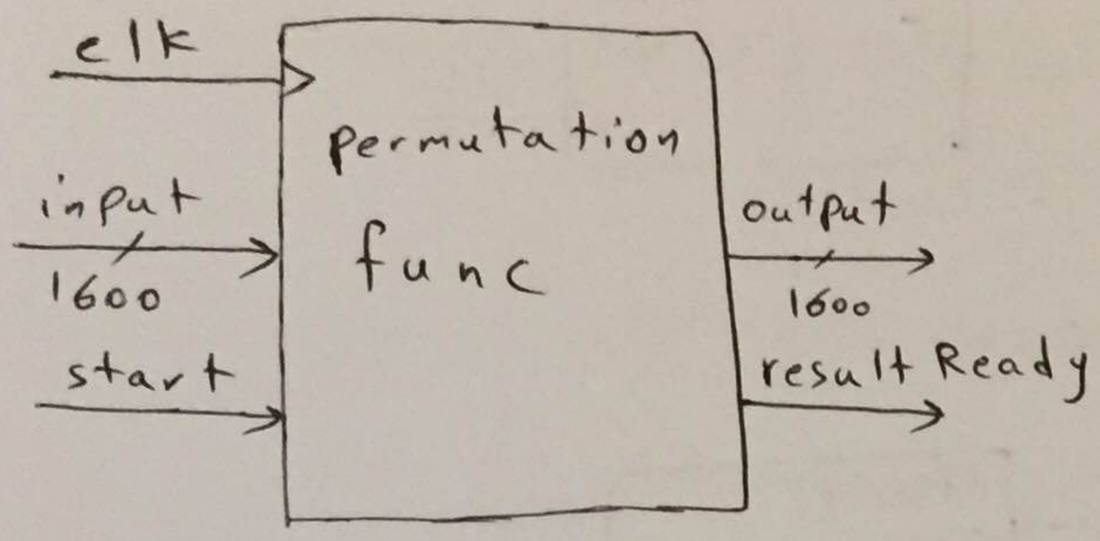
علی مهرانی ۸۱.۱۹۸۵۴۲

گزارش کار پروژه اول

فاز اول

طراحی کلی مدار:

مدار در بالاترین حالت (top_level module) یک ورودی ۱۶۰۰ بیتی و یک خروجی ۱۶۰۰ بیتی دارد و به شکل زیر است



تابع module ما همچنین یک clk به عنوان clock و یک سیگنال به نام start به عنوان ورودی می‌گیرد که در صورتی که start برابر ۱ بود کارش را انجام می‌دهد

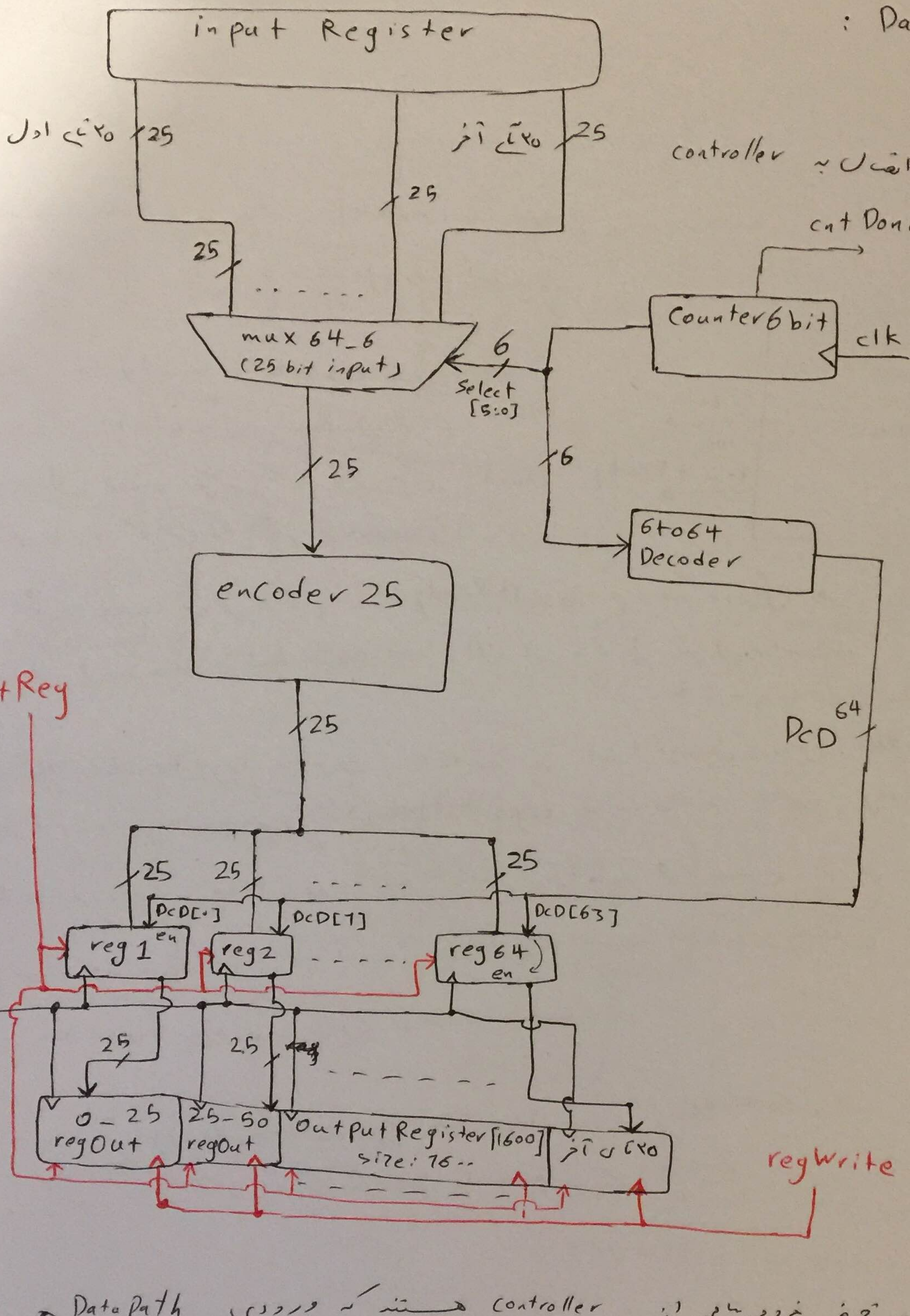
همچنین یک خروجی به نام result Ready می‌دهد که در اگر هر زمان خروجی نهایی آماده شد (در فایل نوشته شد) مقدار اگ از ۰ به ۱ تبدیل می‌شود

