

گزارش کار آزمایشگاه DSD

**آزمایش شماره 5**



26 فروردین 1400

عرشیا اخوان

محمدحسین عبدی

علیرضا ایلامی

|  |  |  |
| --- | --- | --- |
| تاریخ آزمایش: 26 فروردین 1400 | موضوع: طراحي ضرب كننده | **شماره آزمایش: 5** |
| **علیرضا ایلامی**  **97101286** | **محمدحسین عبدی**  **97110285** | **عرشیا اخوان**  **97110422** |

مقدمه:

در این آزمایش ما باید ضرب به روش booth را پیاده سازی کنیم.

شرح آزمایش:

ابتدا یک سیگنال start داریم که باید حتما 1 شود تا الگوریتم شروع شود

تعدادی رجیستر داریم به نام های a,b, x که قرار است رجیستر های a, b در یکدیگر ضرب شوند و در b نوشته شوند. X هم رجیستر کمکی است. و خروجی نهایی حاصلضرب xb است.

یک کانتر n داریم که تعدد مراحل ضرب را نشان میدهد

یک سیگنال end flag داریم که وقتی 1 میشود یعنی خروجی ما آماده است.

یک رجیستر b-1 هم هست که موقع شیفت دادن xb بیرون انداخته میشود. (بیت سمت راست عدد xb)

در ابتدا اعداد را لود میکنیم در a, b

به جای الگوریتم ضرب عادی، ما در روش بوث می آییم در b ، ابتدا index اولین بیت یک، ایندکس آن را i میگیریم. تا وقتی بیت های بعدی 1 هستند ادامه میدهیم و کاری نمیکنیم، تا زمانی که دوباره به یک بیت 0 برسیم و اندیس آن نیز j است.

چیزی که برای ما مهم میشود، دو به توان j منهای دو به توان i است. یعنی

2^j - 2^i

علت آن هم این است که مثلا 111 که برابر با 7 است، برابر با دو به توان 3 منهای دو به توان صفر است. بنابراین الگوریتم بوث تعداد operation های ما را به شدت کاهش میدهد و نتیجه را نیز کاملا درست محاسبه میکند.

ا

کار ما بطور خلاصه این است:

بیت های b0 و b-1 را چک میکنیم.

اگر که b0 b-1 برابر با 00 یا 11 بود:

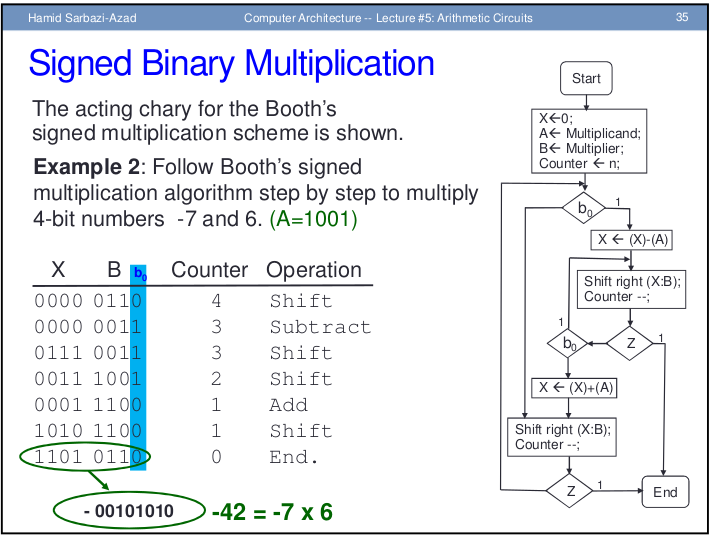
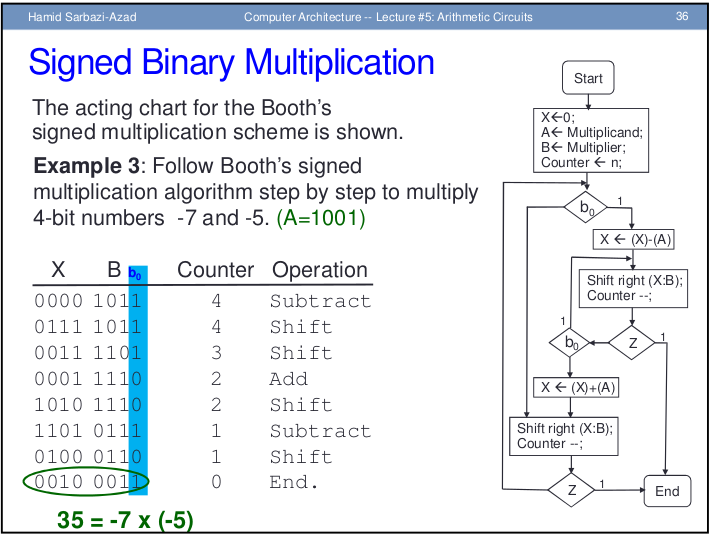
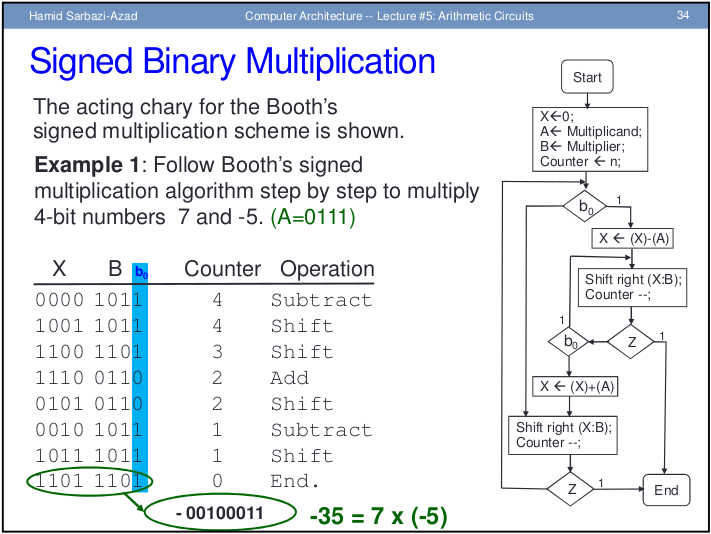
یعنی ما وسط دنباله متوالی در الگوریتم بوث هستیم و نیازی نیست کاری انجام دهیم. و به راحتی فقط شیفت میدهیم و فقط از کانتر یک واحد کم میکنیم.

