پروژه نهایی درس معماری کامپیوتر

پیاده سازی پردازنده چند سیکل به صورت micro program

گروه **17**

اعضای گروه : مهدی مهدوی هزاوه – حسین طاهر آموز

شرح کلی **پروژه** :

به طور کلی دو روش اصلی برای طراحی پردازنده چند سیکل وجود دارد :

1) طراحی مبتنی بر ماشین حالت (FSM)
2) طراحی مبتنی بر ریز برنامه (micro-program)

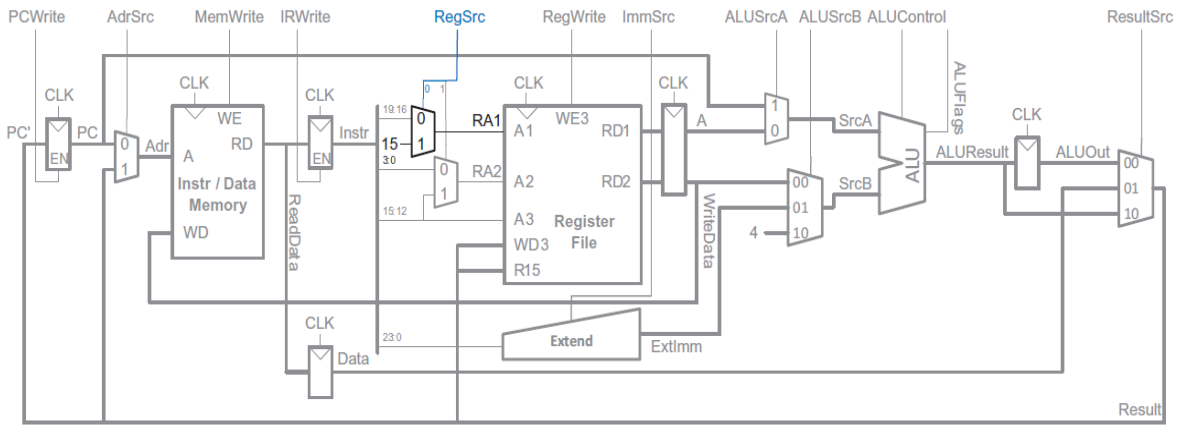
- در روش اول که پیاده سازی مبتنی بر ماشین حالت است ، بخش Main decoder پردازنده یک ماشین حالت است که بر اساس نوع دستور در حال اجرا و تعداد ریز دستورالعمل ها ، سیگنال های کنترلی مربوط به مسیر داده را تولید میکند.

- در روش دوم که امروزه کاربرد چندانی ندارد (منبع این حرف سایت www.geeksforgeeks.org)

برنامه ما مبتنی بر ریزبرنامه (micro-program) است. در این روش با درنظر گرفتن تمام حالت های ممکنی که واحد کنترل میتواند داشته باشد ، سیگنال های مربوط به هر حالت به صورت جدا و با منطق ترکیبی تولید میشود.(تمام سیگنال های هر حالت مستقل از حالت های دیگر است پس میتوان سیگنال های هر حالت را به صورت control word در یه حافظه ROM ذخیره کرد)

پیاده سازی **مسیر داده** :

مسیر داده پردازنده چند سیکل تقریبا مشابه ، مسیر داده پردازنده تک سیکل است ، در پردازنده تک سیکل ما هر دستور را در یک کلاک پردازش میکنم اما در پردازنده چند سیکل ، دستورات به چند قسمت (در این پیاده سازی به 5) تقسیم میشوند که برای این تقسیم بندی علاوه به مسیر داده پردازنده تک سیکل از تعدادی ریجستر غیر معماری هم استفاده میکنیم.



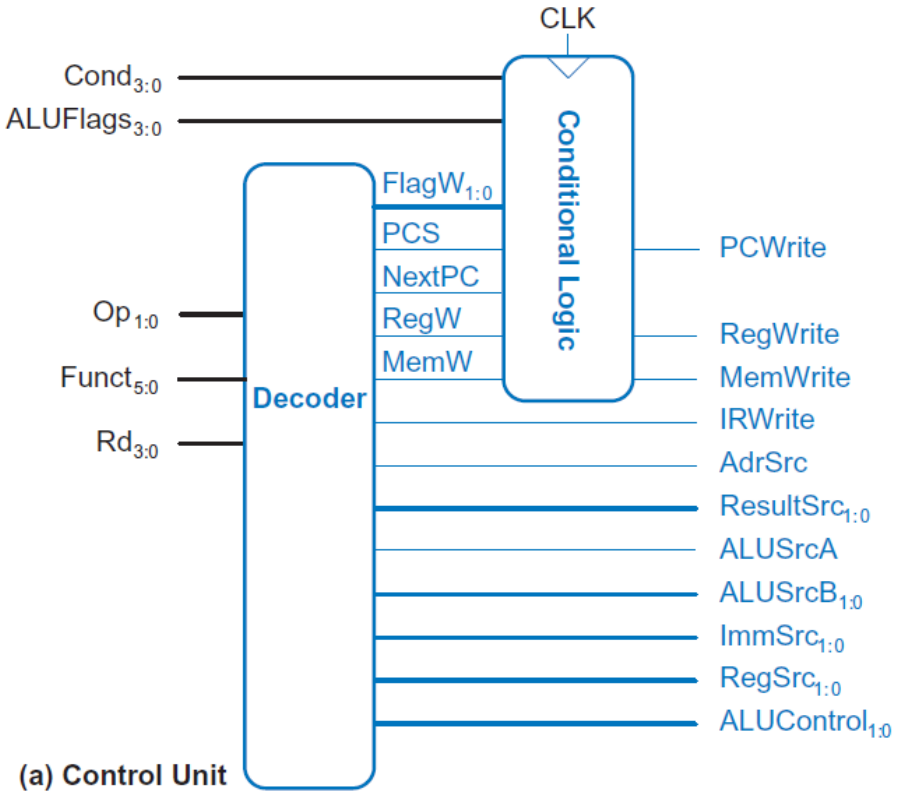
با توجه به مطالب ذکر شده و تصویر بالا ، پیاده سازی مسیر داده پردازنده چند سیکل بسیار راحت است . ماژول های استفاده شده برای پیاده سازی مسیره داده :