در testbench بالاترین module نیز ابتدا از فایل ورودی اطلاعات به صورت یک memory به فرم reg [24:0] mem[0:63] خوانده می‌شود و این memory به شکل یک ورودی ۱۶۰۰ تایی در حد آرایه درمی‌آید و، سپس خروجی ی سه می‌شود و خروجی - در قالب حد آرایه ۱۶۰۰ تایی است دوباره به شکل memory درشده در بالا درمی‌آید و در فایل نوشته می‌شود

مدار ما (Permutation_func) به طور کلی از یک Controller و یک datapath تشکیل شده که در ادامه هرکدام را توضیح می‌دهیم

ل کلی مدار :

شکل Datapath :



سیگنال های خروجی های : controller هستند که ورودی Datapath هستند

☆ نکته : در طراحی مدار با نوشتن کد verilog برای datapath و controller ، بخش counter 6 bit را می توانیم جزء controller در نظر بگیریم

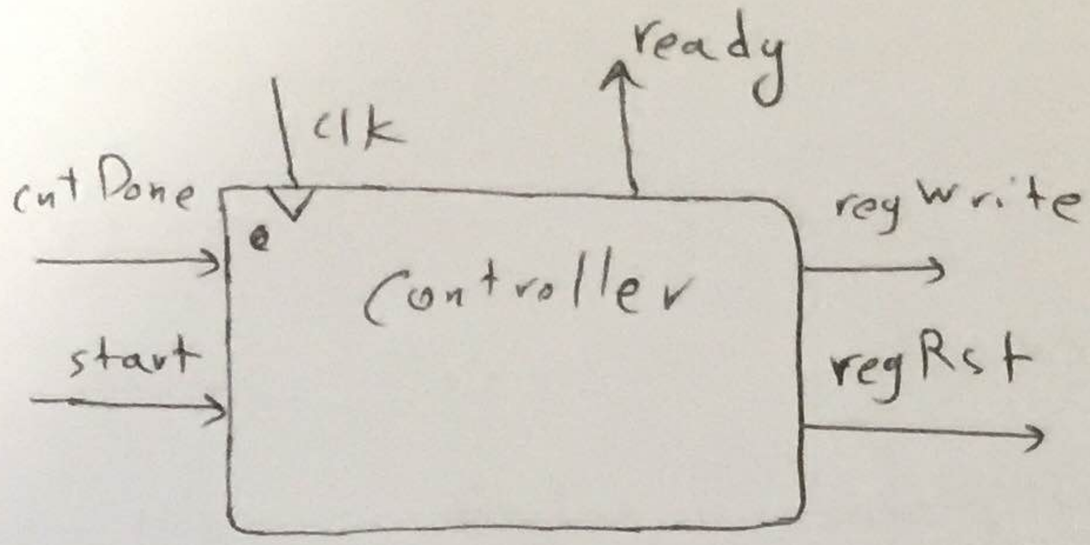
توضیحات DataPath :

ورودی به عنوان یک DataPath داده می‌شود و این ورودی بعضو ۲۵تایی
۲۵ تا جدا می‌شود و ۶۴ ورودی یک multiplexer می‌شوند
همزمان با این کار، counter نیز شروع به کار می‌کند و مقدار خروجی 6bit آن
به عنوان select به multiplexer داده می‌شود، بدین ترتیب در هر لحظه
زمان (یا clock) یک ۲۵ بیتی از ورودی جدا می‌شود و خوانده می‌شود (در لحظه اول
۲۵تای اول سپس ۲۵تای دوم و ... تا ۲۵تای آخر)
خروجی multiplexer سپس به encoder25 داده می‌شود که وظیفه آن رمز
نگاری ورودی ۲۵تایی طبق الگوریتم داده شده است)
همزمان با این کار خروجی counter به decoder داده می‌شود و خروجی ۶۴
بیتی decoder به نام DcD گرفته می‌شود
خروجی encoder25 پس از ایجاد شدن به ۶۴ register ۲۵ بیتی به عنوان
ورودی داده می‌شود و بیت های ۰ تا ۶۳ DcD نیز یک به یک به عنوان enable
برای register به عنوان ورودی داده می‌شوند که بدین ترتیب در هر لحظه فقط خروجی
encoder25 در یکی از register ها نوشته می‌شود و مقدار بقیه ثابت می‌ماند
در نهایت این فرآیند ۶۴ بار انجام می‌شود تا تمامی ۲۵ بیتی ها به ترتیب
رمزنگاری شوند و در خروجی نوشته شوند
counter6bit که خروجی cntDone است
وقتی این کار ۶۴ بار انجام شد سیگنال 1 می‌شود و به controller فرستاده می‌شود و controller نیز ready را 1
می‌کند تا اعلام شود که نتیجه محاسبه شده است
(در این حالت در هر لحظه فقط یک خط (یک ۲۵ بیتی) محاسبه می‌شود)