اگر برابر با 10 بود: باید شروع کنیم به کم کردن دو عدد و سپس یک واحد از کانتر کم میکنیم

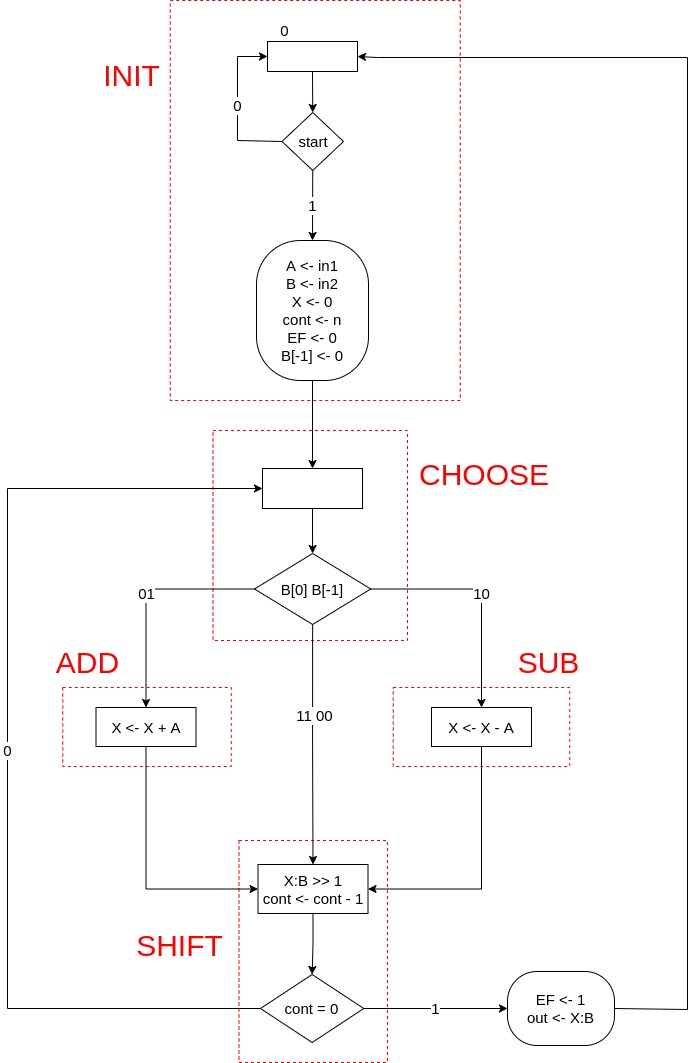
اگر برابر با 01 بود: باید شروع کنیم به جمع کردن دو عدد و مثل قبل کانتر را یک واحد کم میکنیم

