**به نام خدا**

**درس: طراحی سیستم های دیجیتال**

**نیم‌سال تحصیلی دوم 9900**

**مدرس: بهاروند**

| | | | |
|---|---|---|---|
| تمرین شماره: 3 | | موعد تحویل: 1400/02/13 ساعت ۲۳:55 | |
| نام و نام خانوادگی: | سارا آذرنوش | شماره دانشجویی: | 98170668 |

راه حل در همین فایل ارائه شود.

فایل به PDF تبدیل و در سایت بارگذاری شود.

**عنوان تمرین**

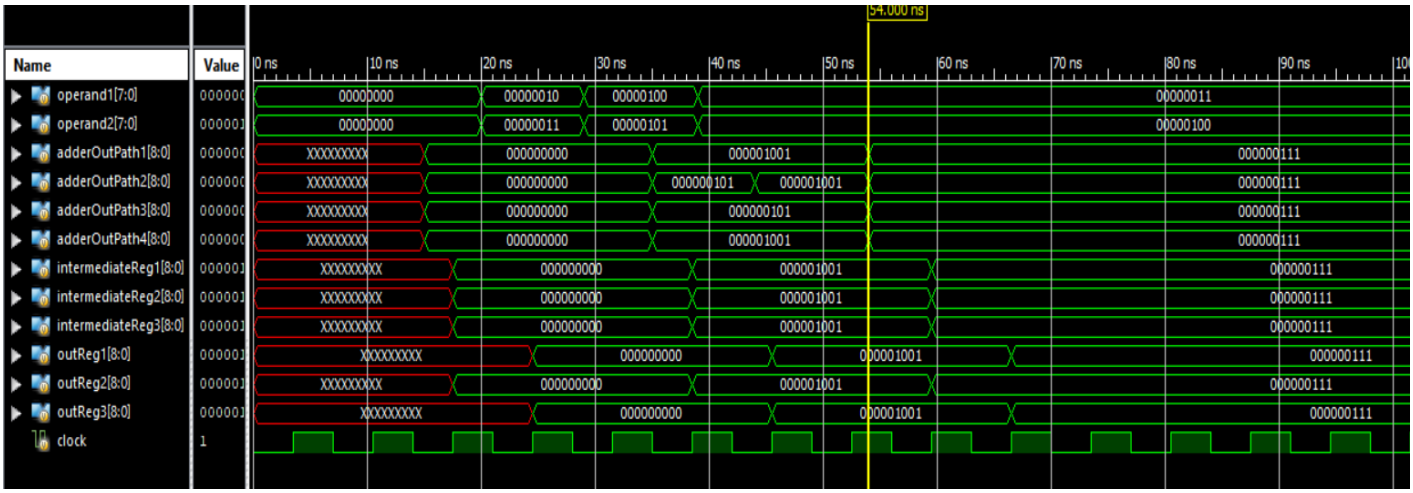
شبیه سازی و تحلیل رفتاری

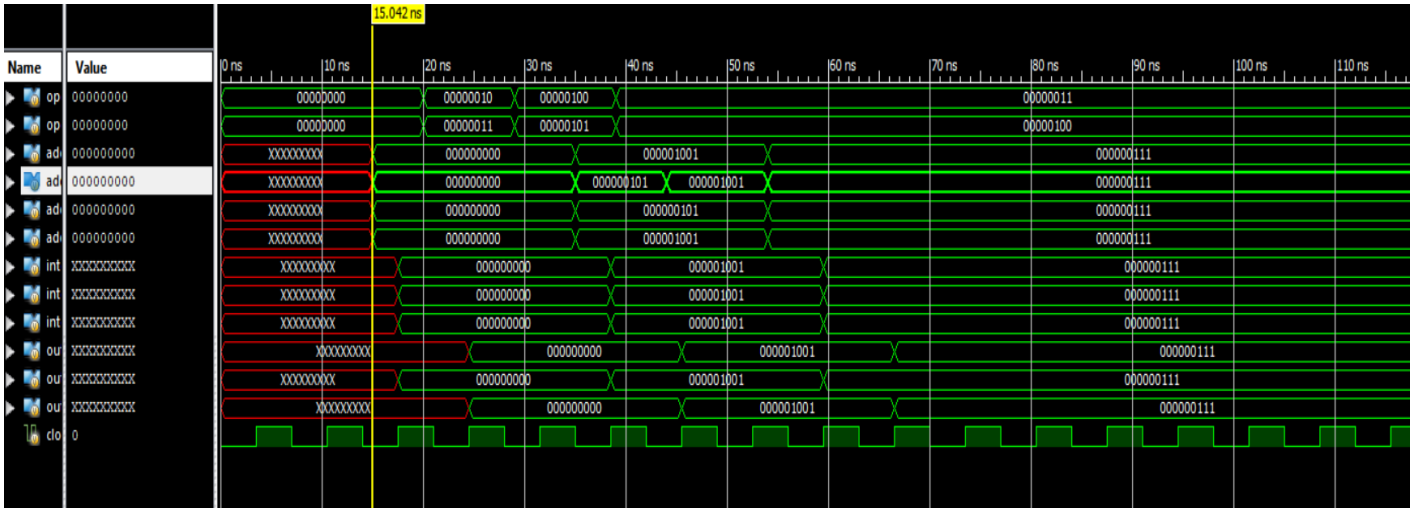
1. کد رفتاری Verilog که در اختیارتان گذاشته شده را شبیه سازی کنید. برای این کار یک پروژه با همان نام فایل درست کنید و سپس این کد را به آن اضافه کنید. با استفاده از اسکریپتی (run.do) که در اختیارتان قرار گرفته میتوانید شبیه سازی و خروجی را به صورت متنی و همچنین شکل موج مشاهده کنید.