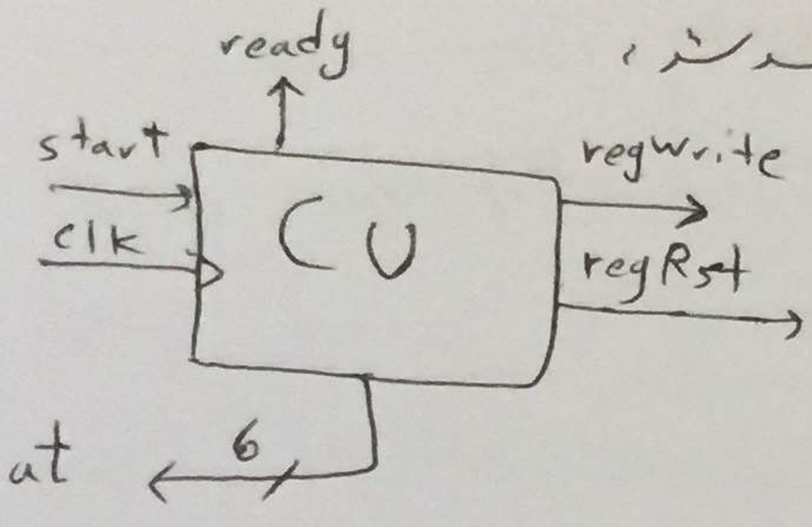
٤/

: Controller ف

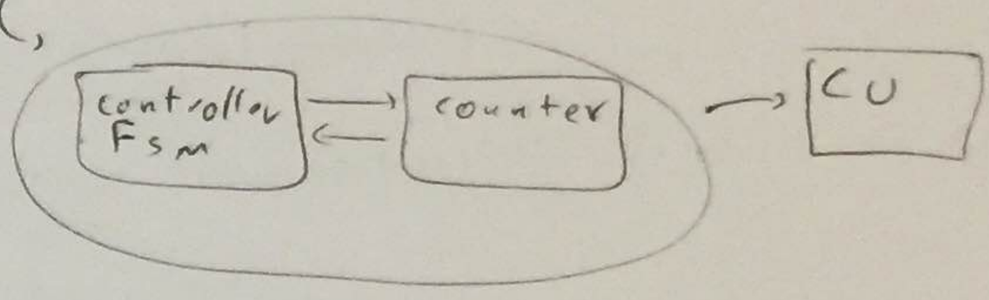
شکل واحد controller به صورت زیر است :



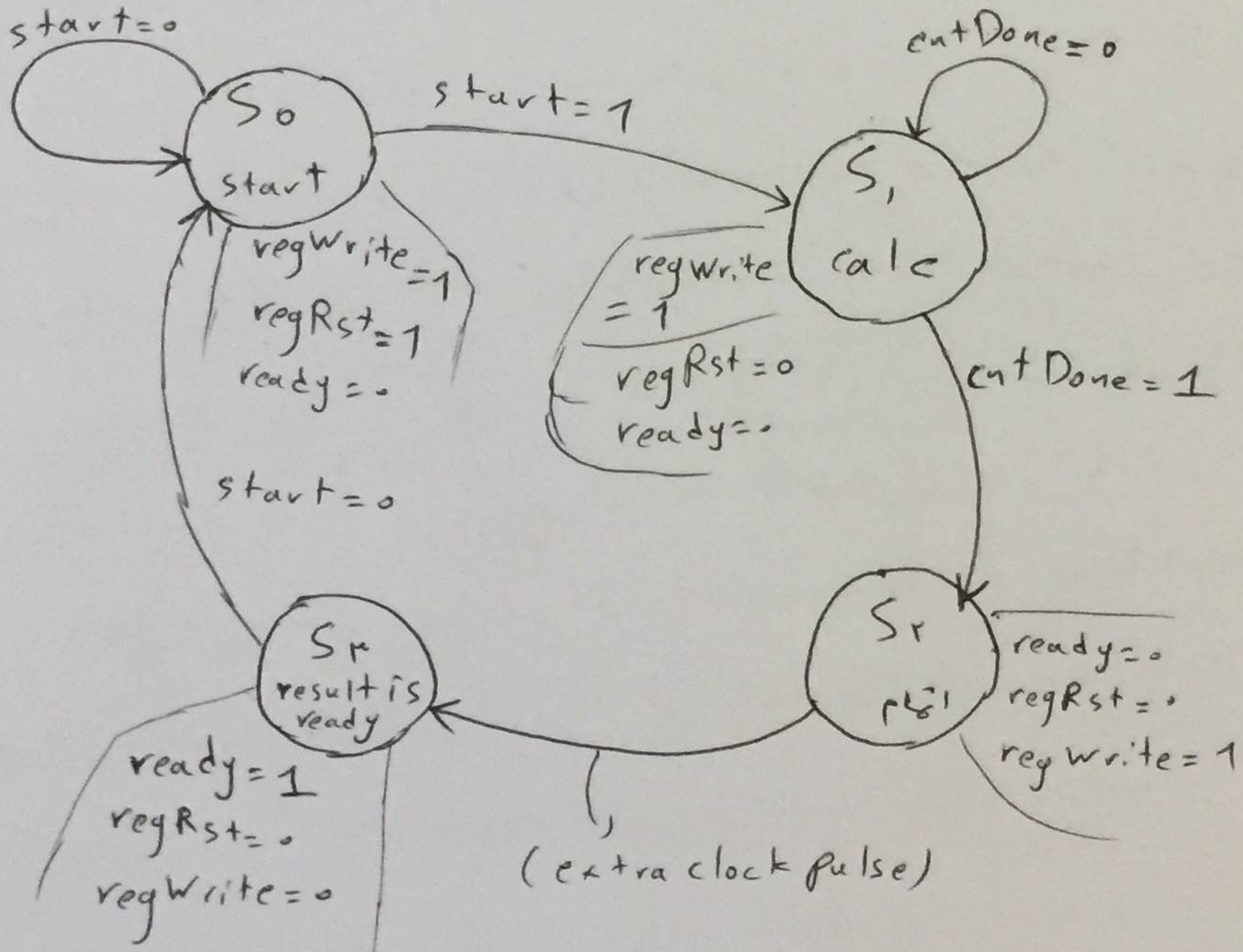
* همانطور - قبلاً گفته شد برای طراحی verilog می‌توانیم Counter6 را بخش از controller در نظر بگیریم که در اینصورت controller به این صورت خواهد شد ،