واضح است که کل الگوریتم زمانی خاتمه پیدا میکند که کانتر ما برابر با 0 شده باشد

تعدادی مثال به روش booth اورده شده است:



تحلیل نمودار ASM chart



در این نمودار ما 5 تا state داریم:

Init, Choose, Sub, Add, Shift

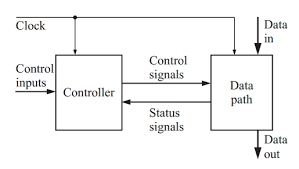
در init state که اگر سیگنال start = 0 باشد همینطور کلاک زده میشود تا وقتی که start = 1 شود و سیگنال های اولیه ست گردند و ما به استیت بعدی برویم.

در choose state ما یک باکس خالی داریم. علت آن در زیر شرح داده شده است:

میدانیم که تمام دستورات در یک asm block که با مستطیل نشان داده میشود، بصورت کاملا موازی و همزمان انجام میشوند. و همچنین هر کدام از این بلاک ها نمایانگر یک کلاک واحد زمانی هستند. ضمن اینکه میدانیم تمام تغییرات ما در یک asm block در بلاک بعدی این تغییرات ما اعمال میشوند.

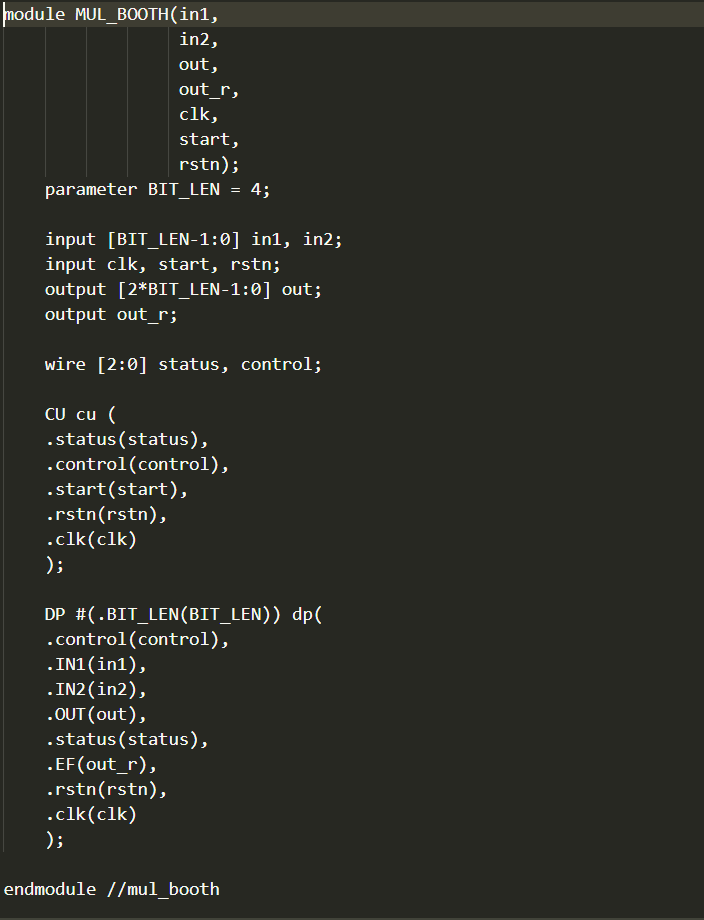
بنابراین آن بلاک خالی باعث میشود شروط b0, b-1 بعد از اجرای استیت های ADD و SUB و SHIFT به درستی مقدار دهی شوند.

تحلیل کد:

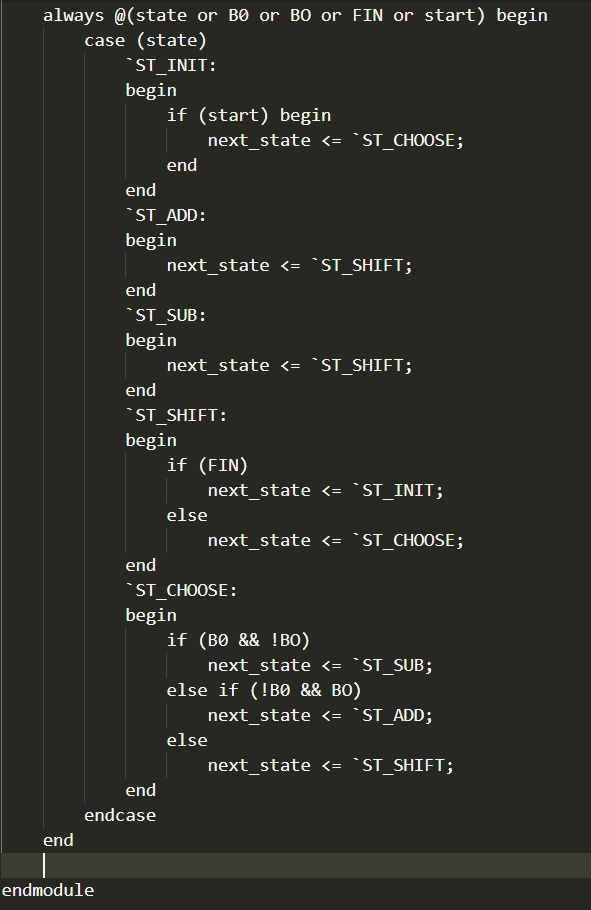
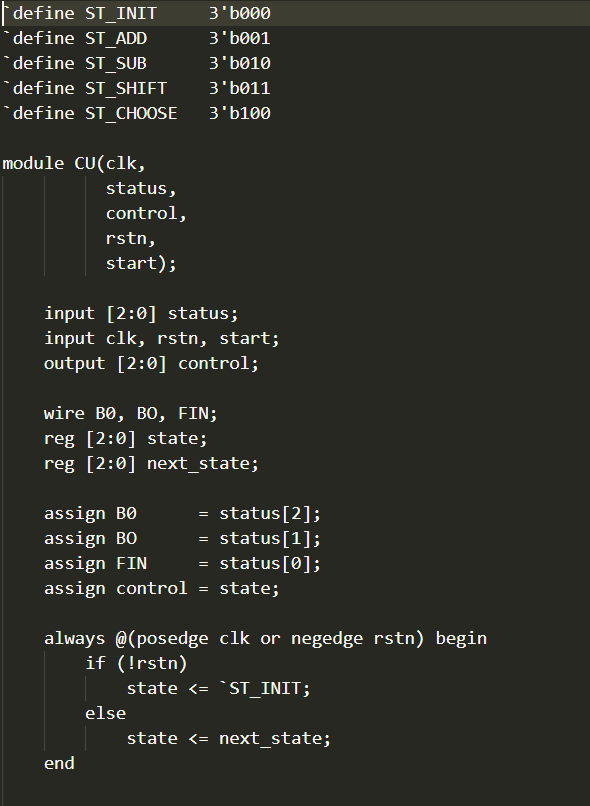


شمایی از ساختار کلی ماژول:

داده ها وارد دیتاپت میشوند، دیتاپت signal های وضعیت فعلی را به واحد Controller که در اینجا Control Unit است میدهد، و واحد کنترلر تصمیم میگیرد که next state چیست و آن را در current state بریزد. و این نتیجه را به دیتاپت اطلاع میدهد. هر دوی این بلاک ها با کلاک کار میکنند.