- ✓ ماژول regfile برای Register File
- ✓ ماژول alu برای Arithmatic logic unit
- ✓ ماژول extend برای پیاده سازی قسمت Extend Immediate
- ✓ ماژول mux3 , mux2 برای پیاده سازی Multiplexer
- ✓ ماژول های floper , flopner هم برای ریجستر های غیر معماری مورد استفاده قرار گرفتند

نمونه های ساخته شده از ماژول های floper , flopner :

- ✓ Pcreg برای نگهداری مقدار Program Counter (PCWrite enable)
- ✓ Instrreg برای نگهداری دستورات (IRWrite enable)
- ✓ Datareg برای نگهداری مقدار خوانده شده از حافظه در دستور LDR
- ✓ Areg برای خروجی اول Register File
- ✓ Readdatareg برای خروجی دوم Register File
- ✓ Aluoutreg برای نگهداری خروجی واحد Arithmatic Logic Unit

پیاده سازی بخش **کنترل** :



هر دستور از تعدادی ریزدستور تشکیل شده است که مجموعه این ریزدستور ها را ریزبرنامه یا firmware می‌گویند که در واقع بخش میانی بین سخت افزار و نرم‌افزار است.

برای طراحی بخش کنترل در پردازنده ریزبرنامه ، باید به ساختار دستورات توجه کرد. هر دستور از تعدادی ریزدستور(micro-instruction) تشکیل شده است که برای اجرای هر کدام از آنها باید یکسری سیگنالهای کنترلی تولید شود. این سیگنال های کنترلی یا فعال هستند یا غیر فعال پس می‌توان آنها را با یک بیت نمایش داد. مجموع این سیگنال‌ها کلمه کنترلی(control word) را تولید می‌کنند، پس هر ریزدستور یک کلمه کنترلی مختص به خود دارد. این کلمه های کنترلی در

حافظه ذخیره شده و متناسب با دستور، آدرس کلمه کنترلی بعدی را هم به انتهای کلمه کنترلی اضافه می‌کنیم.

کلمه کنترلی ما در اینجا افقی (Horizontal) است یعنی متشکل از سیگنال های کنترلی داخلی پردازنده،سیگنال‌های کنترلی باس سیستم، سیگنال مخصوص شرایط اجرا دستور branch و در انتها هم آدرس کلمه کنترلی بعدی آمده است.

اگر بیت مربوط به branch یا jump باشد ،کلمه ها به صوت ترتیبی اجرا می‌شوند در غیر این صورت کلمه کنترلی که آدرس آن در انتهای این کلمه قرار دارد، اجرا می‌شود.

اجزای اصلی بخش کنترلی در شکل نشان داده شده است.control memory ریزدستورات را در خود ذخیره می‌کند. CAR آدرس ریزدستورات بعدی را دارد و از روی حافظه ،کلمه کنترلی خوانده می‌شود و در CBR قرار می‌گیرد. حال از این کلمه کنترلی بدست آمده می‌توان سیگنال های کنترلی را استخراج کرد و از طریق سیگنال های کنترلی به مسیر داده فرستاد.

وظیفه بخش sequencelogic این است که بر اساس aluflags و آدرس کلمه بعدی که از CBR می‌آید ،آدرس جدید را در CAR قرار دهد و دستور read را بدهد.