2. رفتار هریک از بلوک های Procedural Assignment (ترکیبی و ترتیبی) را به دقت تحلیل کنید و در این گزارش بنویسید. بررسی کنید که خروجی های ترکیبی و رجیسترها و متغیرهای موقتی کدام ها هستند، چگونه رفتار میکنند و دلایل آن را بر اساس آموخته های خود شرح دهید.

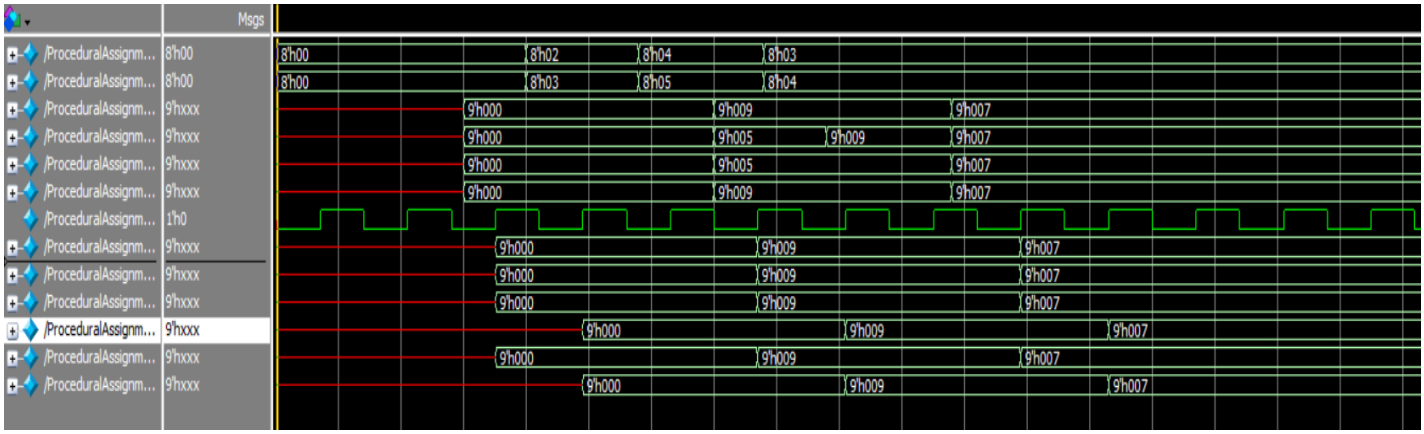
توجه1: از این نمونه اسکریپت (و تکمیل آن توسط خودتان) میتوانید برای شبیه‌سازی های بعدی خود در ModelSim استفاده کنید. برای اجرای اسکریپت در پنجره Transcript این دستور را اجرا کنید. source [*path_to_the_script*]/run.do

توجه 2: مسیر های داخل run.do بر اساس محل قرار گرفتن پروژه شما، ممکن است نیاز به اصلاح داشته باشد.

