گزارش کار میان ترم طراحی کامپیوتری سیستم های دیجیتال

پریا خوش‌تاب 810198387

توضیح کلی الگوریتم:

نحوه عملکرد پشته در این سوال را مشابه تمرین شماره 3 در نظر می‌گیریم پس ابتدا فرض می‌کنیم یک accumulator در datapath وجود دارد که هر بار به شرطm==0 or m==n برسد، مقدار آن یک واحد زیاد می‌شود و در نهایت خروجی تابع یعنی comb(m, n)، خروجی این accumulator می‌باشد.

در ابتدا ورودی های تابع یعنی m و n به ترتیب در رجیسترهای reg_m و reg_n ذخیره می-شوند و سپس در هر مرحله از الگوریتم به ترتیب اعداد m، n-1، m-1، n-1 را به عنوان پارامترهای comb(m, n-1) و comb(m-1, n-1) روی استک push می‌کنیم و سپس یک بار pop می‌کنیم و مقدار pop شده را در رجیستر reg_m ذخیره می‌کنیم و یک بار دیگر pop می‌کنیم و مقدار pop شده را در رجیستر reg_n ذخیره می‌کنیم. حال مقادیر ذخیره شده در این رجیسترها را برای شرط m == 0 or m == n بررسی می‌کنیم، در صورتی که این شرط برقرار باشد، مقدار accumulator یک واحد زیاد می‌شود و دوباره به همین ترتیب دو عدد دیگر pop می‌شود و الگوریتم ادامه پیدا می‌کند، اما در صورتی که این شرط برقرار نباشد به ابتدای الگوریتم می‌رویم و دوباره به همین ترتیب 4 پارامتر روی استک push می‌شوند

و الگوریتم ادامه پیدا می‌کند. این مراحل تا زمانی ادامه پیدا می‌کنند که استک خالی بشود. در واقع زمانی که سیگنال is_empty استک فعال شود، سیگنال done را یک می‌کنیم که در این حالت، خروجی مسئله در خروجی accumulator مشاهده می شود.

نکات طراحی:

- از آنجایی که n و m 4 بیتی می‌باشند و مقادیری که روی استک push می‌شوند، پارامترهای توابع می باشند، بنابراین رجیستر های reg_m و reg_n، تفریق کننده‌ها، استک، مالتی‌پلکسرها، مقایسه کننده ها باید 4 بیتی در نظر گرفته شوند، اما از آنجایی که n حداکثر 15 می‌باشد، حداکثر مقدار comb(m, n) مربوط به m = 7 یا m = 8 می‌باشد:

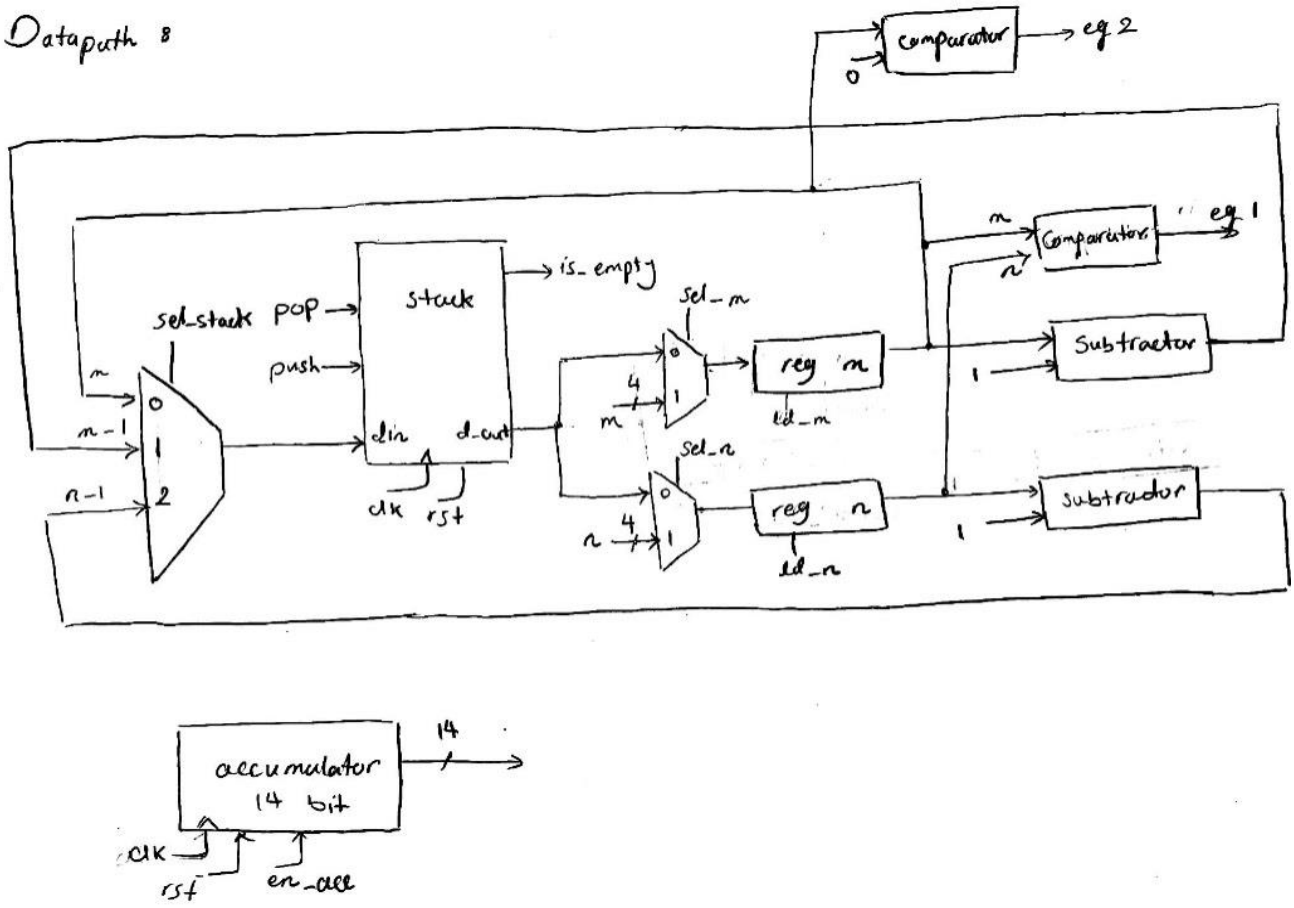
$$comb(7, 15) = comb(8, 15) = 6435 = 1100100100011 \rightarrow 13 \text{ bits}$$

بنابراین accumulator باید 13 بیتی در نظر گرفته شود.

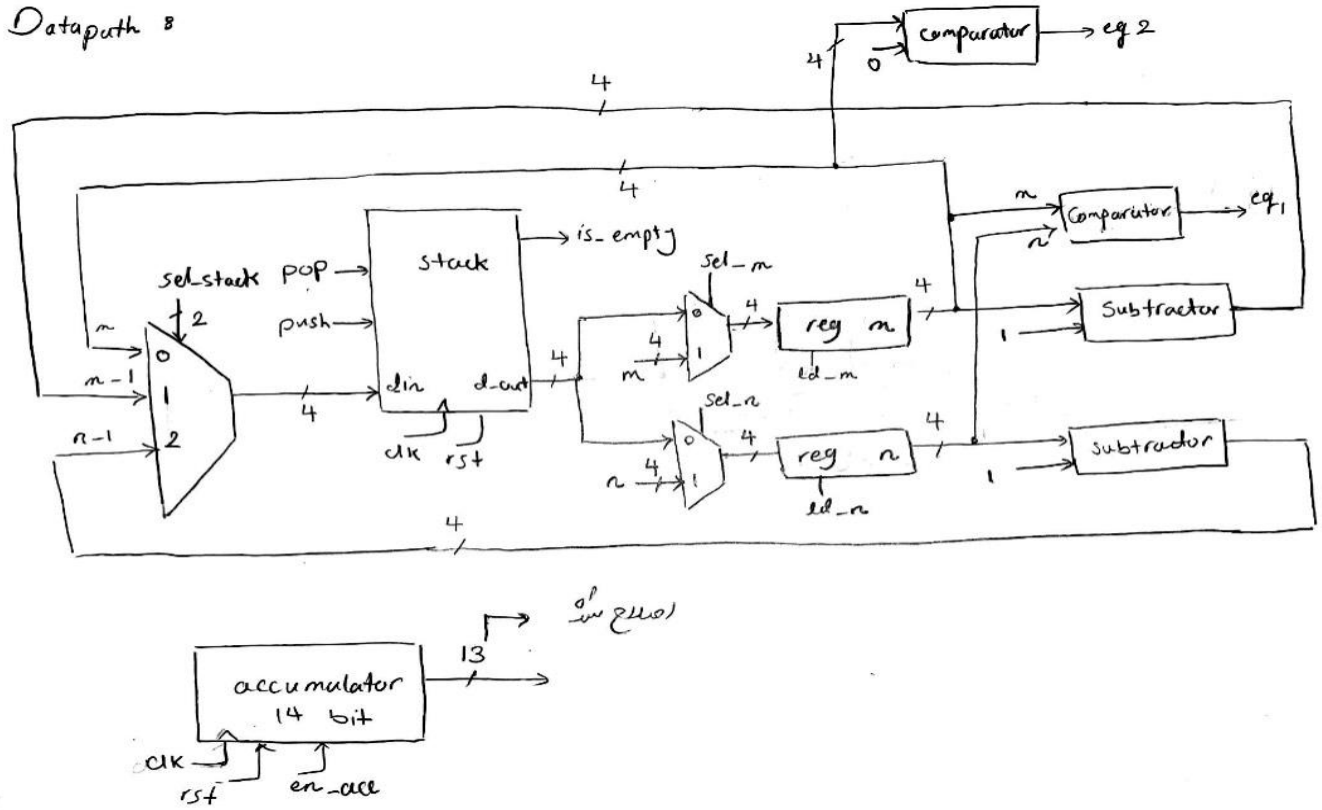
- تمامی سیگنال های کنترلی در هر مرحله از الگوریتم(اول always) به 0 ست می‌شوند.