و FSM ی زیرمجموعه از CU می‌شود (



FSM :



توضیحات FSM:

از S0 که state شروع است کار را آغاز می‌کنیم و تا زمانی که مقدار start برابر ۱ است کاری نمی‌کنیم در غیر اینصورت وارد S1 می‌شویم که در آن در هر clk یک بیت را رمزنگاری می‌کنیم و تا زمانی که ۲۵ بار این کار انجام شده ادامه می‌دهیم وقتی این کار انجام شد به S2 و سپس به S3 می‌رویم

در تمامی state ها به جز آخر (S3) مجاز به نوشتن در register ها هستیم ⇐ regWrite

فقط در state آخر (S3) مقدار ready برابر ۱ می‌شود ⇐ ready

زمانی که می‌خواهیم کار را انجام دهیم و در state های دیگر مقدار rst در register ها باید ۰ باشد ⇐ regRst

☆ نحوه کار encoder

طبق الگوریتم رابطه اندیس ها به این صورت است

۴	۳	۲	۱	۰
۹	۸	۷	۶	۵
۱۴	۱۳	۱۲	۱۱	۱۰
۱۹	۱۸	۱۷	۱۶	۱۵
۲۴	۲۳	۲۲	۲۱	۲۰

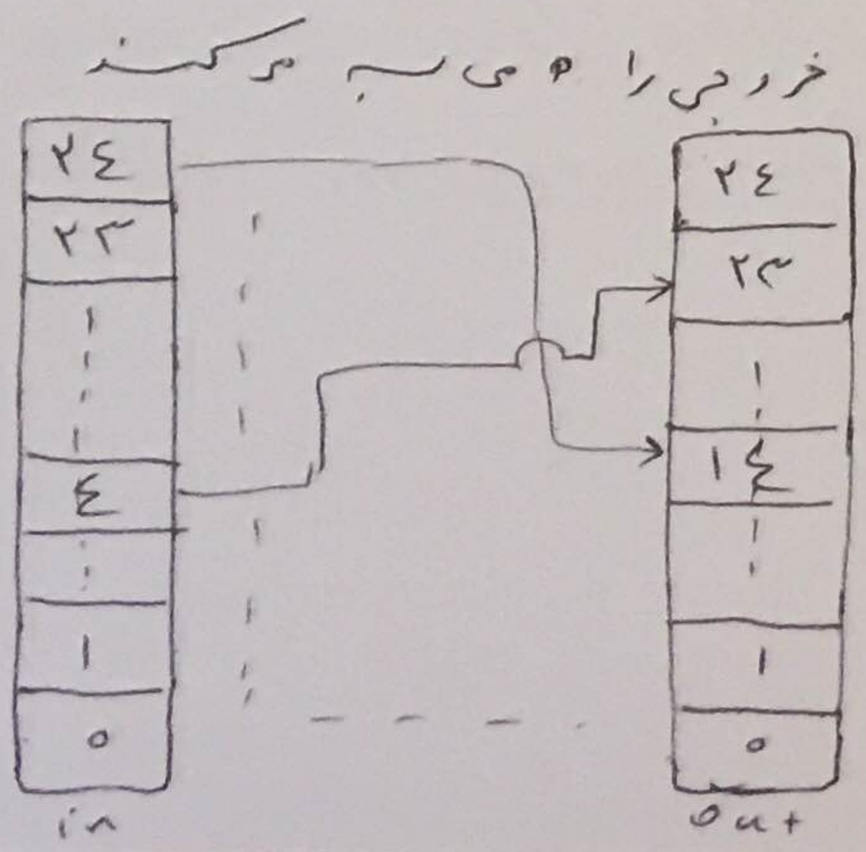
in

⟹

۲۳	۱۷	۱۱	۵	۴
۲۱	۱۵	۱۴	۸	۲
۲۴	۱۸	۱۲	۶	۰
۲۲	۱۶	۱۰	۹	۳
۲۰	۱۹	۱۳	۷	۱

out

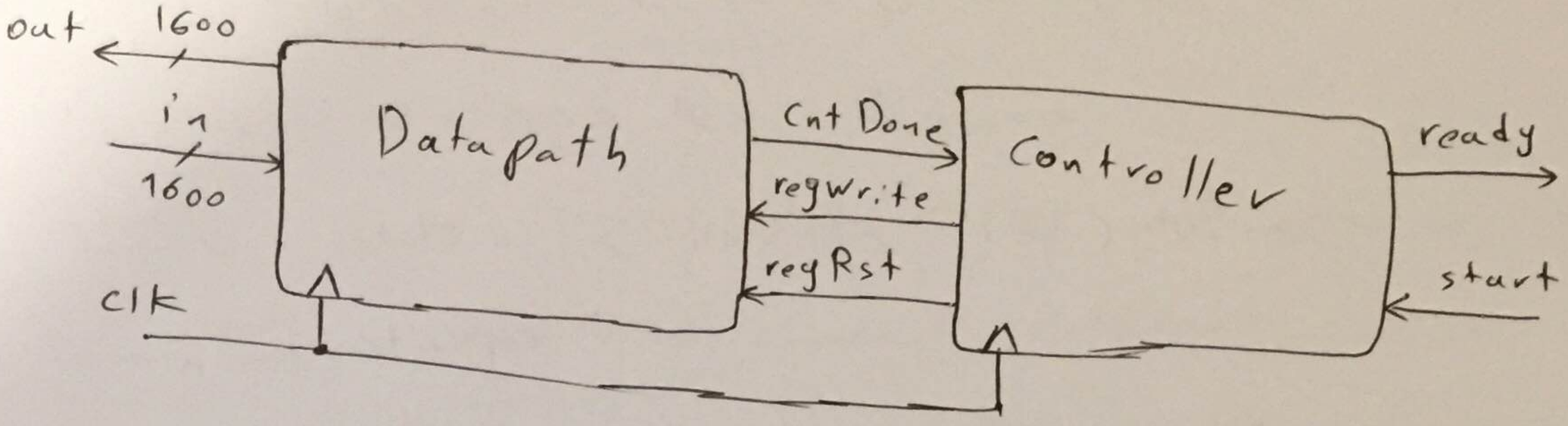
واحد encode25 به طبق رابطه بالا صرفاً مقدیر خروجی را map می‌کند



تغییرات و اشکالات اول فاز اول:

★ regRst به عنوان یک خروجی به controller اضافه شد

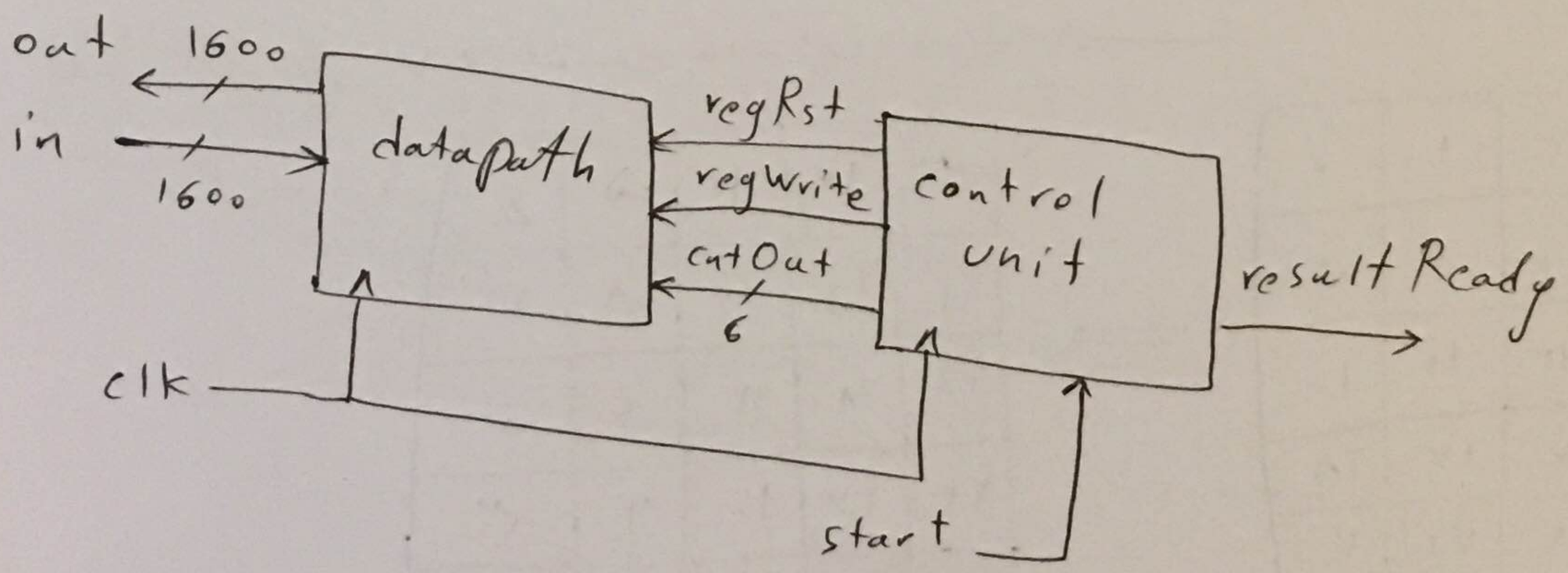
★ فرم کلی اتصال datapath و controller



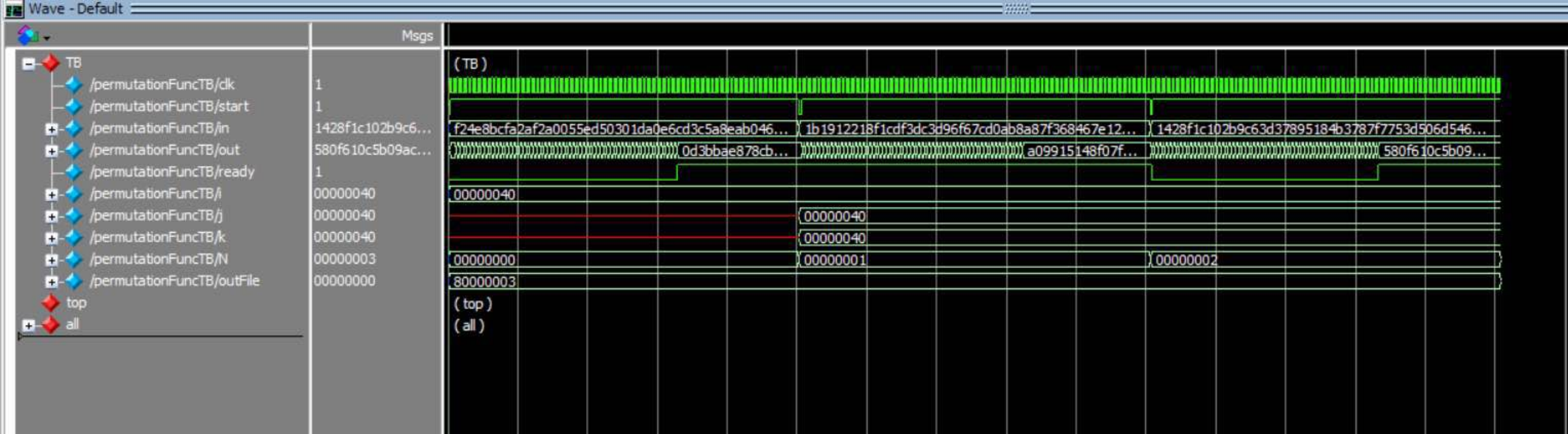
تغییر: در بخش پیاده سازی Controller FSM و counter6bit در verilog

