 , en=1 , sel=1 , out=1110001
a=3 , b= 4 , en=1 , sel=2 , out=1100001
a=10 , b= 4 , en=1 , sel=1 , out=1001111
a=10 , b= 5 , en=0 , sel=1 , out=0000000
a=3 , b= 4 , en=1 , sel=3 , out=1110001

هم همچنان وجود دارد



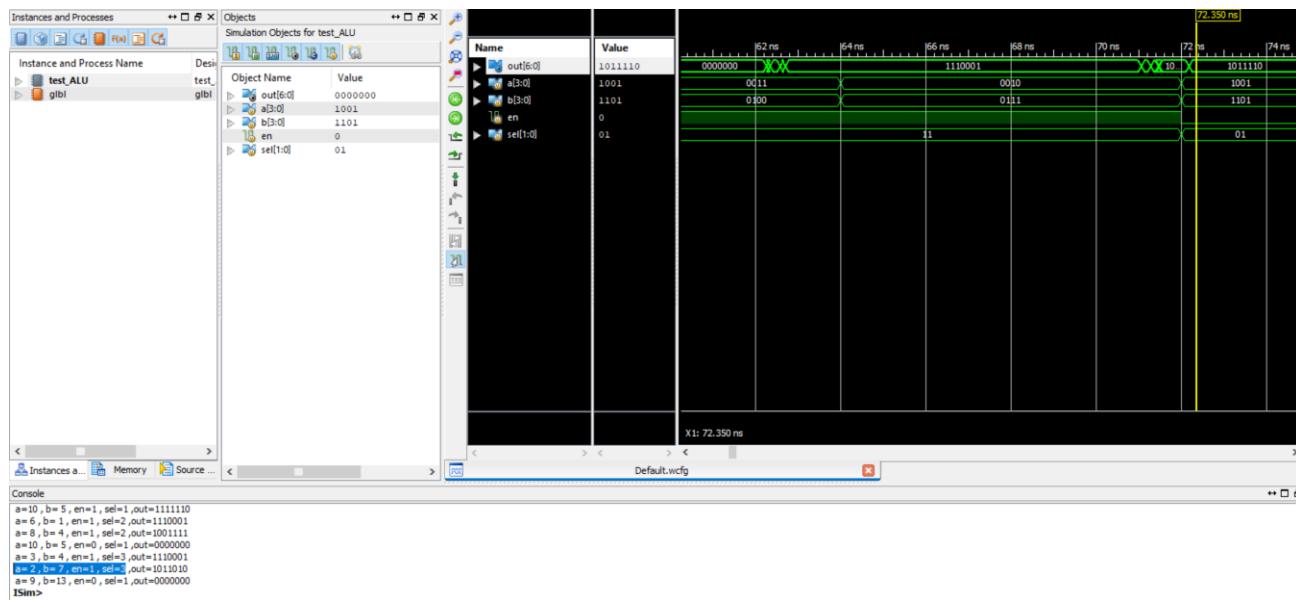
مشابه مرحله قبل با تاخیر کم محاسبه این مرحله که حدود 6.7 نانو ثانیه است

قبل ثانیه 64 ام و اعمال ورودی بعدی خروجی این مرحله را میبینیم



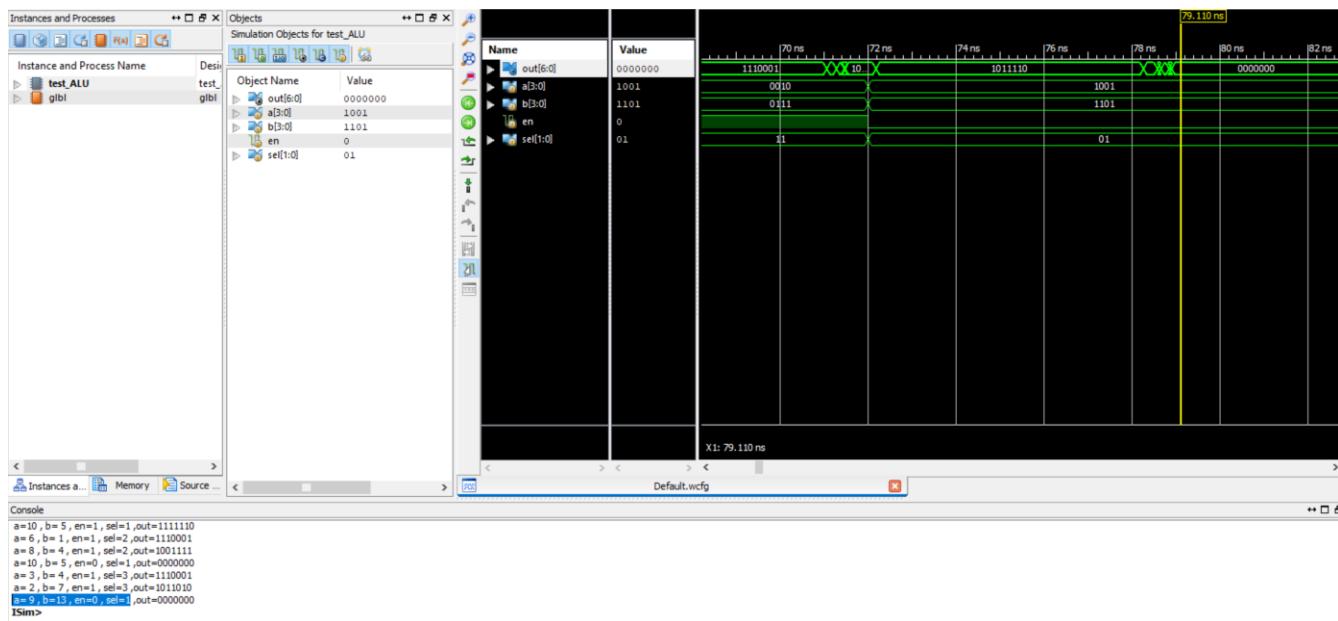
در این مرحله اما بعلت تاخیر بیشتر از 8 نانو ثانیه (8.184) خروجی این مرحله با