:Datapath

❖ **datapath رسم شده در امتحان:**



❖ **datapath اصلاح شده در take-home:**

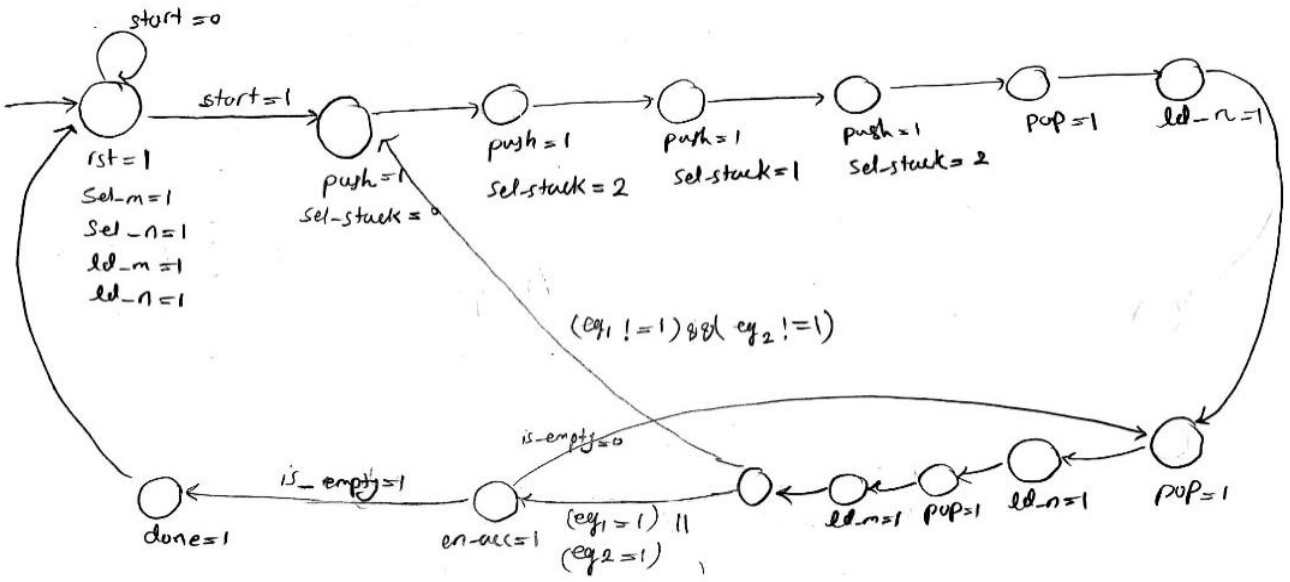


اصلاحات Datapath:

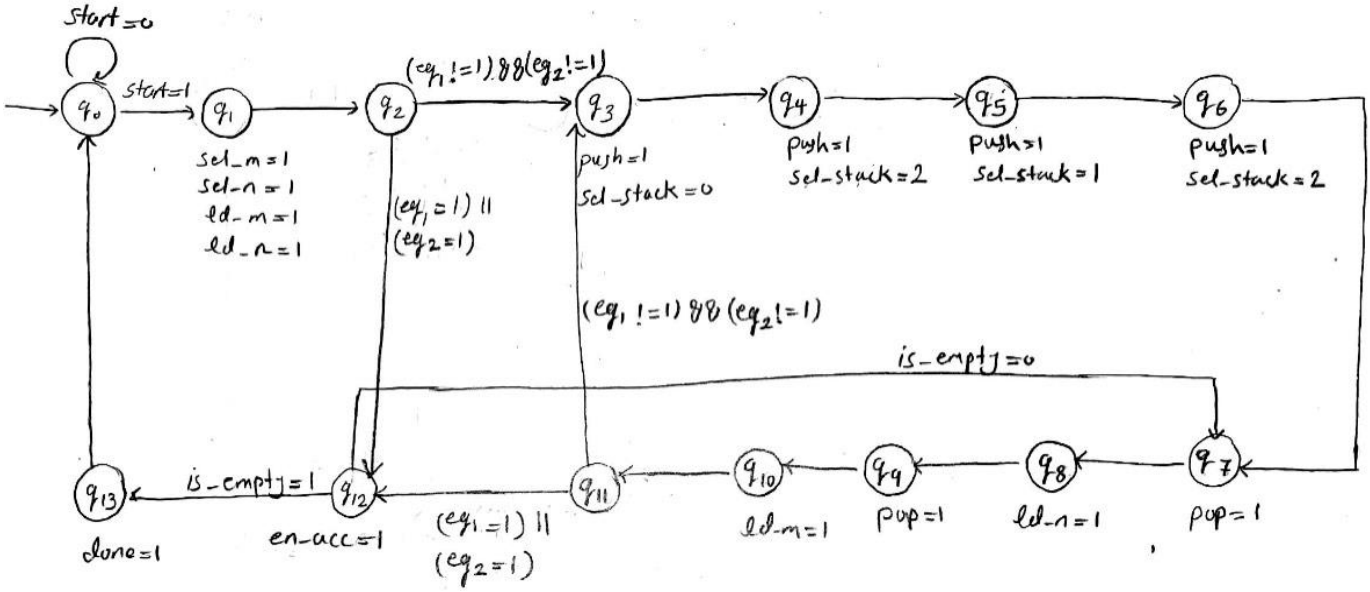
❖ همانطور که در شکل بالا مشخص شده تنها بخش اصلاح شده مربوط به تعداد بیت های خروجی accumulator می‌باشد، که در امتحان به اشتباه تعداد بیت های comb(16, 8) یعنی 14 بیت در نظر گرفته شده بود، اما همانطور که در بالا ذکر شد، مقدار صحیح تعداد بیت های آن مربوط به comb(7, 15) یا comb(8, 15) یعنی 13 بیت می‌باشد.