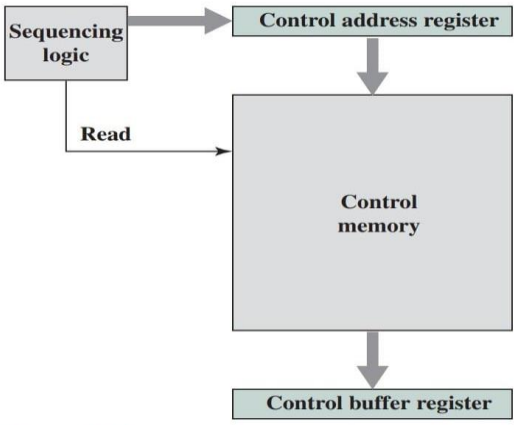
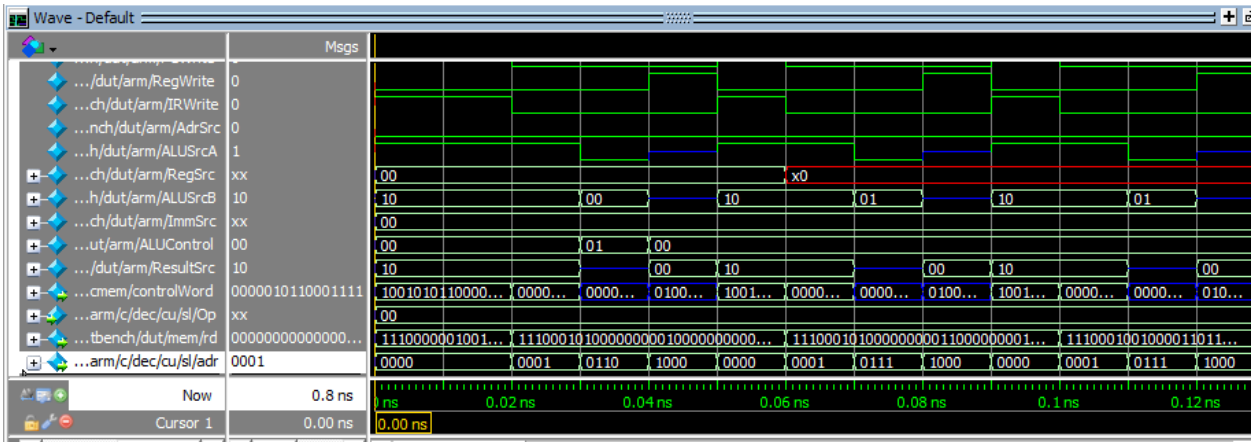
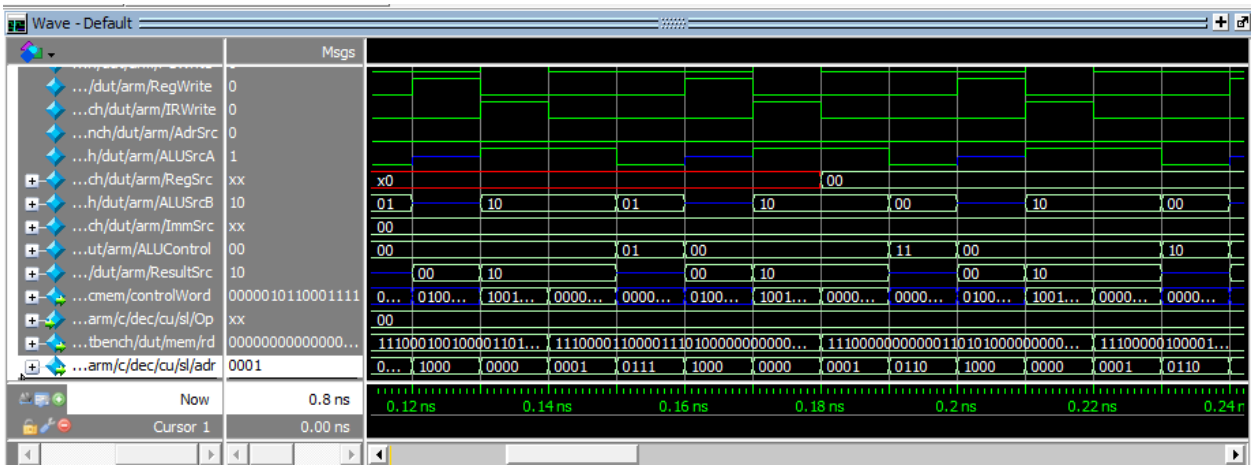
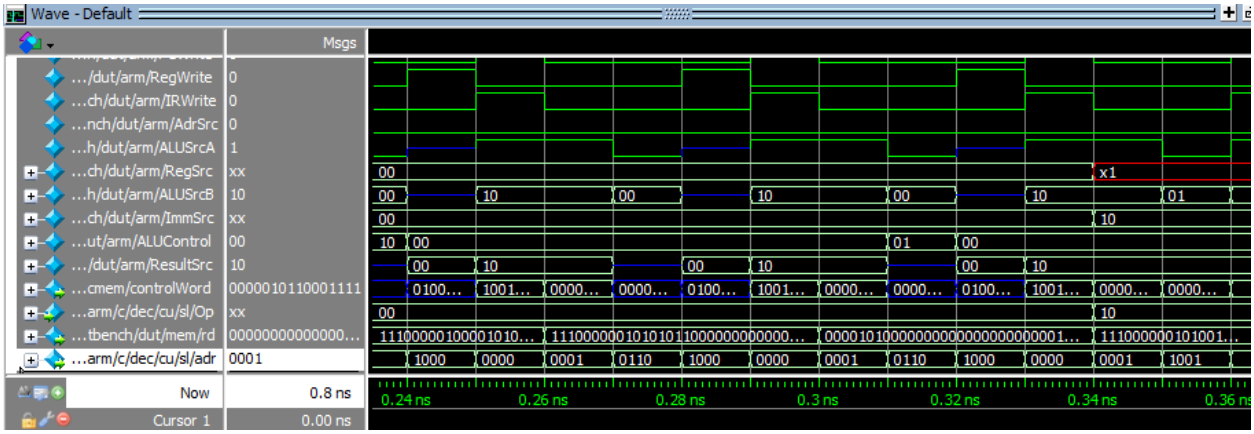


Figure 21.3 Control Unit Microarchitecture

شبیه سازی پروژه :