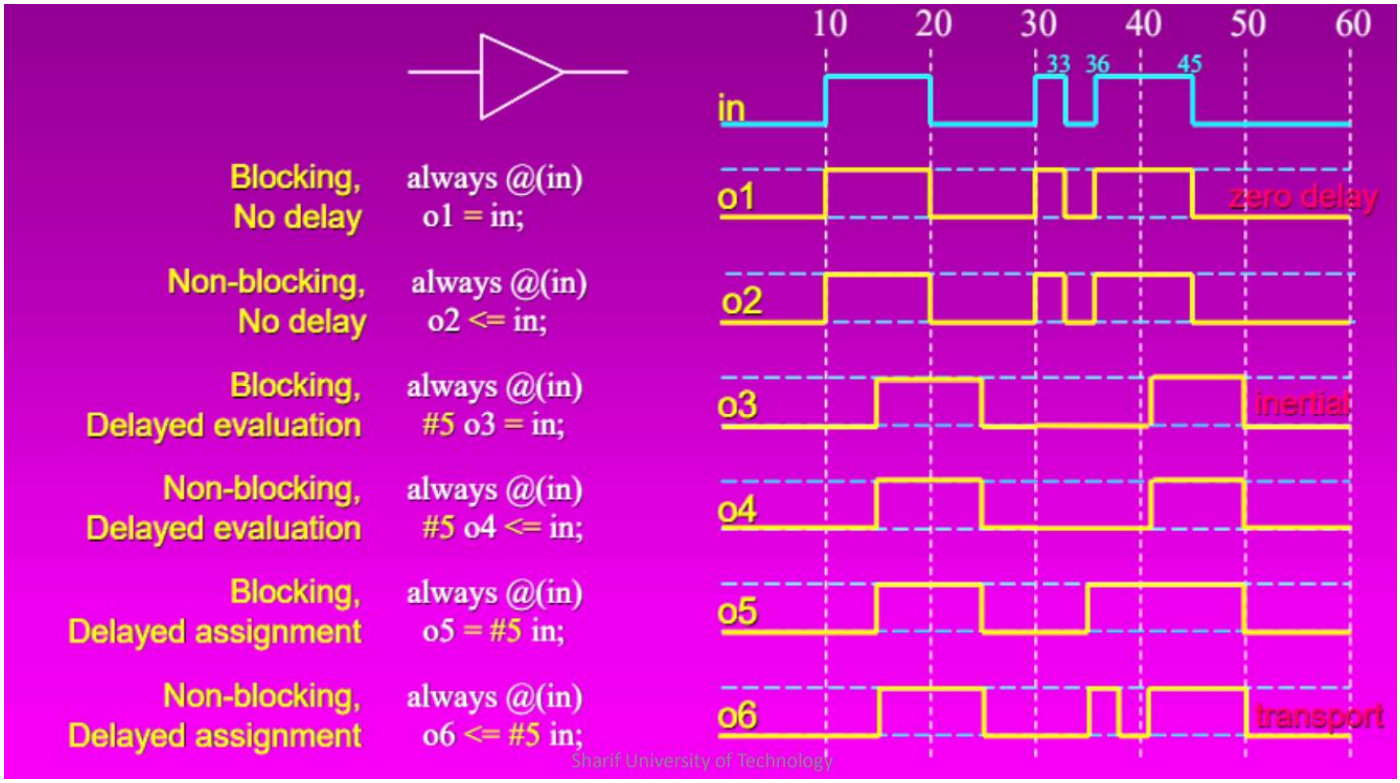
شکل موج‌ها به صورت زیر میشود.