Mul booth

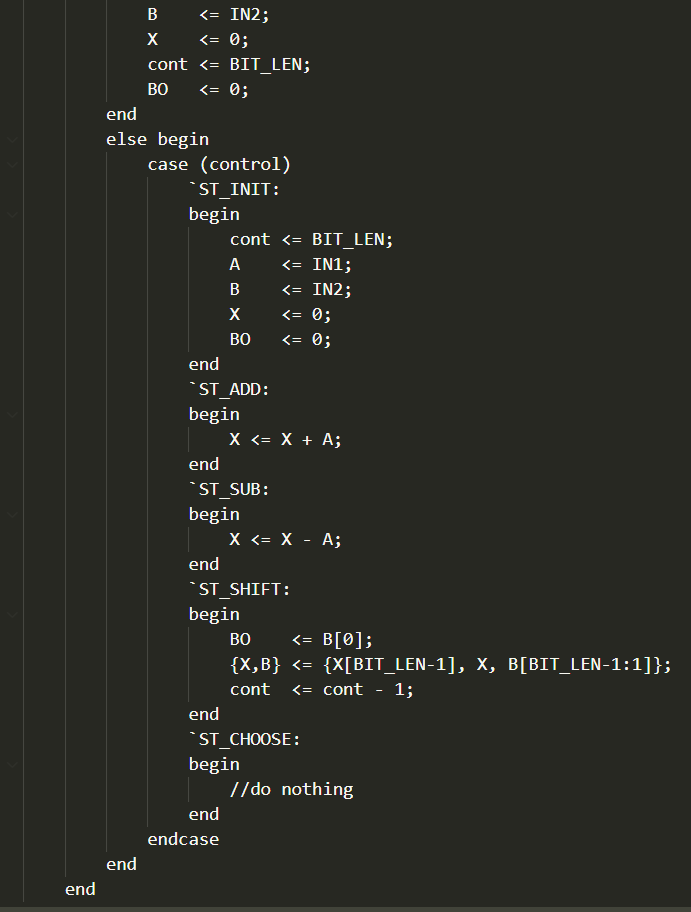
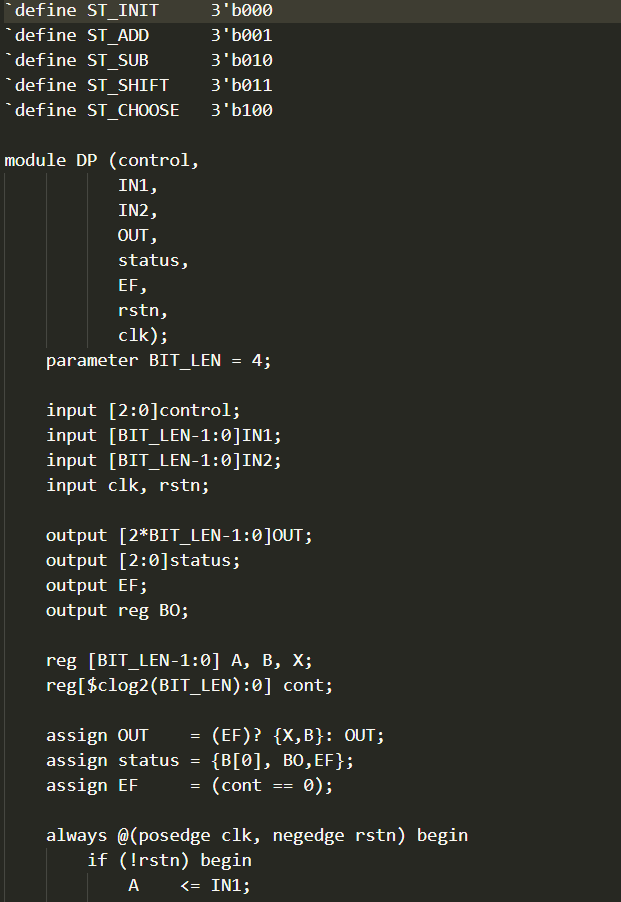
این ماژول صرفا کنترل یونیت و دیتاپت را وصل میکند و حکم واسط را دارد. اصل کار در دو ماژول دیگر است.

Control unit

هربار next state مشخص میشود و در استیت فعلی ریخته میشود.