اعمال ورودی بعدی قابل مشاهده است



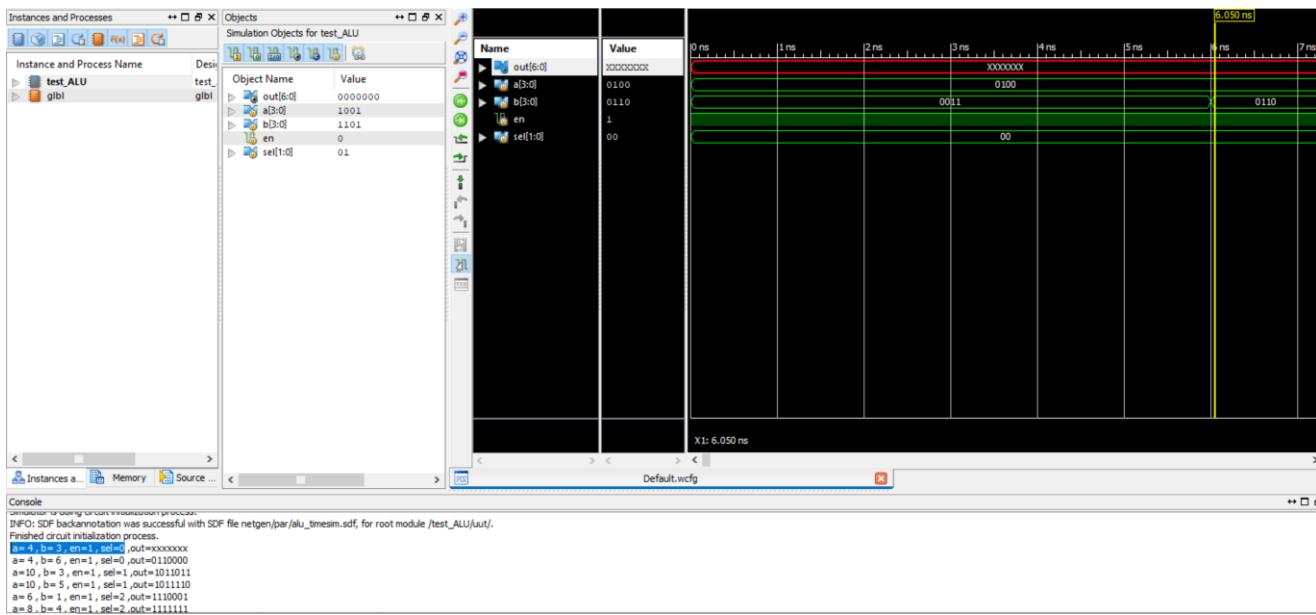
بعدت تاخیر کم در حدود 6.893 نانو ثانیه قبل ثانیه 80 ام نتیجه‌ی این ورودی

قابل مشاهده است

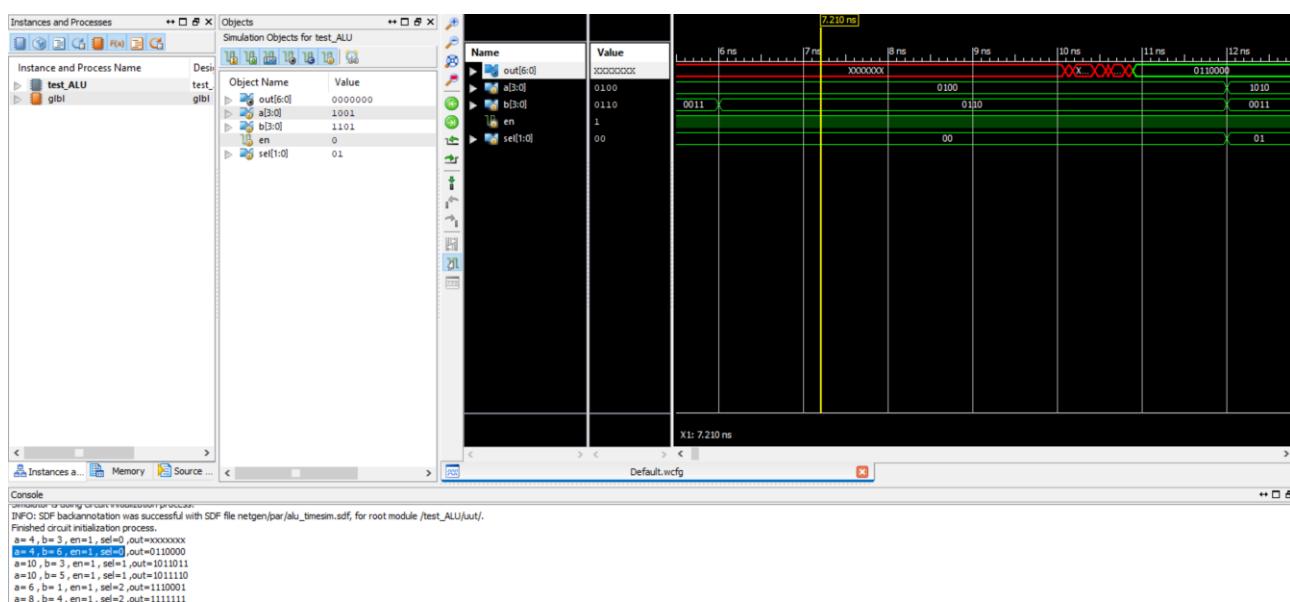


با توجه به اینکه تاخیر‌ها حدوداً در بازه‌ی 6.5 تا 10 نانو ثانیه می‌باشند در صورتی که فاصله‌ی اعمال ورودی جدید را 6 نانو ثانیه 6 نانو ثانیه قرار دهیم انتظار داریم چون این بازه از تمامی تاخیر‌ها کمتر است با اعمال هر ورودی نتیجه‌ی نامربوط و غلط را ببینیم هر چند ممکن است پاسخ هر ورودی بعد از گذرا زمان طی اعمال ورودی‌های بعدی قابل مشاهده باشد اما با اعمال هر ورودی قطعاً در همان بازه قبل از اعمال ورودی بعدی نتیجه‌ی درست عملیات را مشاهده نخواهیم کرد حتی ممکن است در ورودی بعدی نیز نتیجه‌ی عملیات ورودی قبلی اماده و قابل مشاهده نباشد