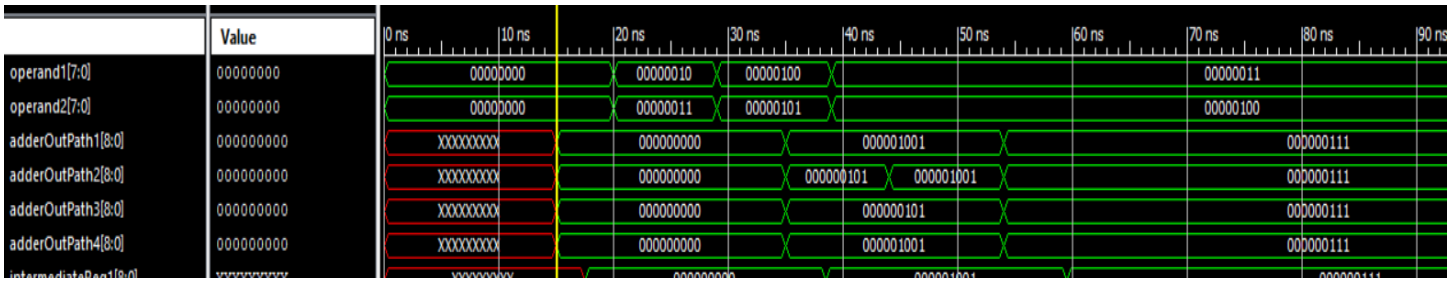



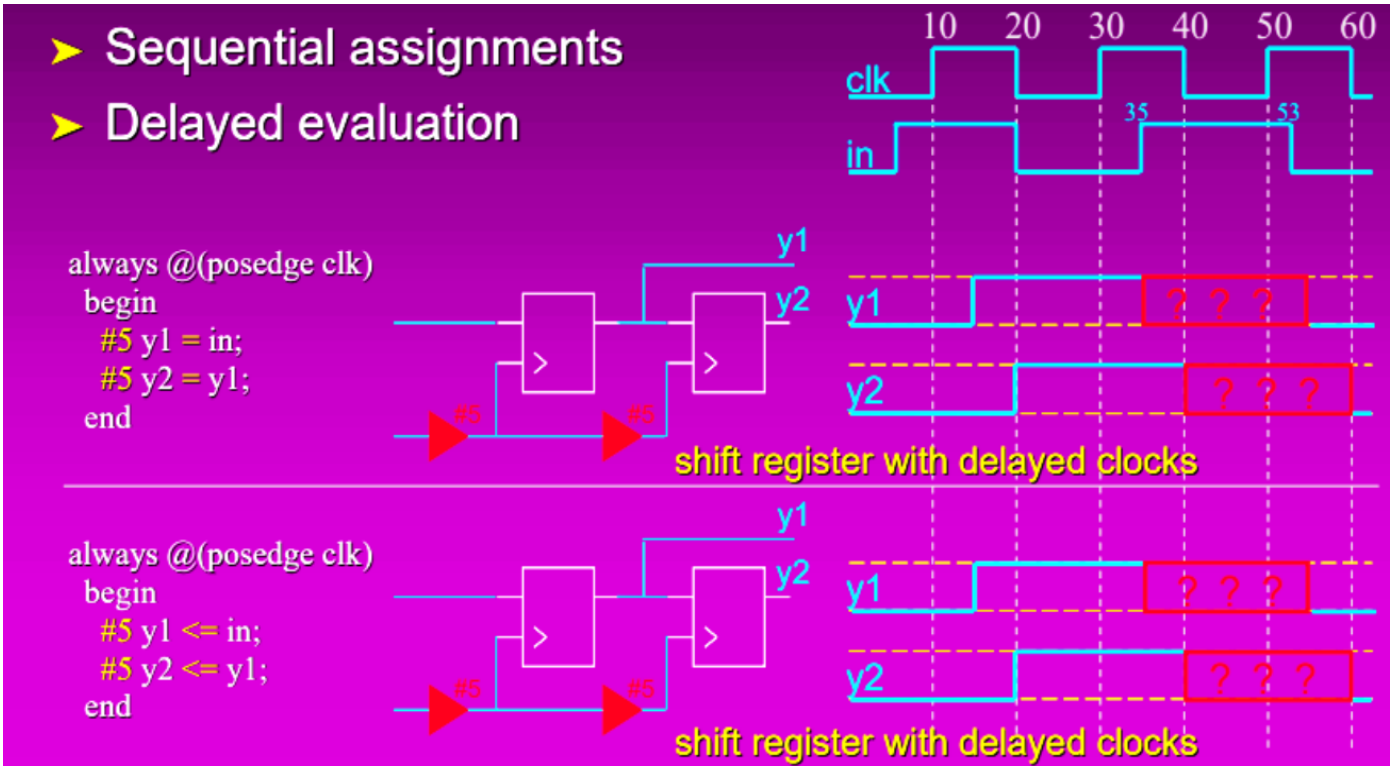



با توجه به مطالب و اسلایدهای کلاس تحلیل میکنیم.




4 بلاک اول به علت استفاده از *@always به تغییر هر یک از اعضایش حساس هستند و با تغییر آن‌ها اجرا میشوند.( **operand1** و **operand2**) و حاصل همه رجیستر است.






(دستورات متفاوت است و شکل به این صورت نمیشود به عنوان مرجع از اسلاید است)

**Adder 1:**

**#15 adderOutPath1 = operand1 + operand2;**

blocking assignment است که با توجه به محل delayed evaluation پس از گذشت تاخیر 15 واحد نسبت