توضیحات مدار ضرب‌کننده - تمرین اول درس طراحی سیستم‌های دیجیتال

پارسا محمدیان - 98102284

الگوریتم مدار خواسته شده ساده است. الگوریتم Shift and Add همان الگوریتم متداول برای انجام ضرب توسط انسان است. برای توصیف این الگوریتم از زبان ASM استفاده می‌کنیم. پس در مرحله اول به سراغ رسم نمودار ASM می‌رویم.

در قدم اول Block آماده‌سازی یا Init را می‌کشیم. در این بلوک، منتظر فعال شدن سیگنال Start می‌مانیم. پس از 1 شدن سیگنال شروع، مقادیر in1 و in2 را به ترتیب در ثبات‌های A و B قرار می‌دهیم. A ثبات عمومی‌ای است که قابلیت شیفت به چپ و Load دارد و ثبات B نیز قابلیت شیفت به راست را در کنار قابلیت Load دارد. همچنین ثبات S و C و فلیپ فلاپ End نیز ریست می‌شوند و مدار داخل تمامی آن‌ها 0 می‌شود. ثبات S حاصل ضرب را در خود نگه میدارد. ثبات C را برای تشخیص پایان عملیات ضرب هر دفعه با یک جمع می‌کنیم. فلیپ فلاپ End نیز برای مشخص کردن تمام شدن عملیات ضرب استفاده می‌شود.

- نکته قابل توجه این است که برای این که تعداد بیت ضرب‌کننده ما محدود نباشد، تعداد بیت تمامی رجیسترها N در نظر گرفته شده. البته تعداد بیت رجیستر A و S دو برابر این مقدار یعنی 2N می‌باشد.

قدم بعدی رسم بلوکی است که در آن عملیات‌های شیفت و جمع کردن انجام می‌شود. نام این بلوک در نمودار Add گذاشته شده است. در ابتدای این بلوک و در State Box مقدار رجیستر A به چپ و مقدار رجیستر B به راست شیفت داده می‌شوند. به مقدار رجیستر C هم یکی اضافه می‌شود. تمامی این مقادیر آپدیت شده در کلاک بعدی استفاده می‌شوند. در ادامه این بلوک مقدار راست‌ترین رقم B بررسی می‌شود، اگر 1 باشد، مقدار A را به جواب خود اضافه می‌کنیم. اگر 0 باشد گویی چیزی به نتیجه اضافه نمی‌شود. این همان عملیات ضرب رقم ضرب‌کننده در مضروب است که در مبنای ده انجام می‌دهیم. در اینجا که مبنا دو است چون دو مقدار 0 یا 1 بیشتر نداریم، می‌توانیم به صورت گفته شده عمل کنیم. در ادامه‌ی هر دو شاخه، برابری مقدار C را با N – 1 مقایسه می‌کنیم. در صورت برابری این دو مقدار به پایان عملیات ضرب رسیده‌ایم چون مضروب ما N بیتی است. در غیر این صورت دوباره به همین بلوک می‌رویم.

برای تست الگوریتم به دست آمده، دو عدد را به عنوان ورودی بررسی می‌کنیم. همچنین ورودی را طوری تعیین می‌کنیم که به Coverage صد درصد برسیم. برای این کار عدد دوممان باید حداقل یک بیت داشته باشد که مقدارش صفر است. در این تست N را برابر 4 در نظر می‌گیریم و اعداد 1001 و 0110 را به عنوان

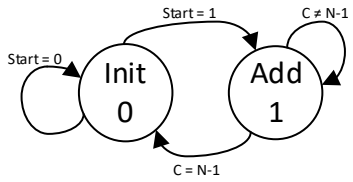
ورودی پردازش می‌کنیم. در جدول زیر مقدار هر یک از رجیسترها پس از خوردن هر کلاک ثبت می‌کنیم و بررسی می‌کنیم که آیا دنبال کردن نمودار ASM ما را به نتیجه‌ی مطلوب می‌رساند یا خیر.

#CLK	A	B	C	S	End
1	00000000	0000	0000	00000000	0
2	00001001	0110	0000	00000000	0
3	00010010	0011	0001	00000000	0
4	00100100	0001	0010	00100100	0
5	01001000	0000	0011	01101100	0
6	10010000	0000	0100	01101100	1

نتیجه به دست آمده برابر 01101100 است به درستی حاصل ضرب 1001 در 0110 را نشان می‌دهد. به عبارت دیگر در مبنای 10 این اعداد برابر $9 \times 6 = 72$ هستند.

- نمودار ASM و بقیه شکل‌ها در فایل Visio و خروجی PDF آن در کنار همین فایل متنی می‌باشند.