:Controller

❖ **controller رسم شده در امتحان:**



❖ **controller اصلاح شده در take-home:**

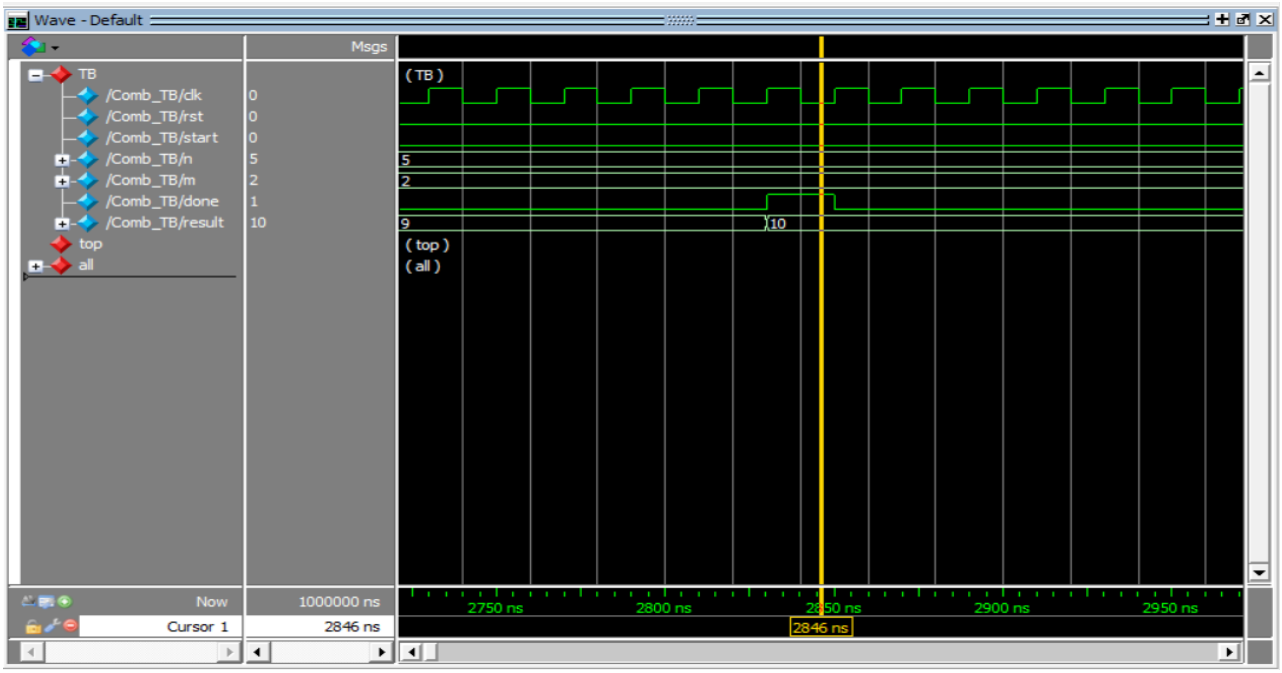


اصلاحات Controller:

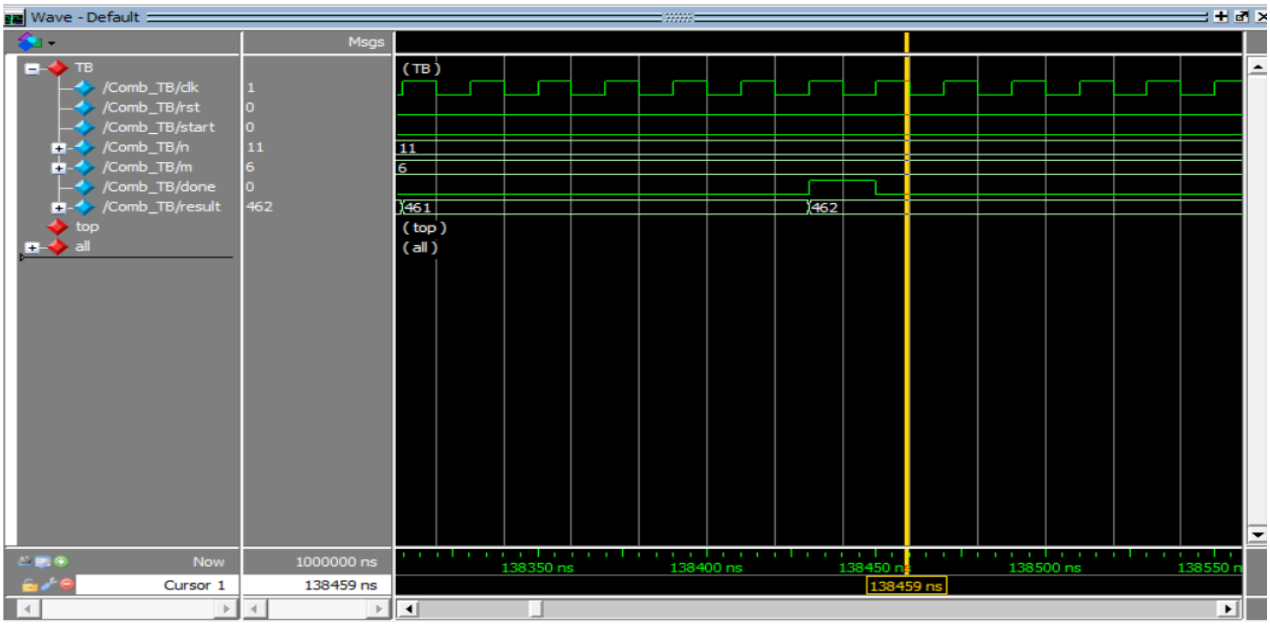
- به دلیل حواس پرتی در امتحان هنگام پاک نویس کردن، دو استیت هرکدام دو بار تکراری نوشته شده‌اند، گرچه در توضیحات اول امتحان، درست نوشتم که باید دوبار pop کنیم، اما در کنترلر حواس پرتی کردم. درواقع دو استیت q7 و q8 در کنترلر اصلاح شده، هرکدام دو بار در کنترلر امتحان تکرار شده اند، که در کنترلر اصلاح شده، این استیت های تکراری حذف شده‌اند.
- سیگنال rst که در کنترلر امتحان در استیت اول یک می‌شود را در طراحی اصلاح شده، پاک کردم زیرا عملیات مربوط به صفر و یک کردن این سیگنال را در تست بنچ انجام می‌دهم.
- یک استیت جدید به عنوان اولین استیت کنترلر (q0) در کنترلر اصلاح شده اضافه کردم، تا همان‌طور که در صورت سوال گفته شده پس از یک شدن start، رجیستر های داخلی لود بشوند، نه زمانی که start هنوز یک نشده است.
- یک استیت جدید (q2) به منظور چک کردن شرط m == n or m == 0 برای ورودی های اولیه n و m اضافه کردم. در این استیت در صورتی که شرط ذکر شده برقرار باشد به استیت q12 می‌رویم و بعد done یک می‌شود و کار تمام می‌شود، در غیر این صورت به استیت q3 می‌رویم و همان مراحل ذکر شده در بالا تکرار می‌شوند.