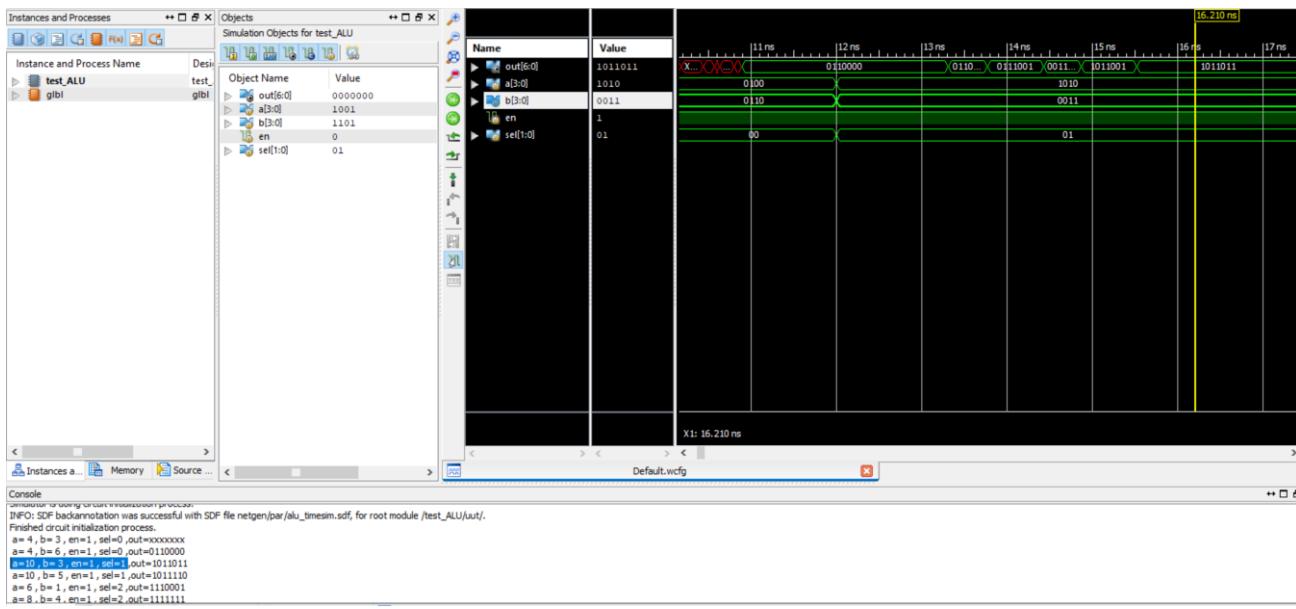
a= 4 , b= 3 , en=1 , sel=0 نتیجه این عملیات باید 0110000 باشد



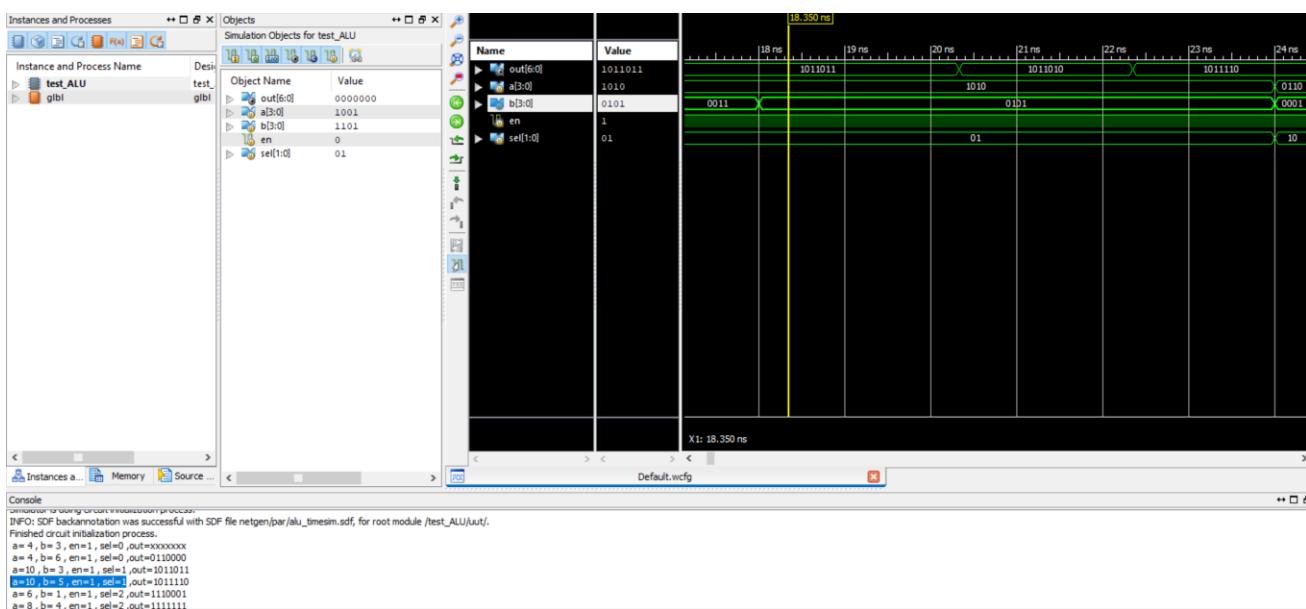
حتی در هنگام اعمال این ورودی نیز نتیجه این عملیات ورودی قبلی قابل مشاهده نیست خروجی نامربوط است



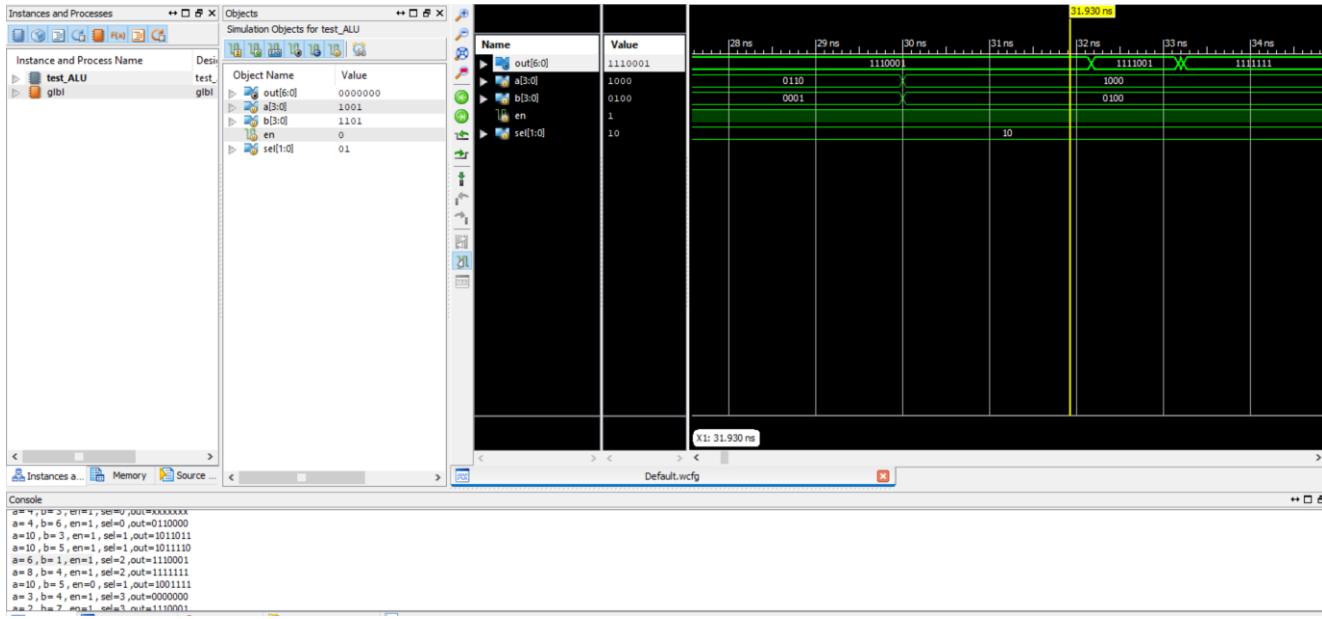
a=10 , b=3 , en=1 , sel=1 پاسخ این عملیات باید 1011110 باشد