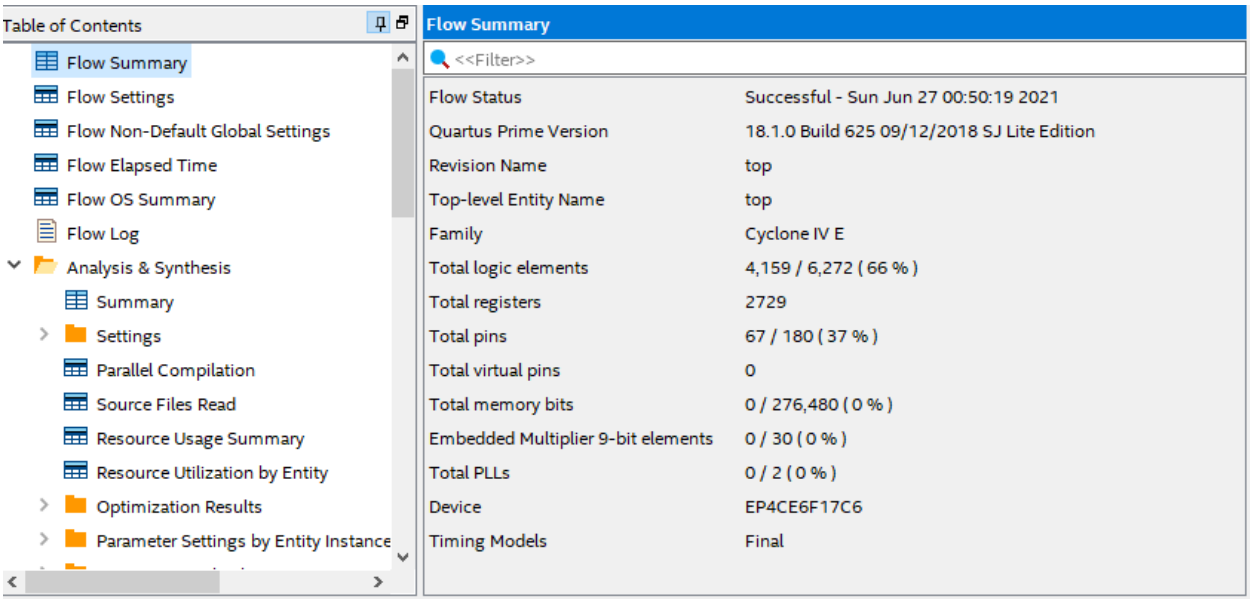






سنتز پروژه :

گزارش مساحت اشغال شده توسط چیپ ، سرعت و بیشینه فرکانس :



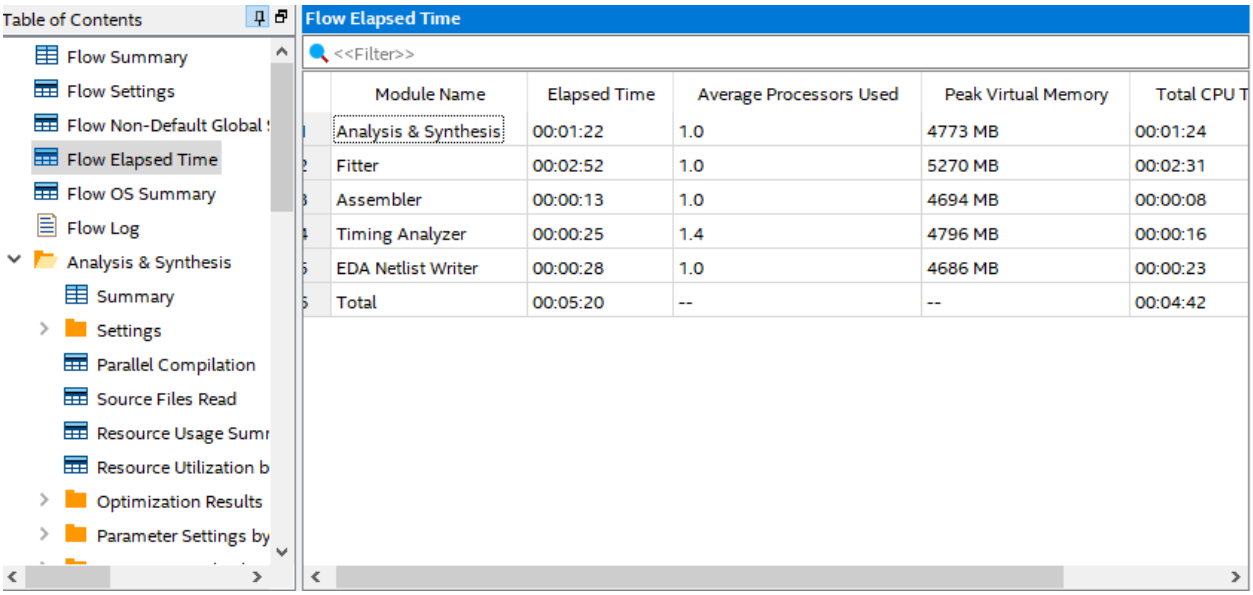


Table of Contents
Flow Summary
Flow Settings
Flow Non-Default Global
Flow Elapsed Time
Flow OS Summary
Flow Log
Analysis & Synthesis
Summary
Settings
Parallel Compilation
Source Files Read
Resource Usage Sumr
Resource Utilization b
Optimization Results
Parameter Settings by
Flow Elapsed Time
<<Filter>>
Module Name
Elapsed Time
Average Processors Used
Peak Virtual Memory
Total CPU T
Analysis & Synthesis
00:01:22
1.0
4773 MB
00:01:24
Fitter
00:02:52
1.0
5270 MB
00:02:31
Assembler
00:00:13
1.0
4694 MB
00:00:08
Timing Analyzer
00:00:25
1.4
4796 MB
00:00:16
EDA Netlist Writer
00:00:28
1.0
4686 MB
00:00:23
Total
00:05:20
--
--
00:04:42