به تغییر اعضایش حاصل جمع آخرین اعداد را در adderOutPath1 می ریزد. تا قبل از پایان تاخیر این دستور اجرا نخواهد شد و پس از پایان آخرین مقداری که اعداد دارند محاسبه میشود. این مدار به صورت ترکیبی است.

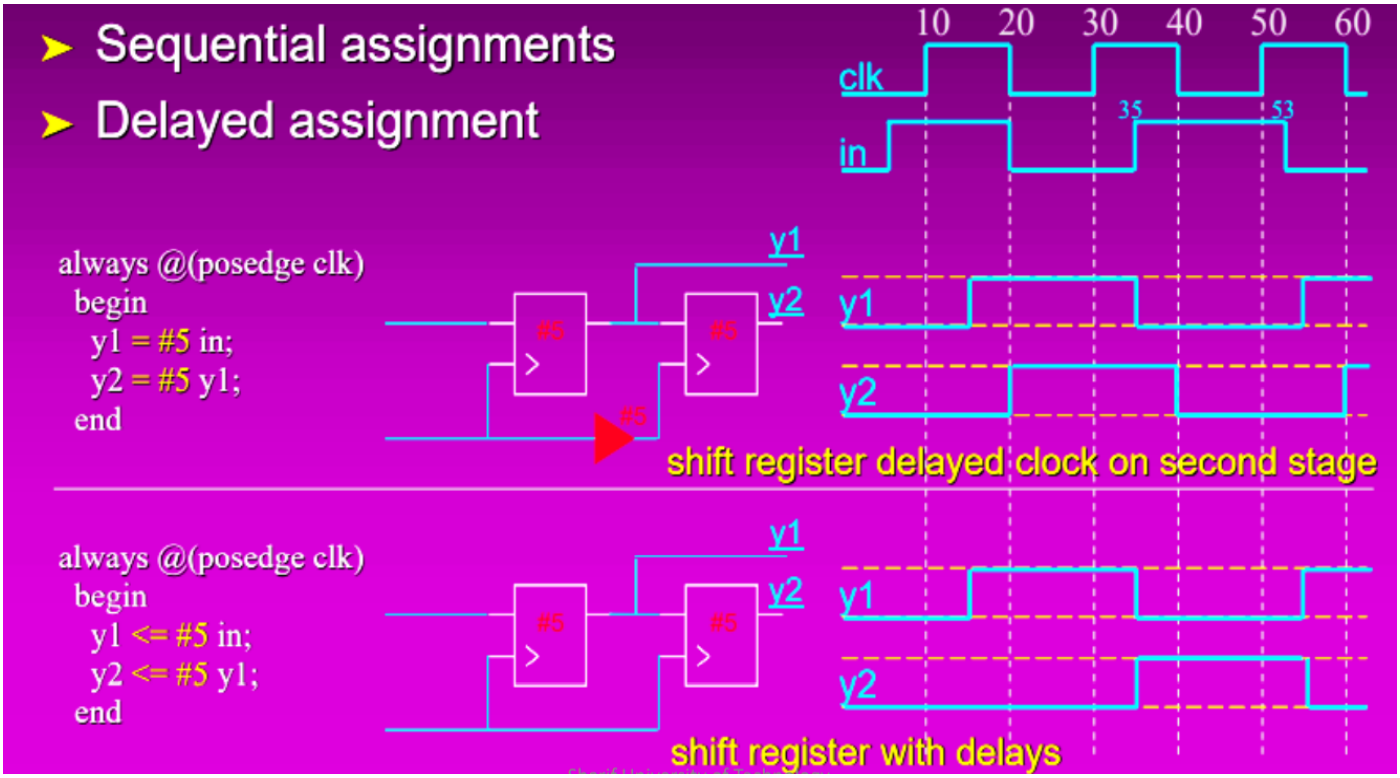
جمع 3 و 2 که اولین تغییر ورودی ها هستند به عنوان جواب خروجی داده نمی‌شود زیرا پس از تغییر 15 واحد صبر میکند و سپس مقدار آخر را جمع میکند و اولین جواب حاصل جمع 5 و 4 است (مقداری که پس از پایان مدت زمان تاخیر اپرندها دارند) و با تغییر ورودی‌ها به 3 و 4 حاصل جمع 7 را به عنوان خروجی می‌دهد.