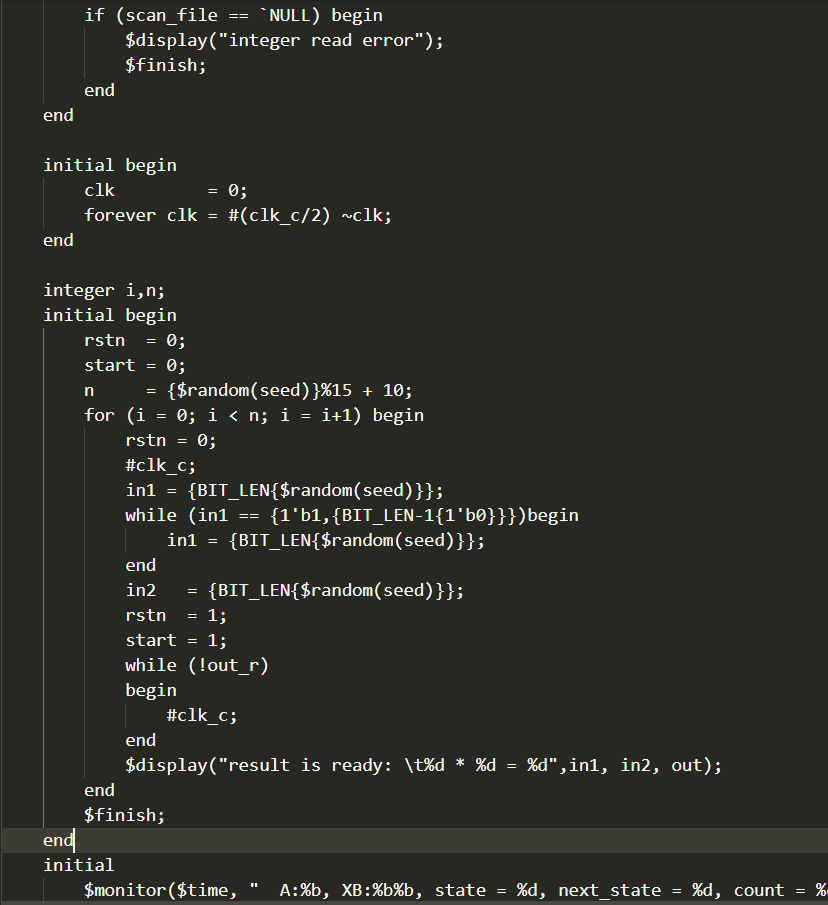
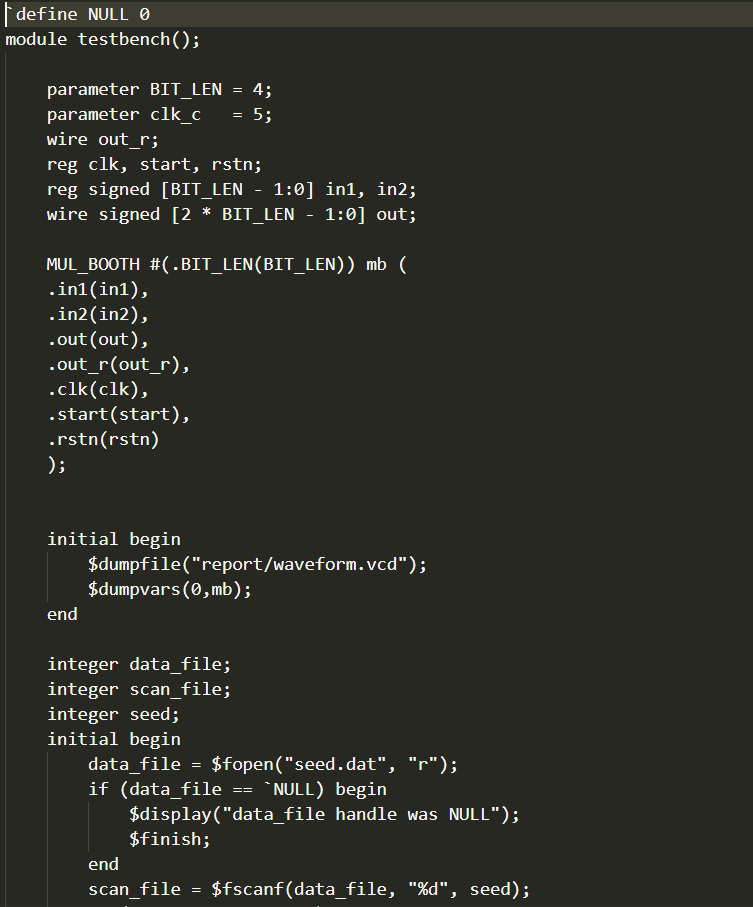
نکست استیت از روی سیگنال های کنترلی که دیتاپت می فرستد آپدیت میشوند.

Datapath



استیت ها را به دیتاپت میدهیم و دیتاپت متناسب با اینکه در چه استیتی قرار دارد، کارهای مرتبط با آن استیت را انجام میشود

تست بنچ:



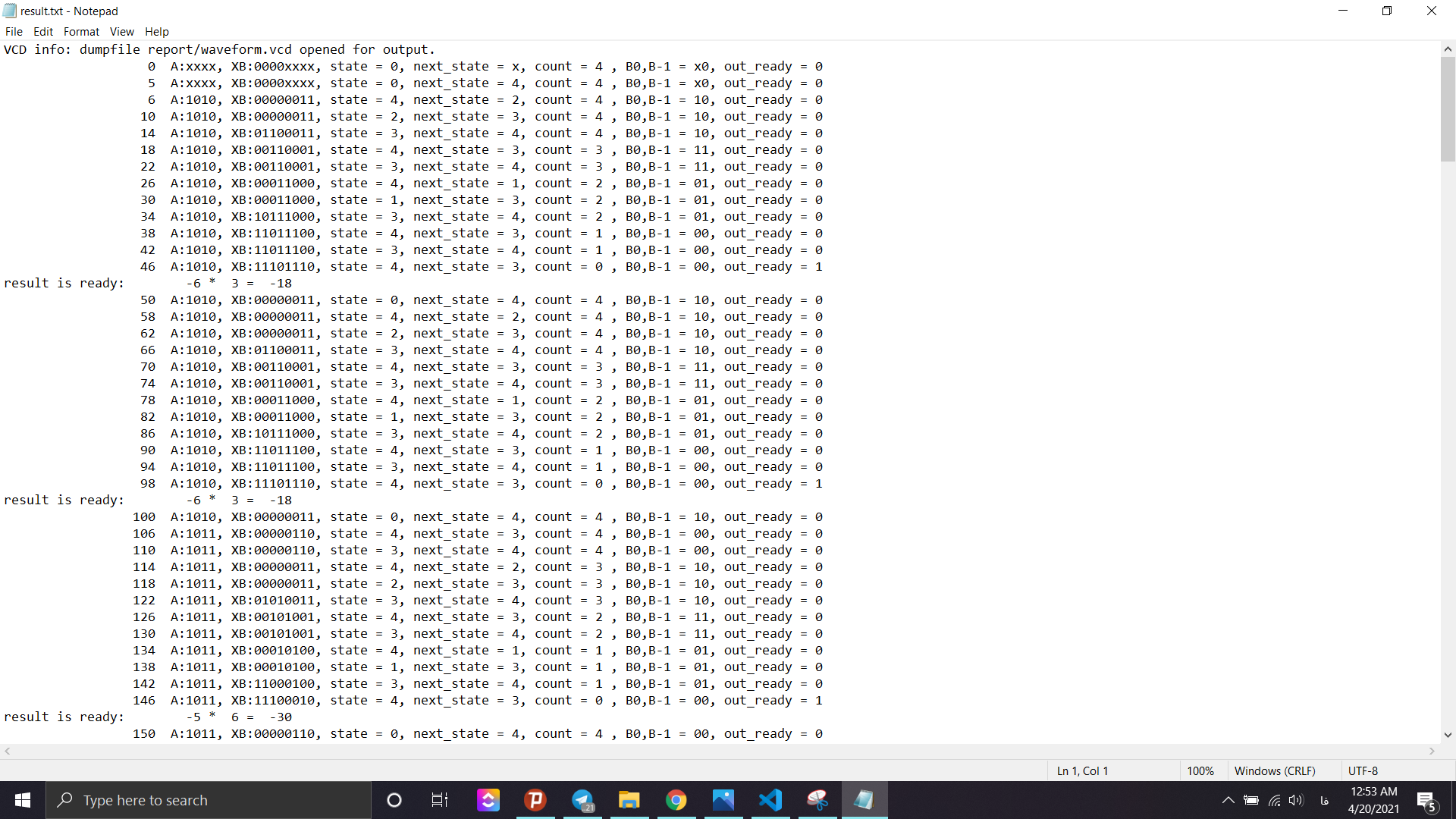
مثل ازمایش سوم ما از عدد رندوم استفاده میکنیم.