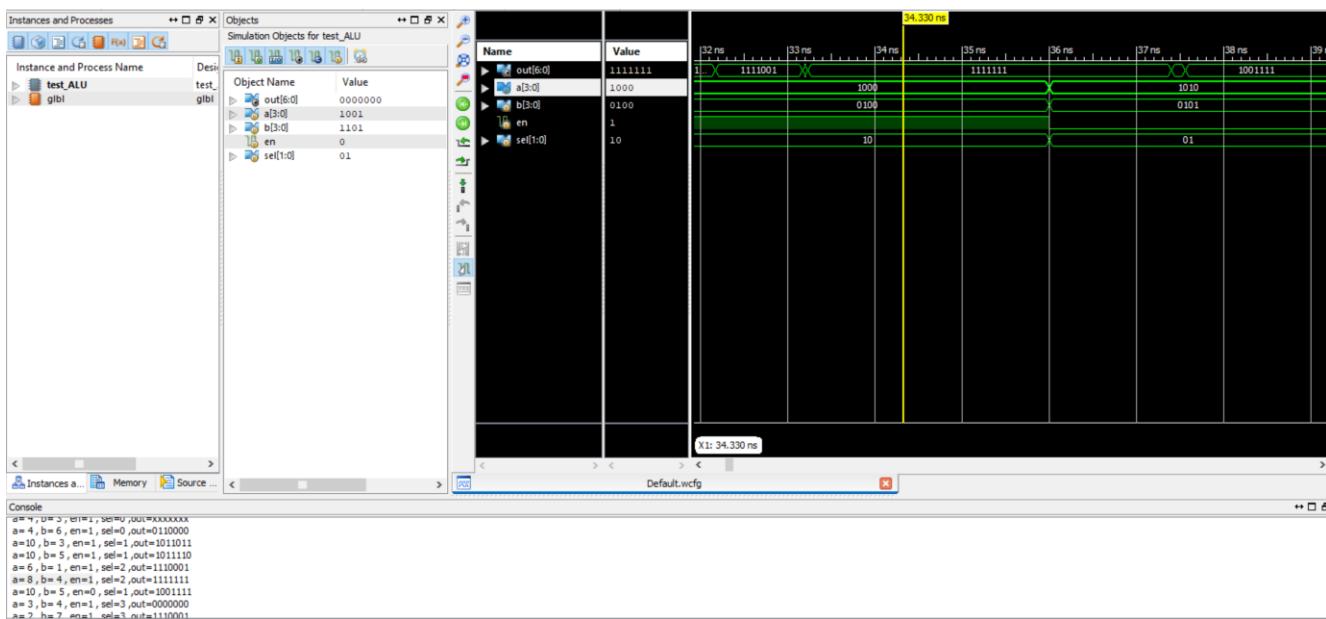
خروجی باید 1110001 باشد a=10 , b= 5 , en=1 , sel=1



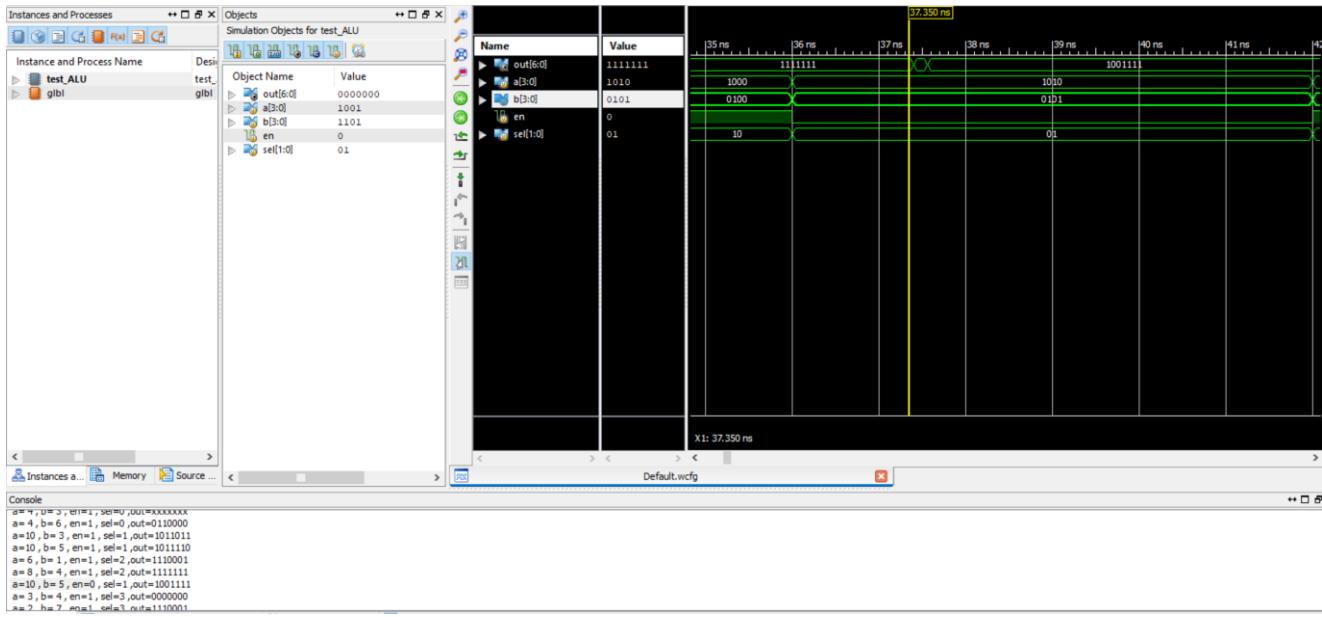
خروجی باید 1111111 باشد a=6 , b= 1 , en=1 , sel=2