**Adder 4:**

**#15 adderOutPath4 <= operand1 + operand2;**

non-blocking assignment است که با توجه delayed evaluation، پس از گذشت تاخیر 15 واحدی نسبت به تغییر اعضایش حاصل جمع آخرین اعداد را در adderOutPath4 می ریزد. تا قبل از پایان تاخیر این دستور اجرا نخواهد شد و پس از پایان آخرین مقداری که اعداد دارند محاسبه میشود. این مدار به صورت ترکیبی است. مانند حالت قبل است و تنها در non-blocking متفاوت است.

جمع 3 و 2 که اولین تغییر ورودی ها هستند به عنوان جواب خروجی داده نمی‌شود زیرا پس از تغییر 15 واحد صبر میکند و سپس مقدار آخر را جمع میکند و اولین جواب حاصل جمع 5 و 4 است (مقداری که پس از پایان مدت زمان تاخیر اپرندها دارند) و با تغییر ورودی‌ها به 3 و 4 حاصل جمع 7 را به عنوان خروجی می‌دهد




(دستورات متفاوت است و شکل به این صورت نمیشود. به عنوان مرجع از اسلاید است)

**Adder 3:**

**adderOutPath3 = #15 (operand1 + operand2);**

blocking assignment است و ابتدا ورودی ها گرفته و منتظر می ماند تا زمان تاخیر تمام شود و به مقادیر جدید ورودی‌ها در مدت زمان تاخیر توجه نمی‌کند و پس از گذشت 15 واحد تاخیر در خروجی نمایان می شود.