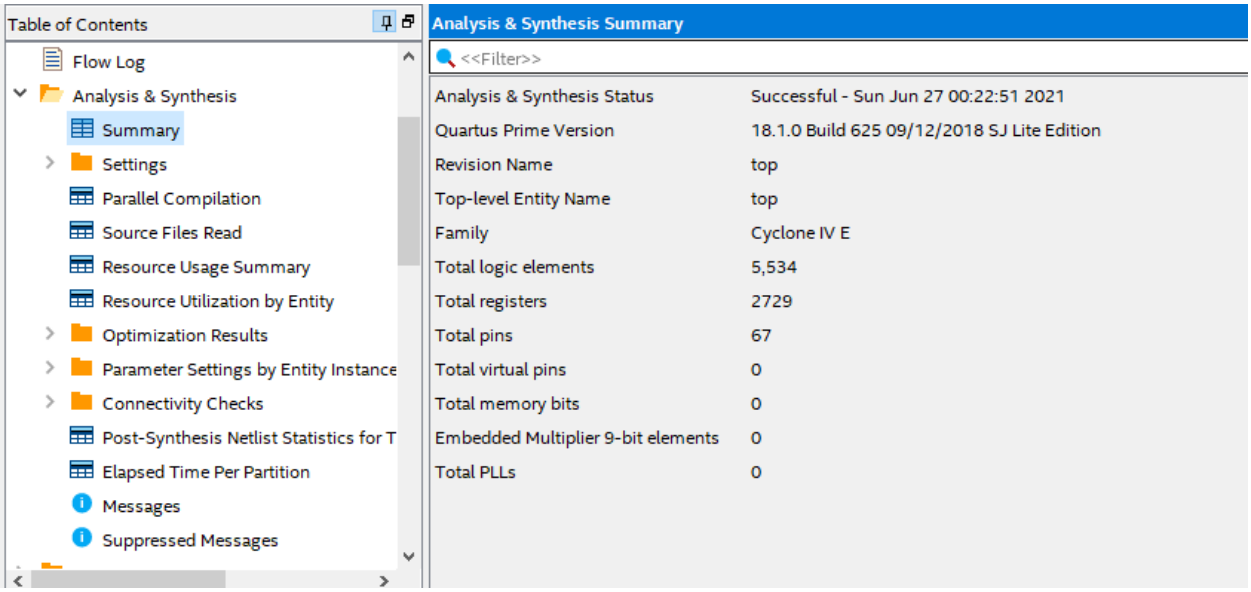
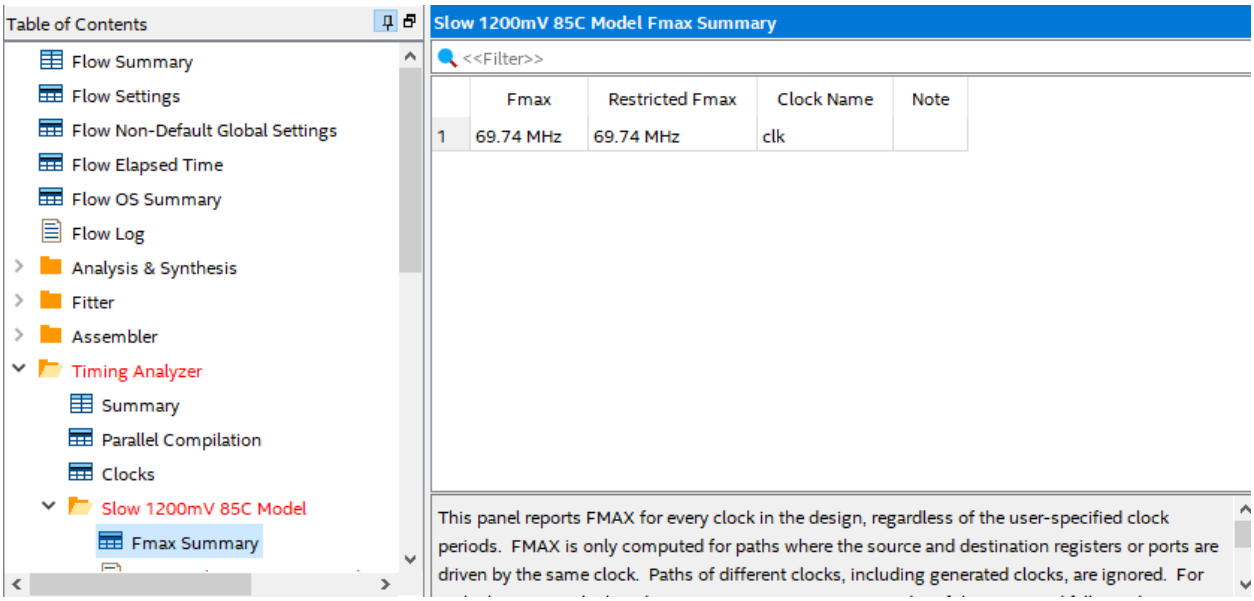