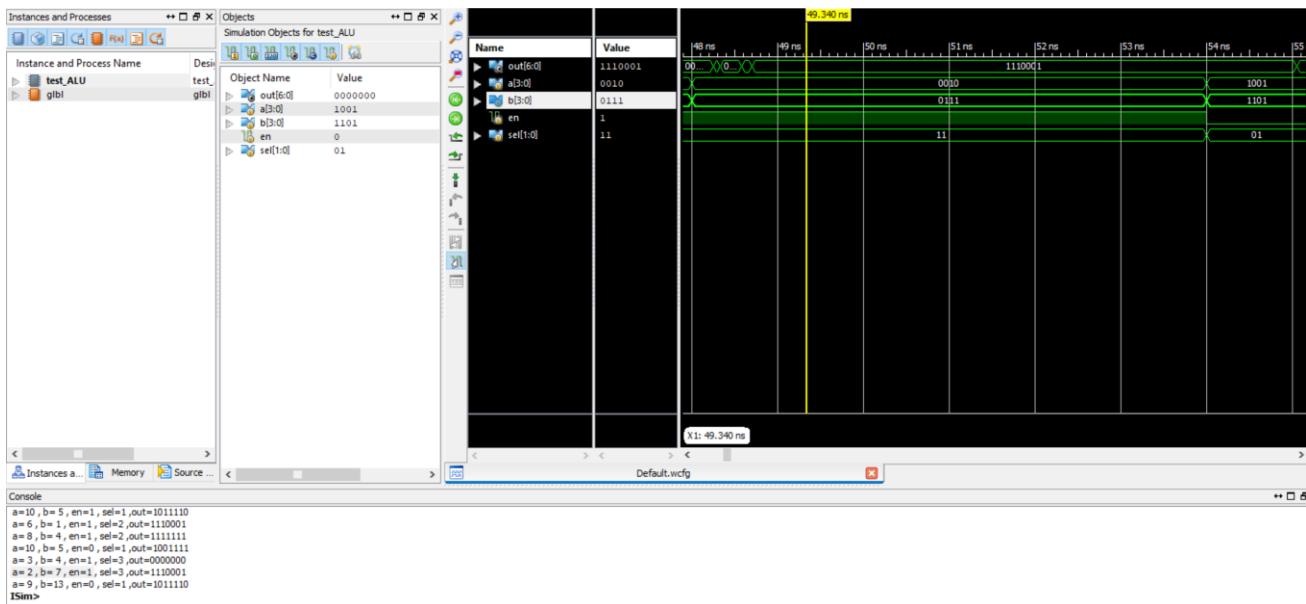
خروجی باید 1001111 باشد a=8 , b=4 , en=1 , sel=2



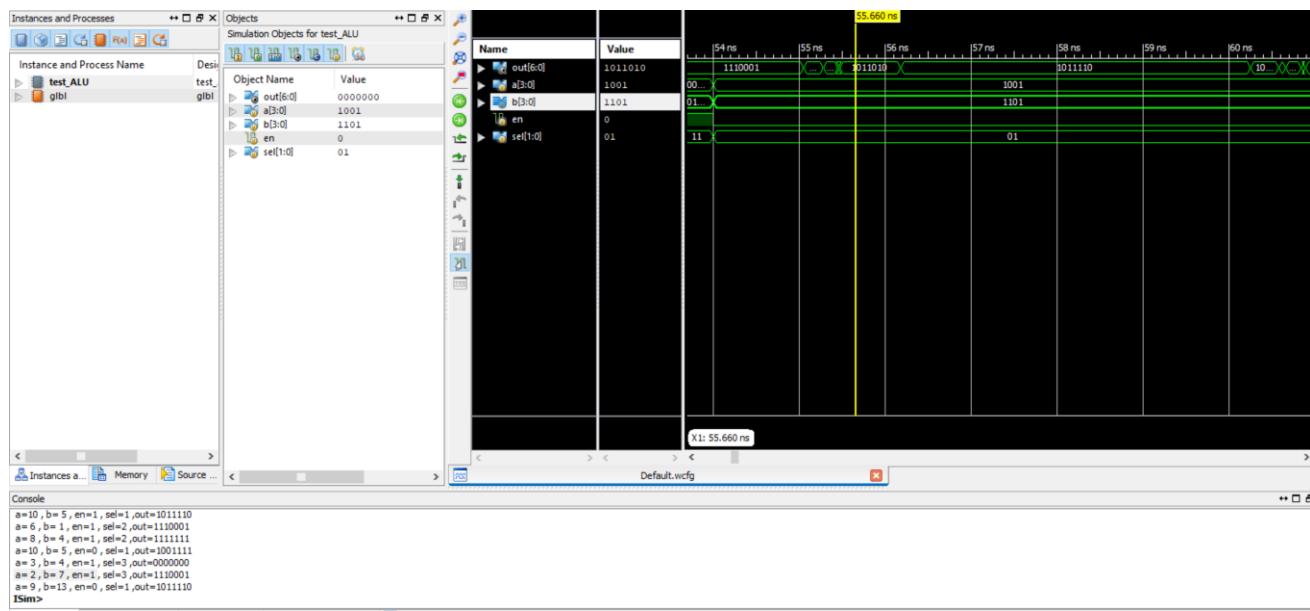
خروجی باید صفر باشد $a=10, b=5, en=0, sel=1$