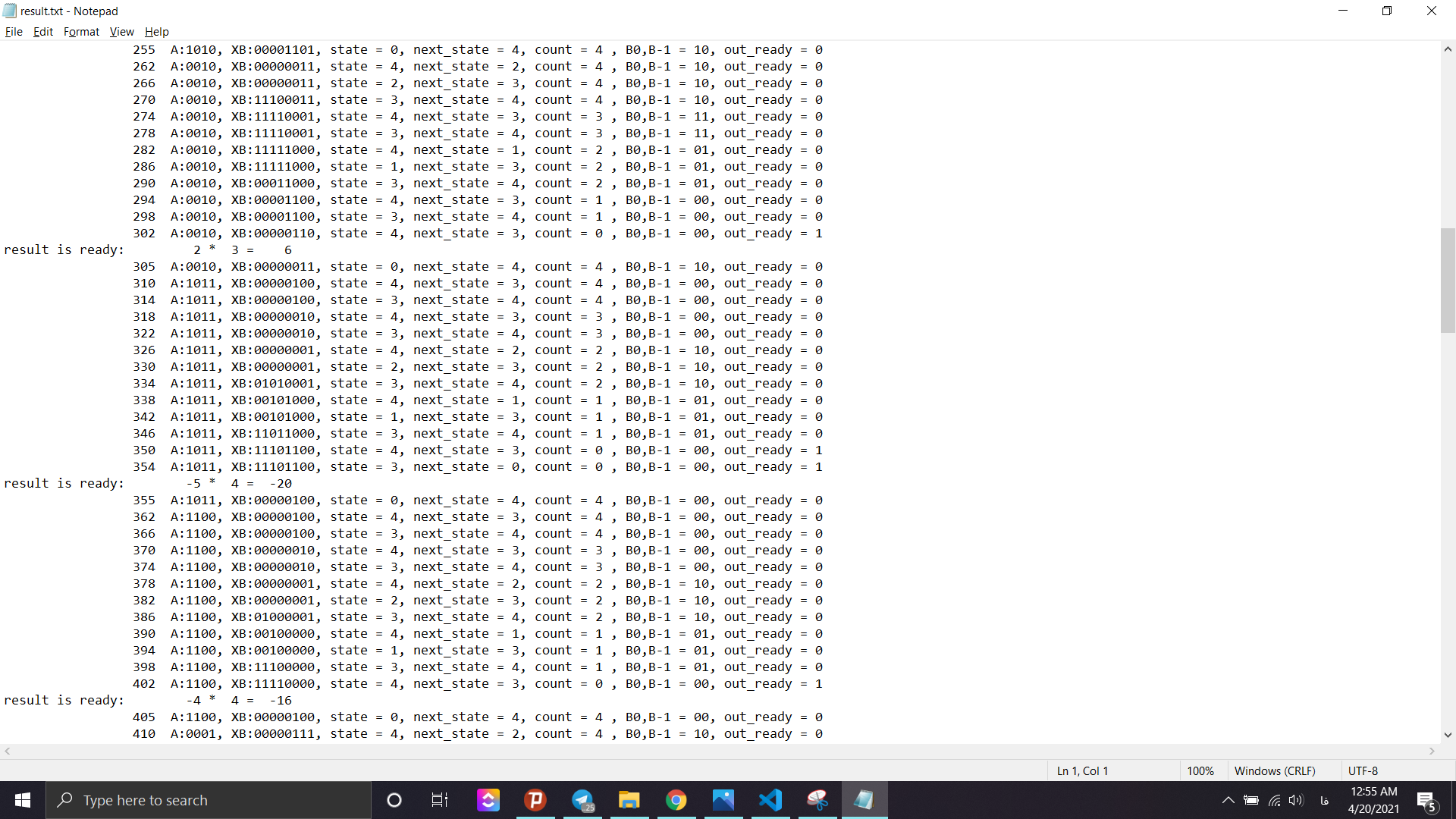
در فایل seed.dat یک عدد رندوم قرار داده ایم.

این عدد رندوم، همان random$ ماست.

برای اینکه با هر سیمولیت این رندوم بودن ما تفاوت کند، ما به کمک یک بش در فایل seed.dat عدد رندوم را نوشته ایم. (به علت محدودیت لینوکس اینکار را کرده ایم. در نسخه ویندوز برای مادلسیم $random flag وجود دارد.)

نتایج:

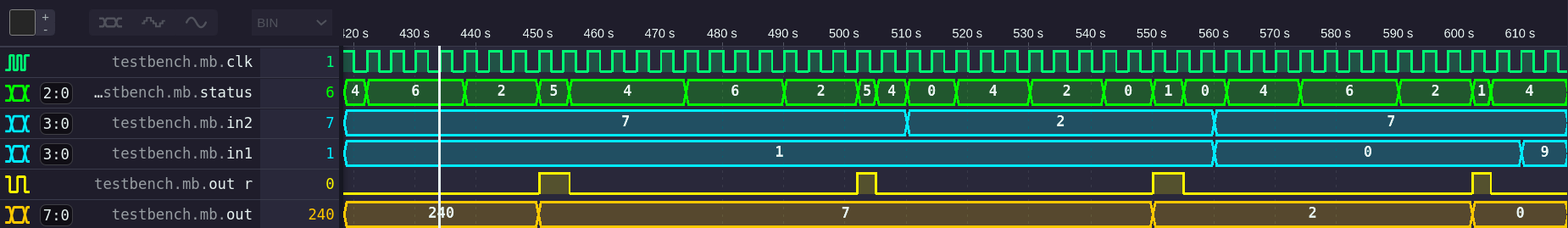




تعداد تست های گرفته شده زیاد است. تنها تصویر بعضی از آنها را در گزارش آورده ایم.

اطلاعات تکمیلی در فایل results.txt در پوشه اصلی موجود است.

نتایج Waveform:



**پایان**