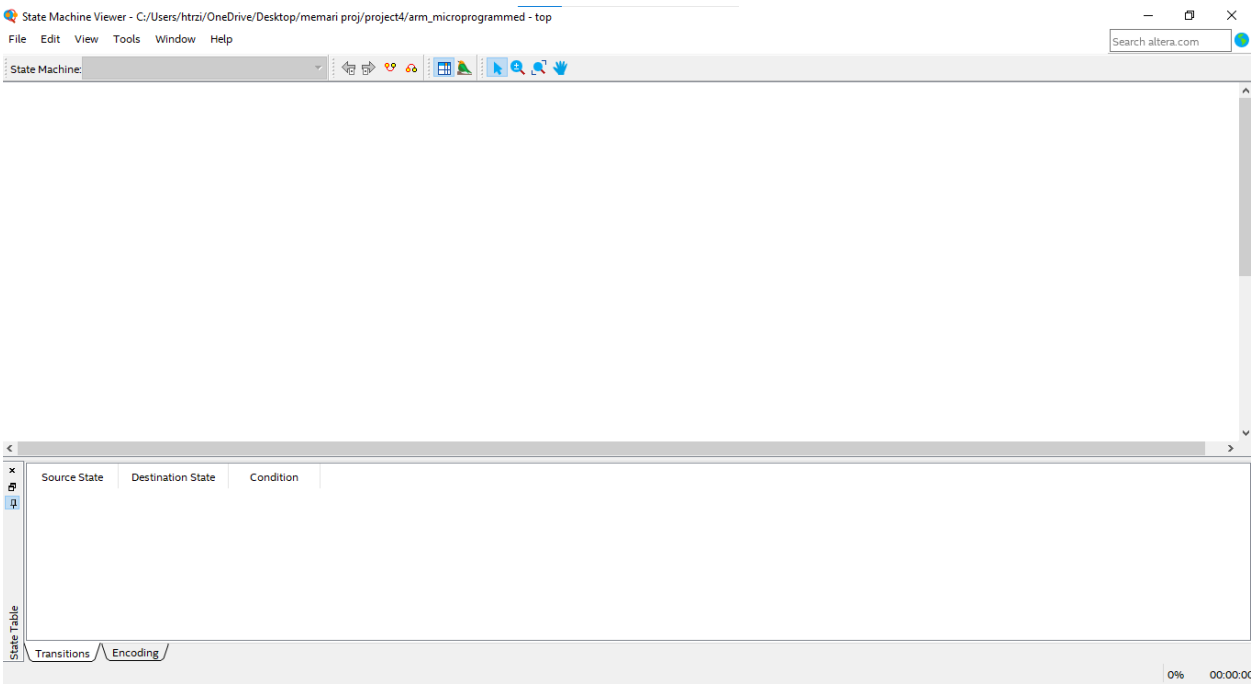