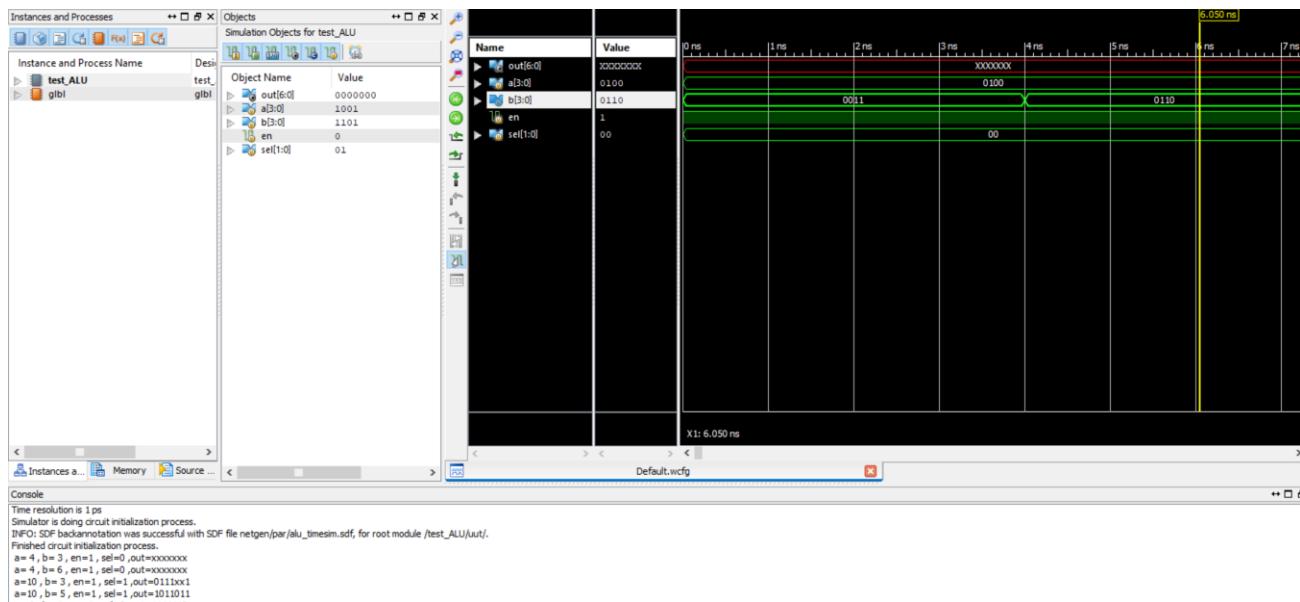
خروجی باید 1110001 باشد $a=2, b=7, en=1, sel=3$

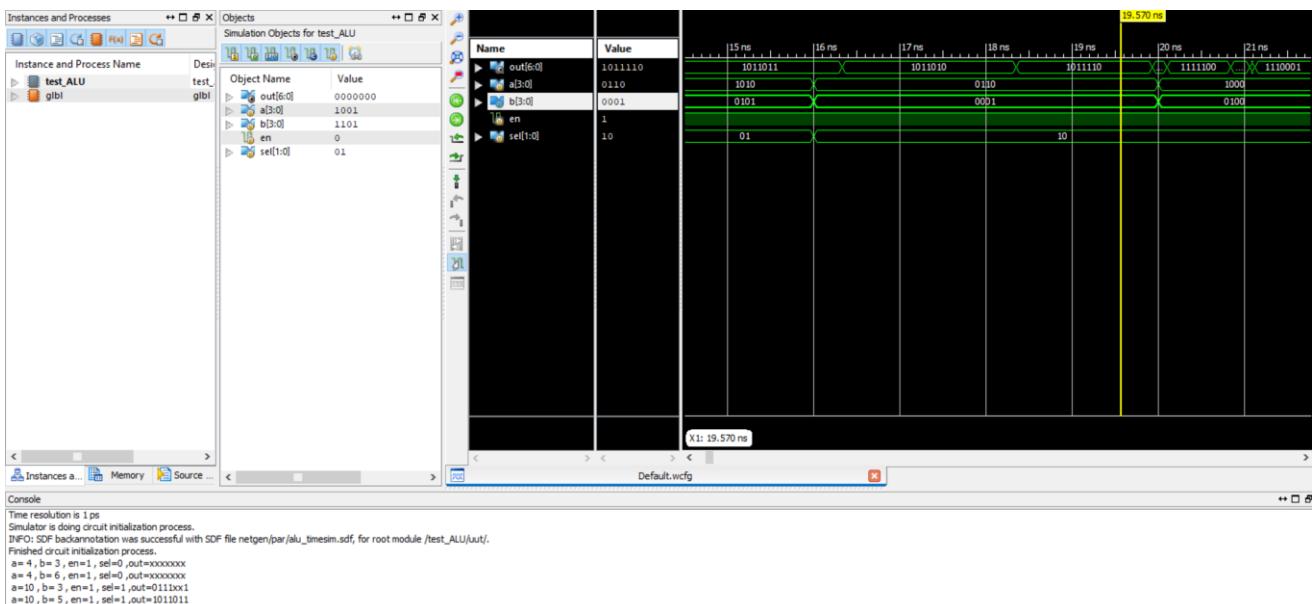
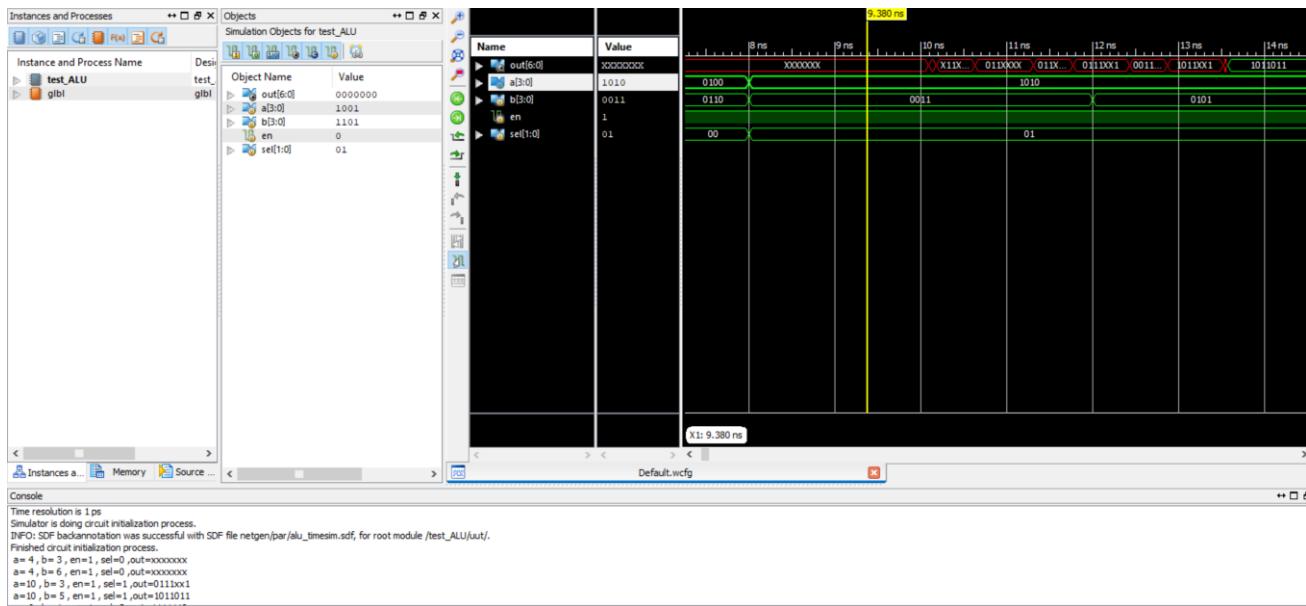