چند تست:

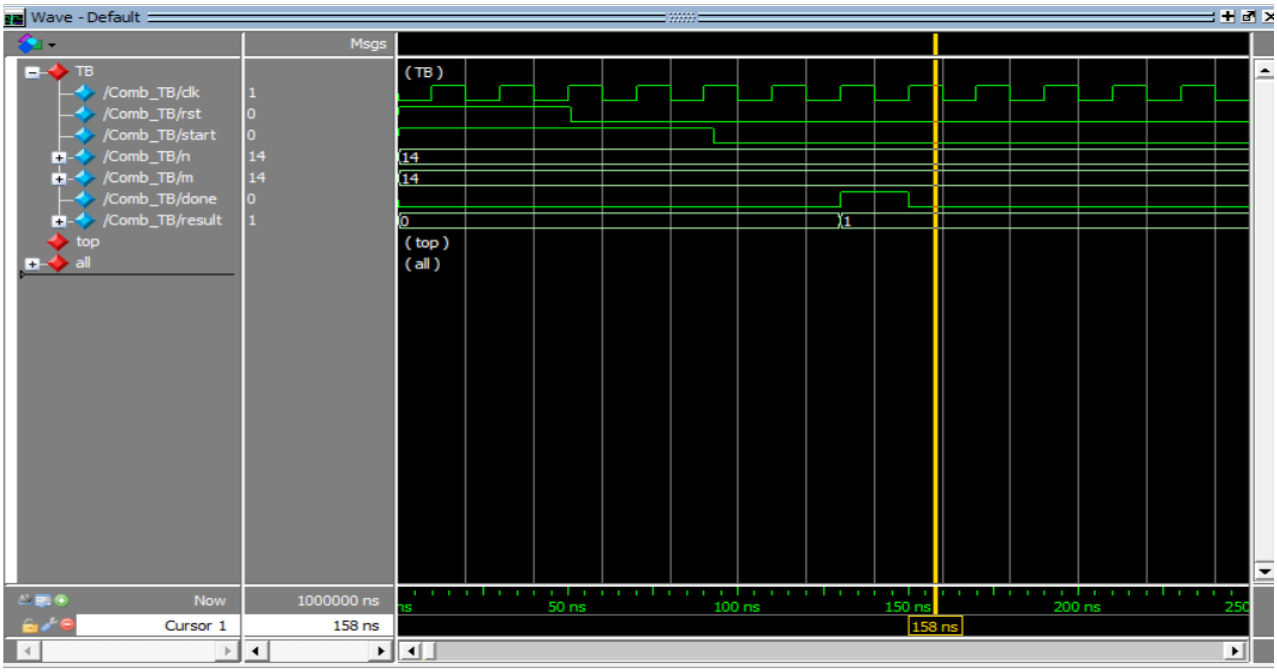
❖ $n = 5, m = 2 \rightarrow \binom{n}{m} = \binom{5}{2} = 10$



❖ $n = 11, m = 6 \rightarrow \binom{n}{m} = \binom{11}{6} = 462$



❖ $n = 14, m = 14 \rightarrow \binom{n}{m} = \binom{14}{14} = 1$



❖ $n = 15, m = 0 \rightarrow \binom{n}{m} = \binom{15}{0} = 1$