Table of Contents
Flow Log
Analysis & Synthesis
Summary
Settings
Parallel Compilation
Source Files Read
Resource Usage Summary
Resource Utilization by Entity
Optimization Results
Parameter Settings by Entity Instance
Connectivity Checks
Post-Synthesis Netlist Statistics for T
Elapsed Time Per Partition
Messages
Suppressed Messages
Analysis & Synthesis Summary
<<Filter>>
Analysis & Synthesis Status
Successful - Sun Jun 27 00:22:51 2021
Quartus Prime Version
18.1.0 Build 625 09/12/2018 SJ Lite Edition
Revision Name
top
Top-level Entity Name
top
Family
Cyclone IV E
Total logic elements
5,534
Total registers
2729
Total pins
67
Total virtual pins
0
Total memory bits
0
Embedded Multiplier 9-bit elements
0
Total PLLs
0

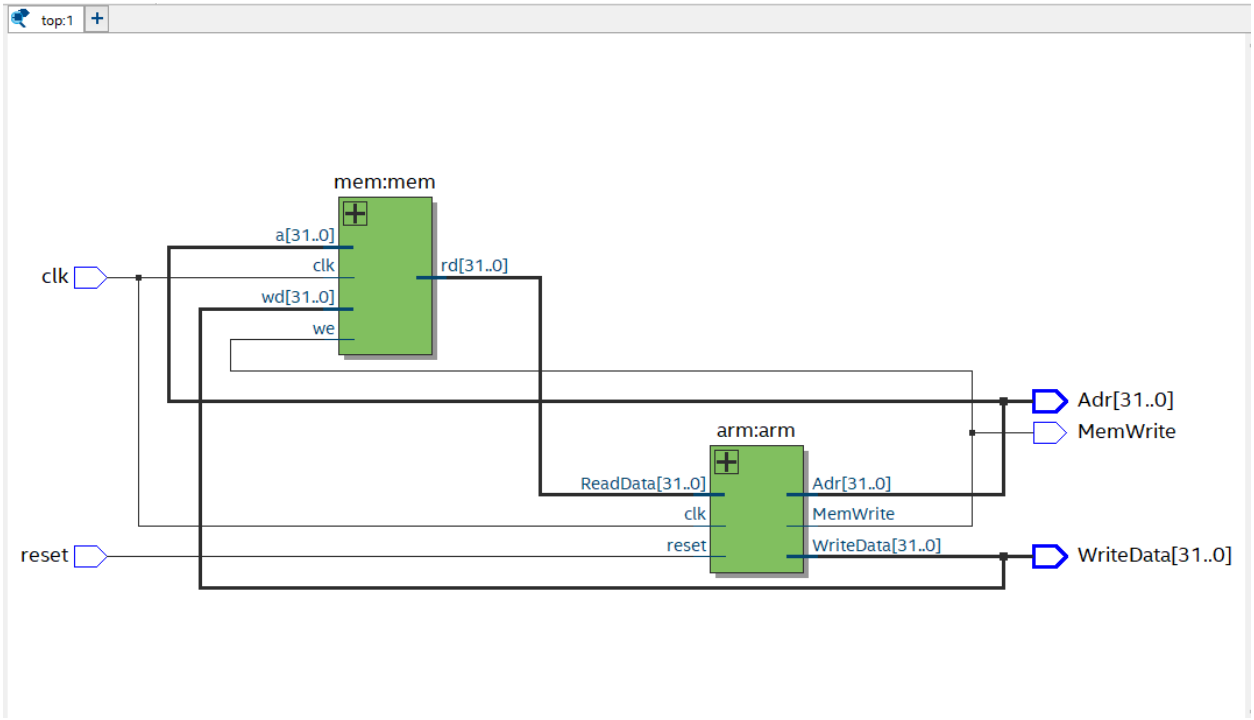


گزارش ماشین حالت : همان طور که انتظار داشتیم ، هیچ ماشین حالتی تشکیل نشده است.

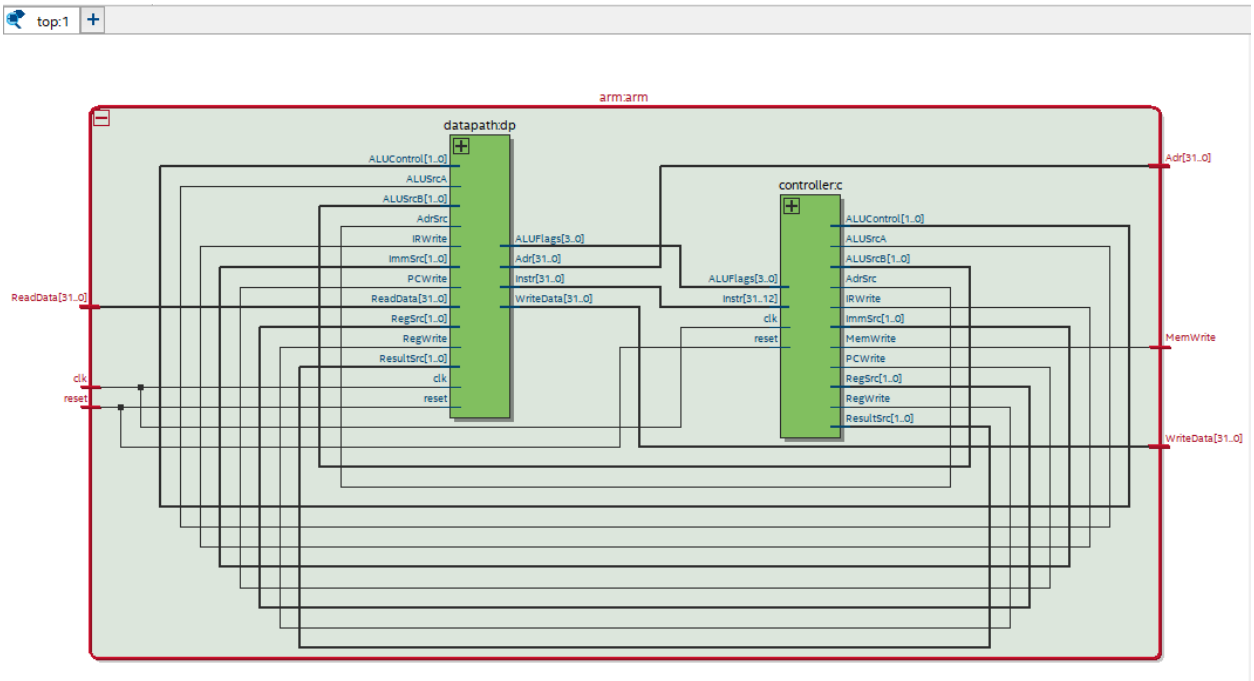


RTL Viewer :