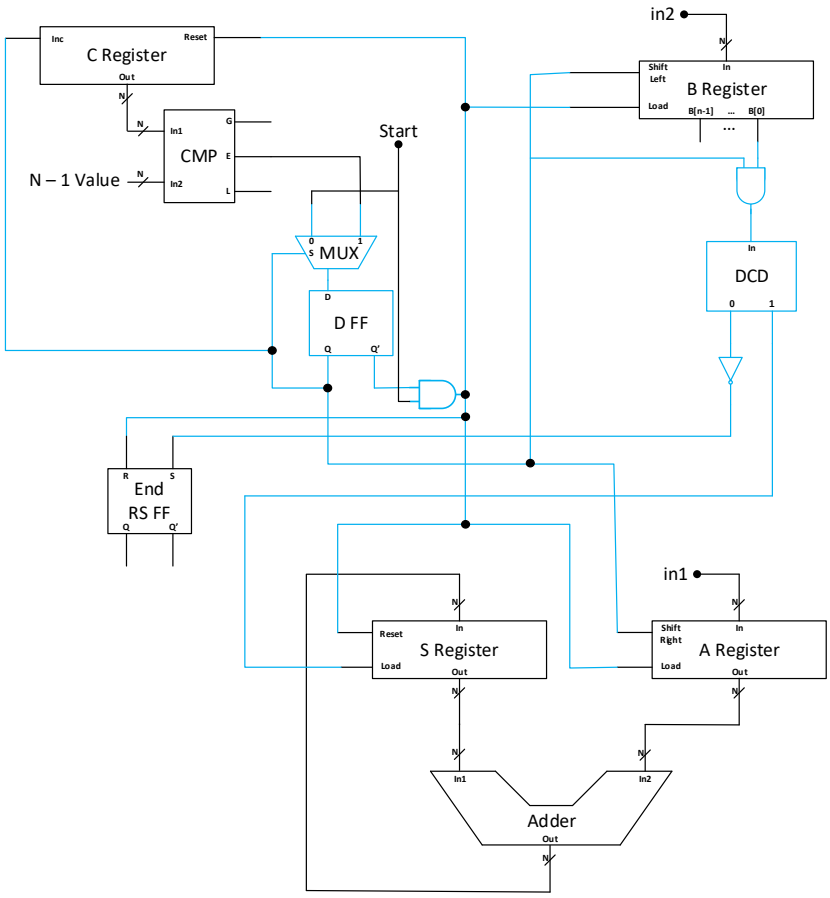
حال که نمودار ASM تکمیل شد، به سراغ پیاده سازی می‌رویم. برای پیاده‌سازی واحد کنترل، نیاز به رسم نمودار حالت داریم. پس از رسم نمودار حالت، واحد کنترل را به روش Flip Flop and Multiplexer پیاده سازی می‌کنیم. در این روش به $\log_2 n$ فلیپ فلاپ که در آن n تعداد حالات در نمودار حالت است، نیاز داریم. پس با توجه به نمودار حالت که شکل آن در ادامه آمده است، چون 2 حالت داریم به 1 فلیپ فلاپ نیازمندیم. واحد کنترل را با روش‌هایی که از مدار منطقی بلد هستیم پیاده سازی می‌کنیم و به سراغ مسیر داده می‌رویم.



مسیر داده را با کشیدن قطعات استفاده شده شروع می‌کنیم. قطعات مورد استفاده 5 رجیستر هستند که ویژگی‌های هر کدام قبلا ذکر شده است. یک فلیپ فلاپ برای سیگنال End داریم که در اینجا از فلیپ فلاپ RS استفاده شده است. در آخر یک مقایسه کننده و یک Adder هم داریم. همچنین یک سری سیگنال مانند Start نیز داریم.

در آخر با اضافه کردن چند قطعه در سطح Gate منطقی واحد کنترل را کامل می‌کنیم و به مسیر داده متصل می‌کنیم. در نهایت به مدار زیر خواهیم رسید.

- بدیهی است که کلاک به تمام اجزای متصل است ولی در شکل کشیده نشده است.


in2
N
Shift Left
In
B Register
Load
B[n-1]
...
B[0]
Inc
Reset
C Register
Out
N
N
In1
G
CMP
E
In2
L
N – 1 Value
N
Start
0
1
S
MUX
D
D FF
Q
Q'
In
DCD
0
1
R
S
End
RS FF
Q
Q'
in1
N
N
Reset
In
S Register
Load
Out
Shift Right
In
A Register
Load
Out
N
N
N
In1
In2
Adder
Out
N