را در یک module کلی با نام Controlunit نوشتم و از آن به عنوان

این صورت در verilog در data path و controller

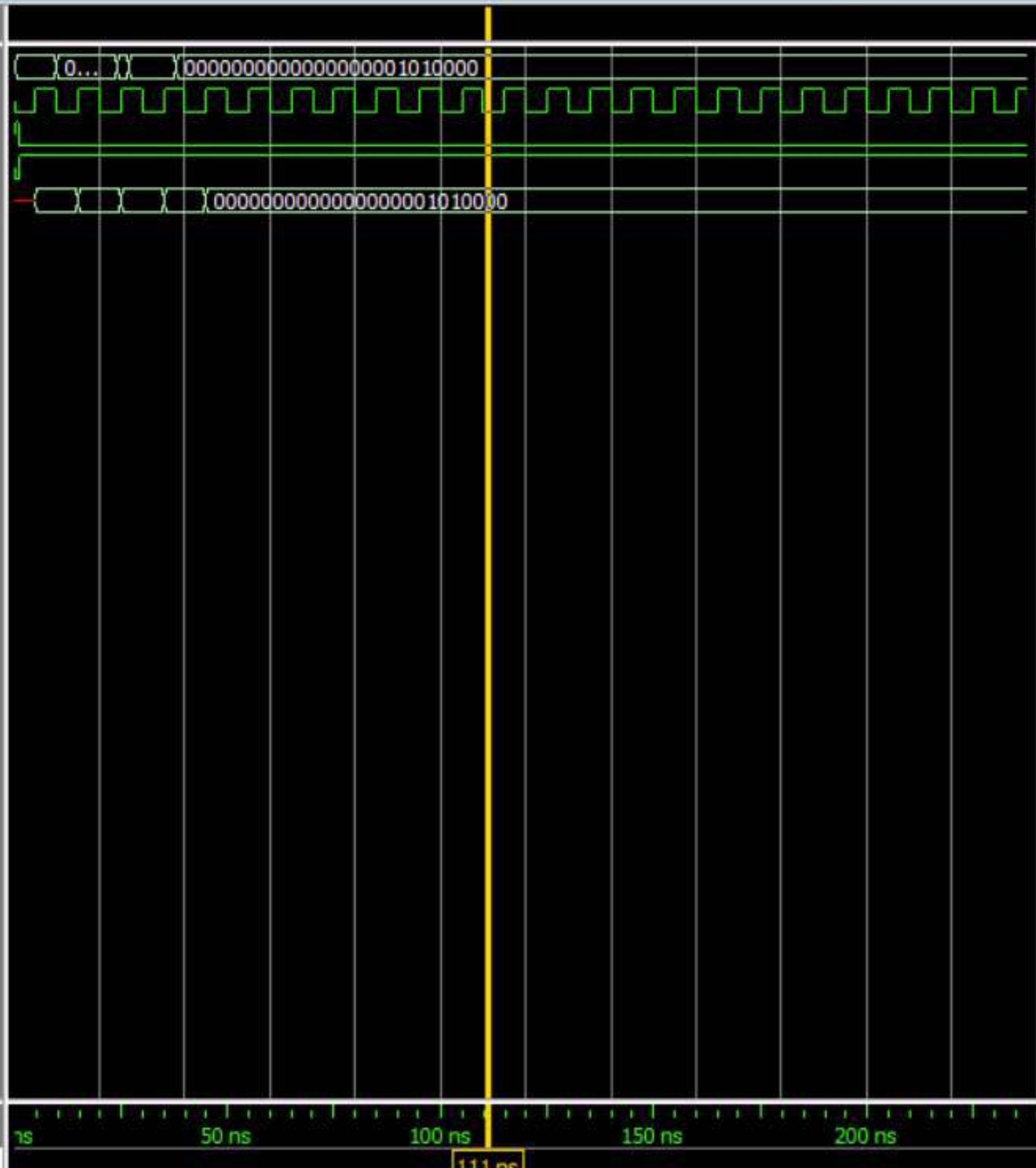


★ نکته: همین می توان به طور کلی از سیگنالی به نام regWrite استفاده نکرد و با همان en برای register ها کار را انجام داد