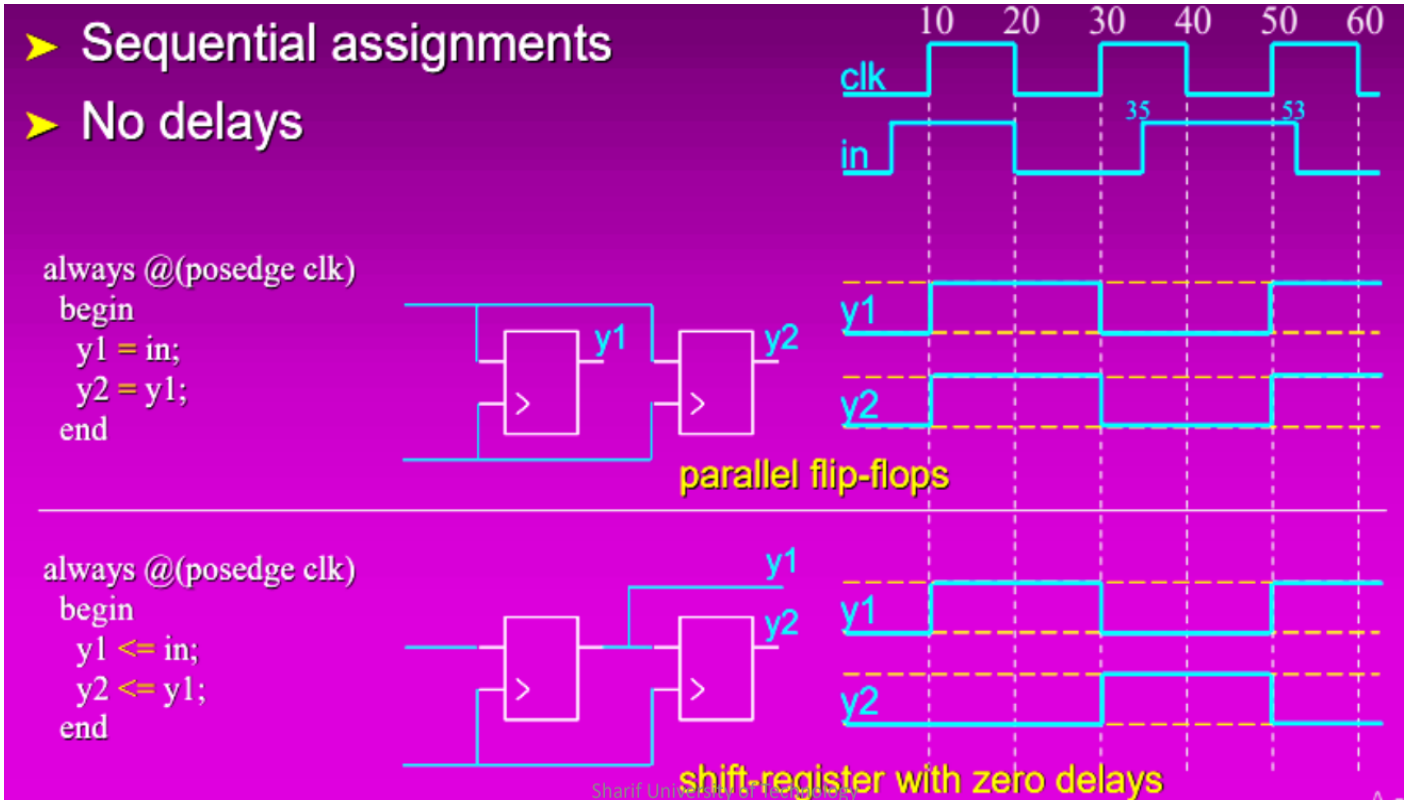
اعداد 2 3 وارد شده جمع 2+3=5 پس از 15 ثانیه نمایش داده شده و سپس مقدار موجود 3 و 4 هستند که محاسبه و نمایش داده میشوند.( 4 و 5 در زمان تاخیر ابتدایی وارد شدند بنابراین چون مقدار 2 و 3 بودند حاصل 4+5 نمایش داده نشد.)

**Adder 2:**

**adderOutPath2 <= #15 (operand1 + operand2);**

non-blocking assignment است تاخیر پس از برسی کردن ورودی ها به مدت 15 واحد و قبل از اساین کردن حاصل جمع به خروجی اتفاق می‌افتد. پس شبیه ساز منتظر اتمام این دستور نمی‌ماند و هر وقت ورودی ها تغییر کند، حاصل جمع ورودی های جدید با یک تاخیر 15 واحدی به خروجی میرود. 15 واحد بعد از هر تغییر ورودی، جواب در خروجی نمایش داده میشود. این مدار به صورت ترکیبی است.

اعداد 2 3 وارد شده جمع 2+3=5 پس از 15 ثانیه نمایش داده شده و سپس به همین صورت 4 و 5 در این بین وارد شده و 4+5= 9 پس از 15 ثانیه از وارد شدن 4 و 5 خروجی داده میشود و در آخر 3+4=7 پس از 15 ثانیه از تغییر و وارد شدن نمایش داده میشود.