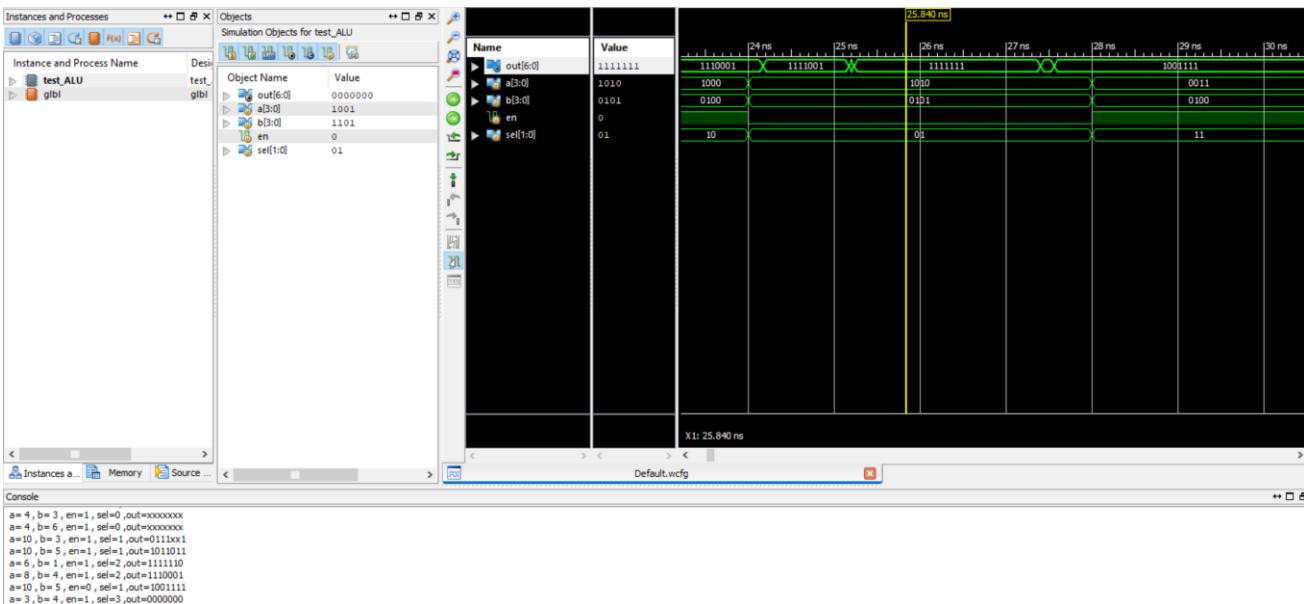
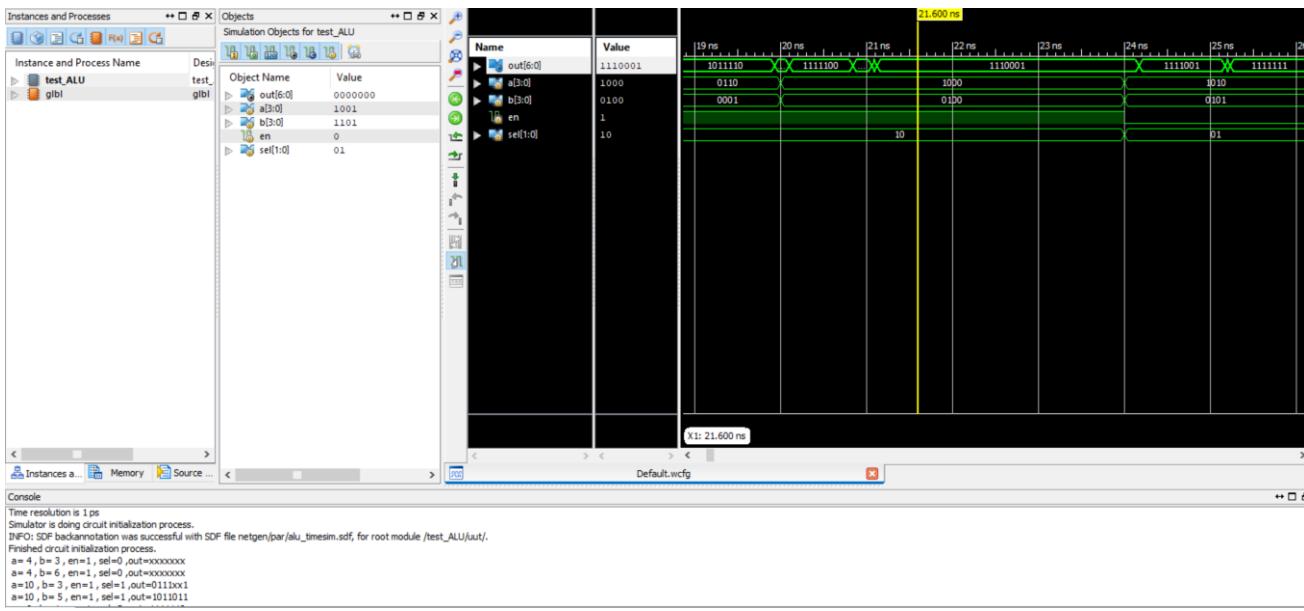
خروجی باید صفر باشد $a=9, b=13, en=0, sel=1$