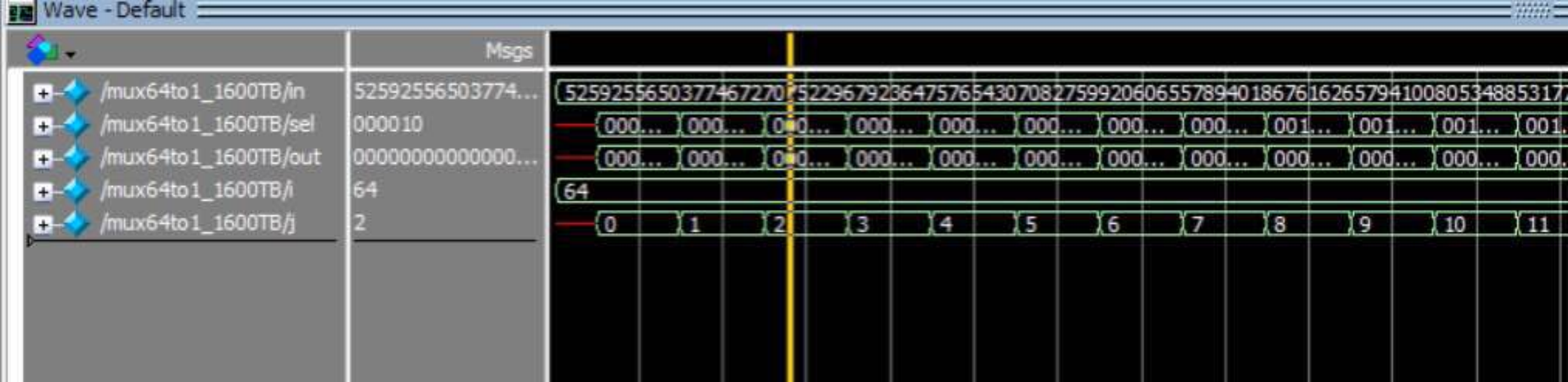

Wave - Default
Msgs
TB
/permutationFuncTB/clk
/permutationFuncTB/start
/permutationFuncTB/in
/permutationFuncTB/out
/permutationFuncTB/ready
/permutationFuncTB/i
/permutationFuncTB/j
/permutationFuncTB/k
/permutationFuncTB/N
/permutationFuncTB/outFile
top
all
1
1
1428f1c102b9c6...
580f610c5b09ac...
1
00000040
00000040
00000040
00000003
00000000
(TB)
f24e8bcfa2af2a0055ed50301da0e6cd3c5a8eab046...
1b1912218f1cdf3dc3d96f67cd0ab8a87f368467e12...
1428f1c102b9c63d37895184b3787f7753d506d546...
0d3bbae878cb...
a09915148f07f...
580f610c5b09...
00000040
00000040
00000040
00000000
00000001
00000002
80000003
(top)
(all)

Wave - Default

	Msgs
/register25bitTB/in	00000000000000000010 10000
/register25bitTB/clk	0
/register25bitTB/rst	0
/register25bitTB/en...	1
/register25bitTB/out	00000000000000000010 10000