بلاک‌های زیر با هر تغییر کلاک آغاز به کار میکنند.

**blocking in sequential:**

**intermediateReg2 = adderOutPath1;**

**outReg2 = intermediateReg2;**

blocking assignment است و به لبه ی بالا رونده ی clock حساس است و دستورات به ترتیب اجرا میشوند. و چون adderOutPath1 به intermediateReg2 رفته و مقدار جدید آن به outReg2 میرود مقدار intermediateReg2 همان adderOutPath1 است ورودی هر دو مقدار یکسان adderOutPath1 میشود.intermediateReg2 موقت است و outReg2 غیر موقت و رجیستر است.

با فلیپ فلاپ مانند شکل اول شبیه سازی میشود.

**non-blocking in sequential:**

**intermediateReg1 <= adderOutPath1;**

**outReg1 <= intermediateReg1;**

non-blocking assignment است و به لبه‌ی بالارونده‌ی clock حساس است. وقتی لبه‌ی کلاک بالارونده‌ی است مقدار adderOutPath1 بر روی intermediateReg1 قرار میگیرد و همان موقع مقدار قبلی intermediateReg1 به outReg1 میرود. و در اولین مقدار دهی مقدار نامشخص intermediateReg1 به outReg1 خواهد رفت و outReg1 همیشه مقدار یک کلاک قبل از intermediateReg1 را دارد و یک کلاک عقب تر است. هر دو غیر موقت و رجیسترند.

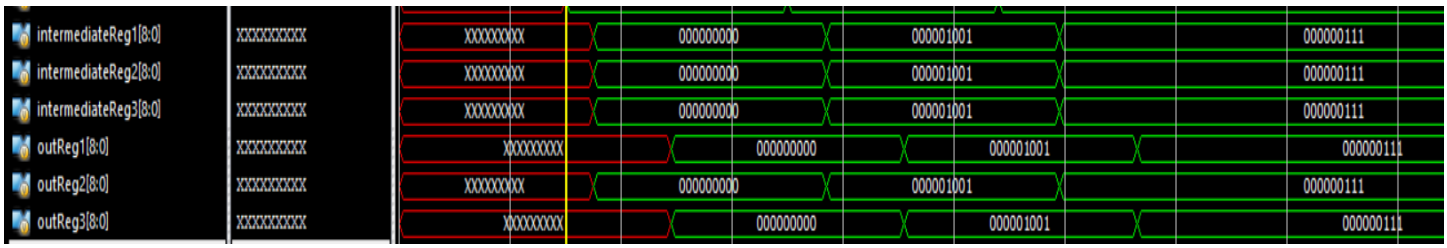
با فلیپ فلاپ مانند شکل دوم شبیه سازی میشود.

**blocking in sequential (order changed)**:

```
outReg3         = intermediateReg3;
intermediateReg3 = adderOutPath1;
```

blocking است و به لبه ی بالا رونده ی clock حساس می باشد و دستورات به ترتیب اجرا میشوند. در اینجا ترتیب متفاوت است و در ابتدا مقدار intermediateReg3 نامشخص است و در outReg3 می رود وadderOutPath1 در intermediateReg3 میرود و در کلاک بعدی مقدار سابق adderOutPath1 که در intermediateReg3 رفته بود به outReg3 میرود به همین ترتیب همیشه مقدار outReg3 یک کلاک عقب‌تر از adderOutPath1 است. و intermediateReg3 مانند اولی موقت نیست.

با فلیپ فلاپ مانند شکل دوم شبیه سازی میشود.



2 بلاک پایین برای تست کردن مدار است که به ورودی‌ها مقدار داده میشود و از بلاک initial که تنها یک بار استفاده میشود و سنتز نیز نمیشود. استفاده شده است.

**generating clock for the test bench:**

در ابتدا و یک بار به کلاک مقدار 0 داده شده و سپس هر 3.5 واحد مقدار مخالف آن در خود نوشته میشود.

**applying inputs and monitoring the results:**

تنها یک بار انجام شده و مقادیر مختلف oprand1 و oprand2 در بازه زمانی های مختلف معین میشود.