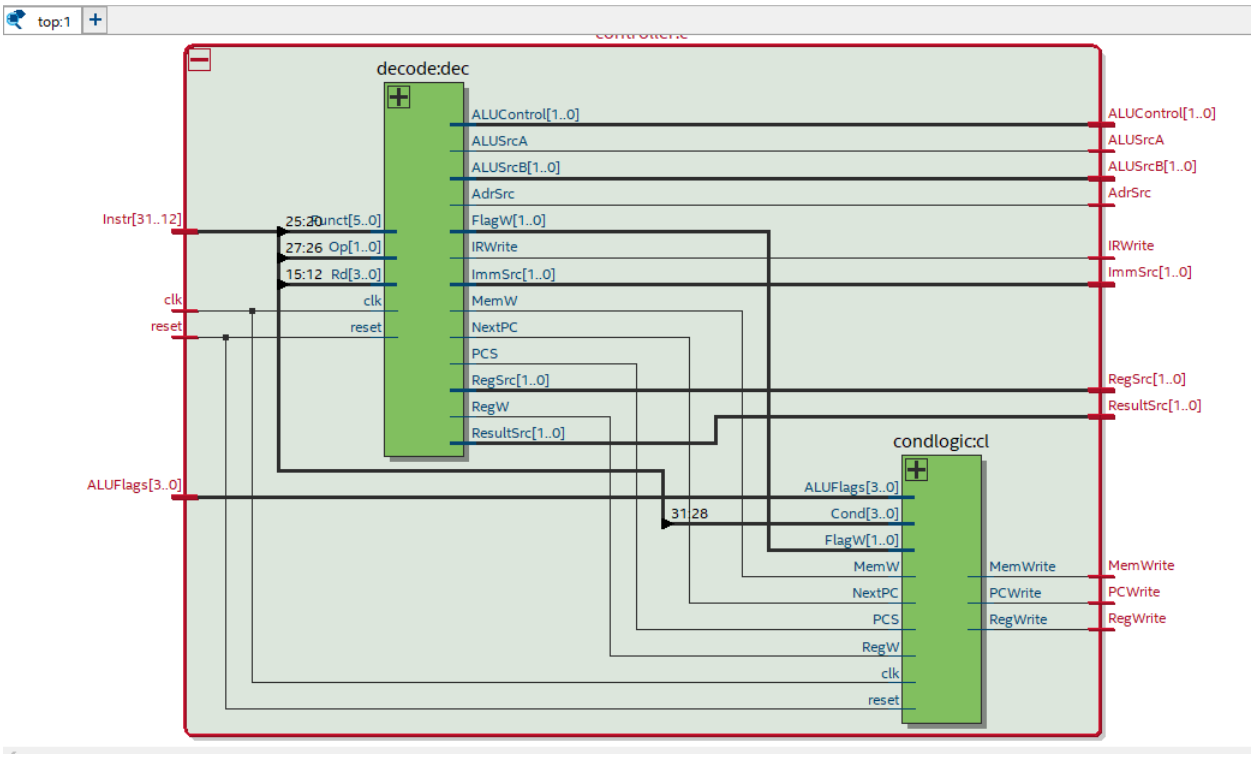
پردازنده و حافظه :



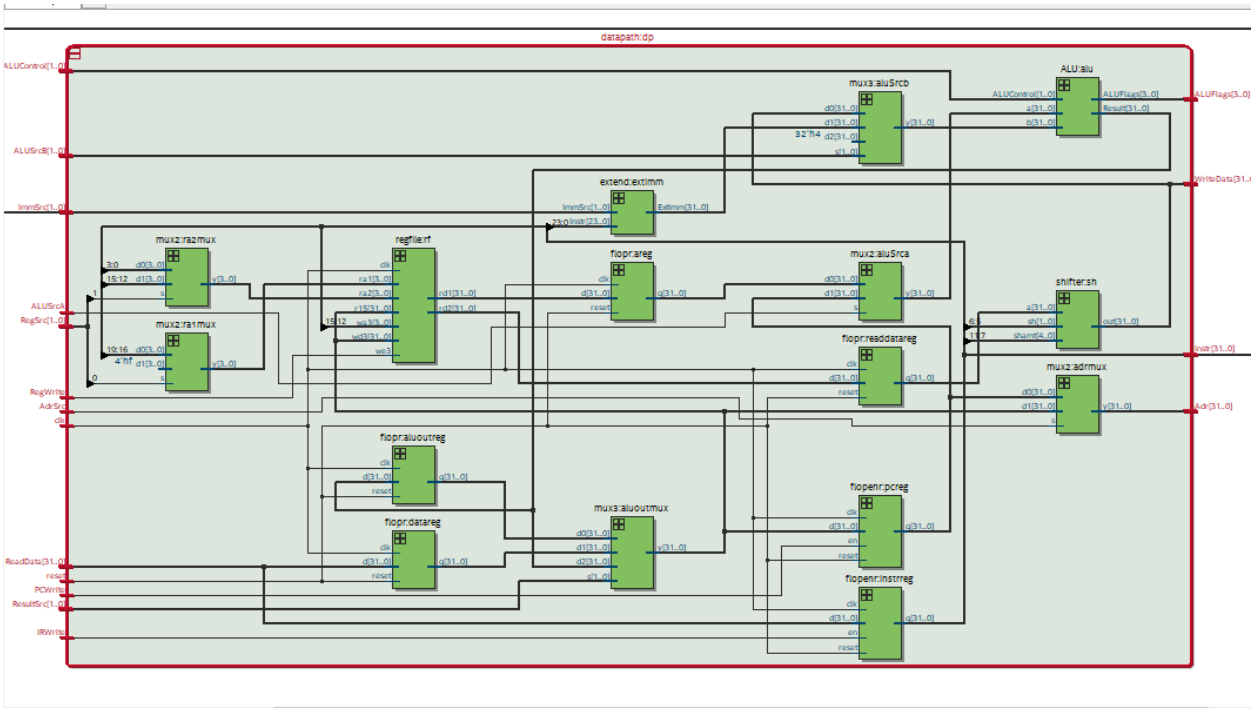
ساختار کلی پردازنده :



ساختار بخش کنترل :



ساختار مسیر داده :



ساختار قسمت decoder :