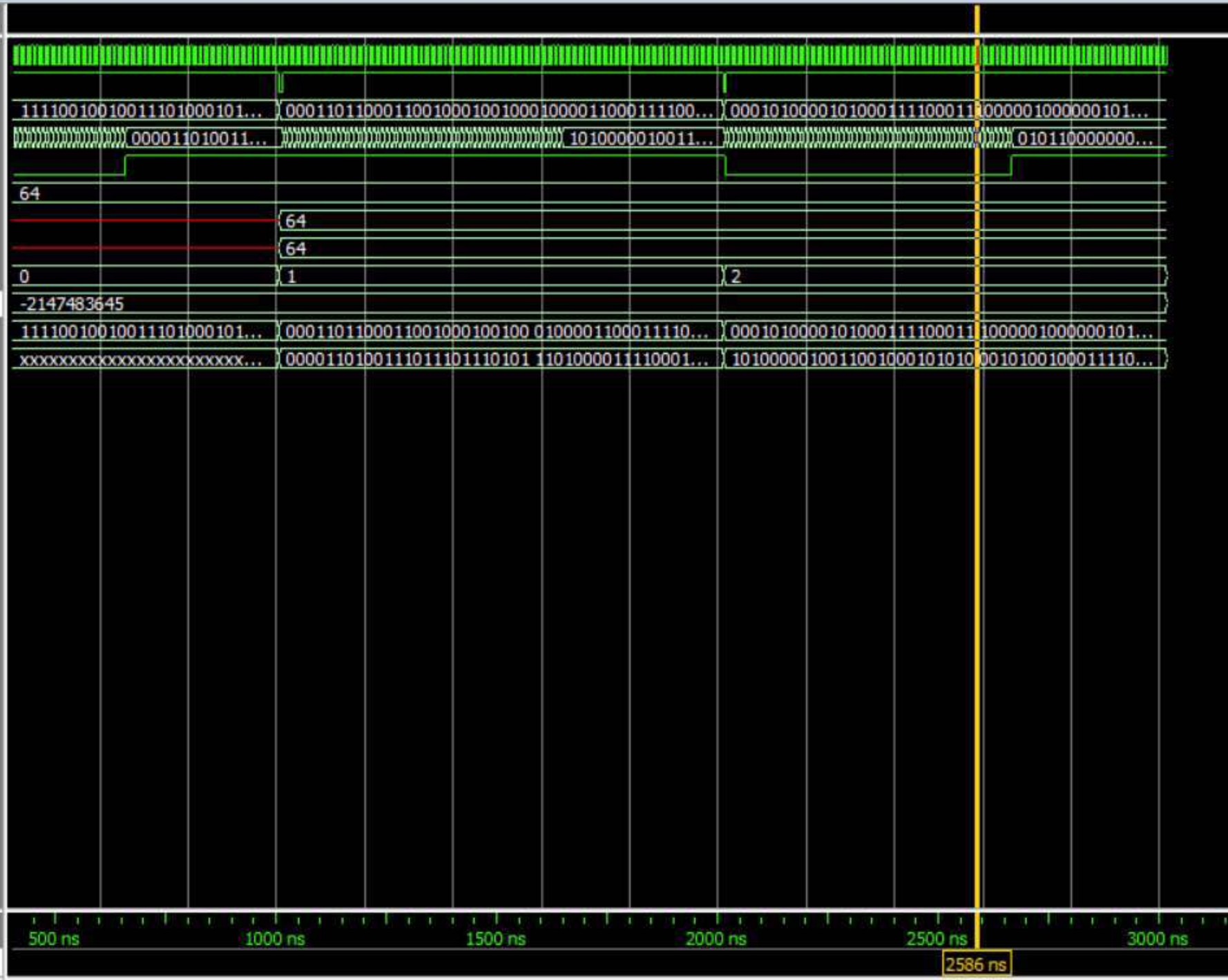


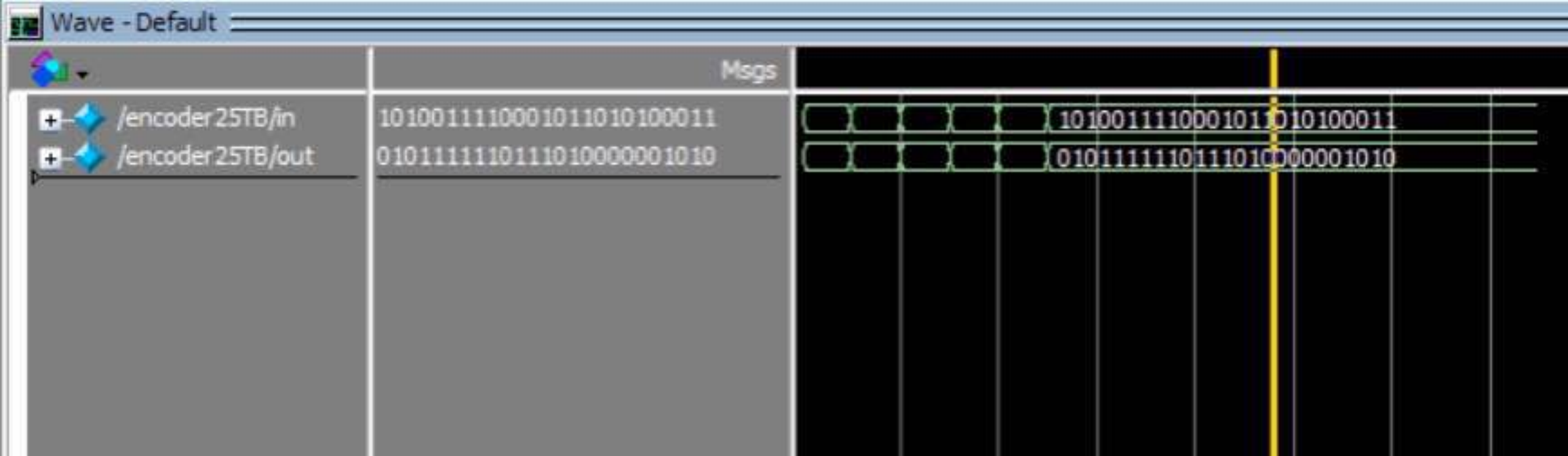
Now	238 ns
Cursor 1	111 ns

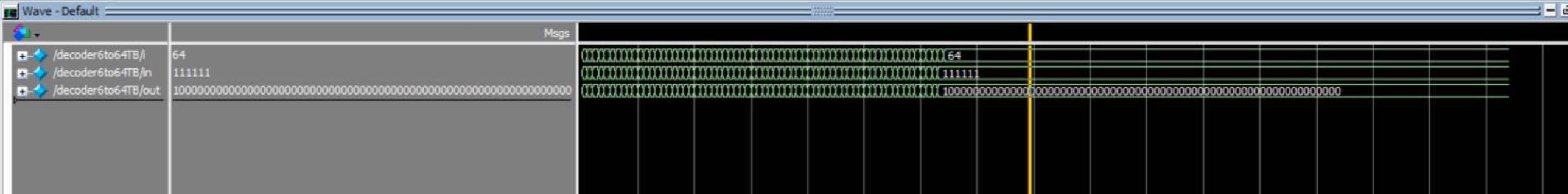

Wave - Default
Msgs
/mux64to1_1600TB/in
525925565037746727075229679236475765430708275992060655789401867616265794100805348853177
/mux64to1_1600TB/sel
000010
/mux64to1_1600TB/out
00000000000000...
/mux64to1_1600TB/i
64
/mux64to1_1600TB/j
2
64
0
1
2
3
4
5
6
7
8
9
10
11

Wave - Default

	Msgs
/permutationFuncTB/clk	1
/permutationFuncTB/start	1
/permutationFuncTB/in	000101000010 10...
/permutationFuncTB/out	000000000000 00...
/permutationFuncTB/ready	St0
/permutationFuncTB/i	64
/permutationFuncTB/j	64
/permutationFuncTB/k	64
/permutationFuncTB/N	2
/permutationFuncTB/outFile	-2147483645
/permutationFuncTB/fileInMem	000101000010 10...
/permutationFuncTB/fileOutMem	101000001001 10...




Wave - Default
Msgs
/encoder25TB/in
/encoder25TB/out
10100111100010110101000011
01011111101110100000001010


Wave - Default
Msgs
/decoder6to64TB/i
/decoder6to64TB/in
/decoder6to64TB/out
64
111111
10000000000000000000000000000000000000000000000000000000000000000
64
111111
10000000000000000000000000000000000000000000000000000000000000000

	Msgs	
/counter6bitTB/clk_	1	
/counter6bitTB/rst_	0	
/counter6bitTB/out_	111111	
/counter6bitTB/cnt...	St1	
Now	662 ns	ns 100 ns 200 ns 300 ns 400 ns 500 ns 600 ns
Cursor 1	635 ns	635 ns