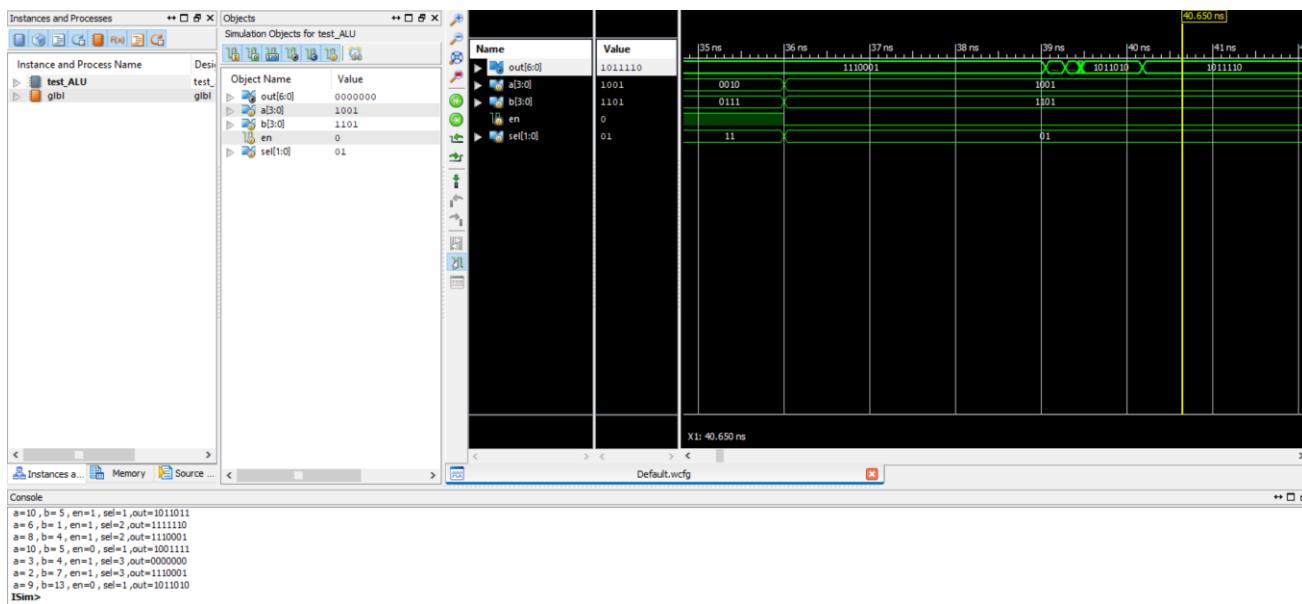
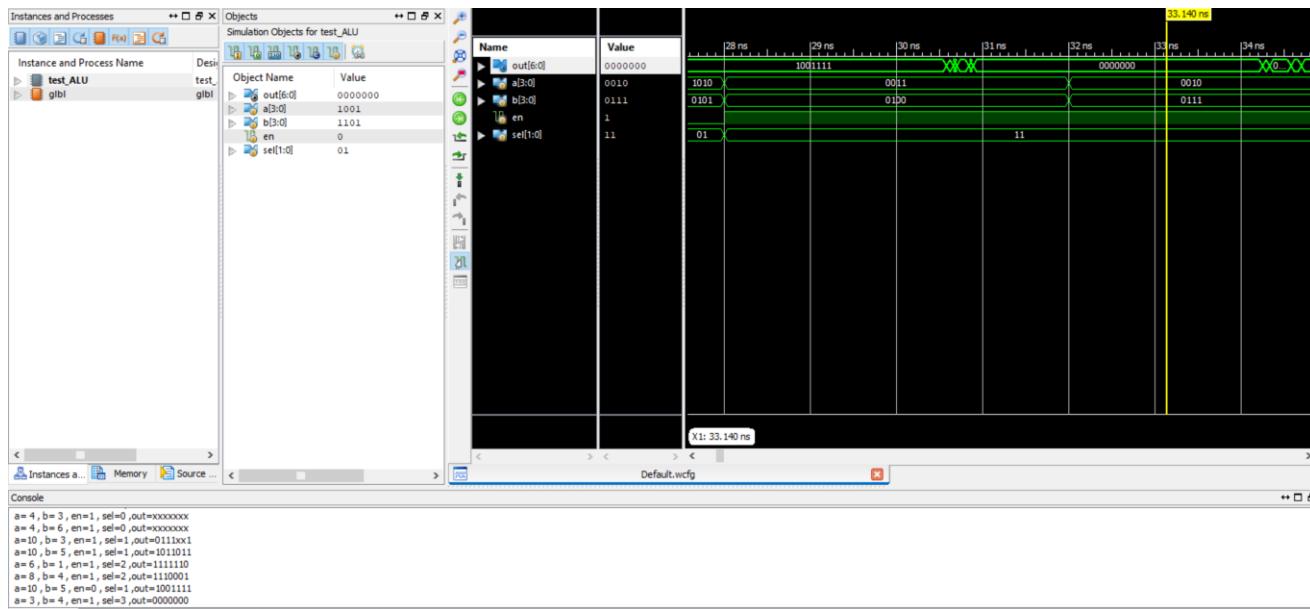


در این حالات ممکن است با گذر زمان بعضی از خروجی های مربوط به ورودی های مختلف را بتوان مشاهده کرد اما اگر فاصله‌ی بین ورودی ها را باز هم کاهش دهیم و به 4 نانو ثانیه برسانیم با توجه به تصاویر زیر مشاهده میشود که خروجی ها کاملاً نا مربوط و غلط هستند









بنابراین نتیجه میشود که باید هنگام اعمال ورودی ها دقت کرد در صورتی که ورودی ها با فاصله ای بیشتر از بیشترین تاخیر موجود در عملیات اعمال شوند با اعمال هر ورودی پاسخ مربوط به آن ورودی را قبل از اعمال ورودی بعدی خواهیم دید با کم کردن این فاصله ممکن است پاسخ هر ورودی را با اعمال آن نبینیم بلکه با فاصله بعد از اعمال ورودی بعد ببینیم اما اگر بیش از این مقدار فاصله بین اعمال ورودی ها کم شد یعنی قبل از پایدار شدن خروجی و تشکیل خروجی صحیح ورودی ها تغییر کنند خروجی های نامربوط به

ورودی و غلط دریافت میکنیم و دیگر با اعمال ورودی خروجی صحیحی دریافت نمیکنیم در واقع نمیتوان به میزان دلخواه فرکانس ورودی را زیاد و فاصله‌ی بین اعمال ورودی‌ها را کم کرد زیرا پاسخ ناپایدار و نتیجه‌ها غلط بدست می‌ایند