دانشگاه صنعتی شریف

دانشکده مهندسی برق

پروژه درس مدارهای منطقی و سیستم های دیجیتال

پردازنده ۴ بیتی پایه

گروه دکتر شعبانی

پاییز ۱۴۰۲

دستیار آموزشی:

حسین چیت‌ساز

به نام خدا

شرح کلی:

در پروژه این درس قصد داریم یک پردازنده (CPU) ۴ بیتی پایه را با استفاده از FPGA پیاده سازی کنیم.

در حالت کلی یک پردازنده هیچ کاری انجام نمی‌دهد به جز دستورالعمل هایی که طراح سخت افزار پردازنده برای آن تعیین می‌کند. در پروژه این درس می‌خواهیم تعدادی از این دستورات را بر روی FPGA پیاده سازی کنیم.

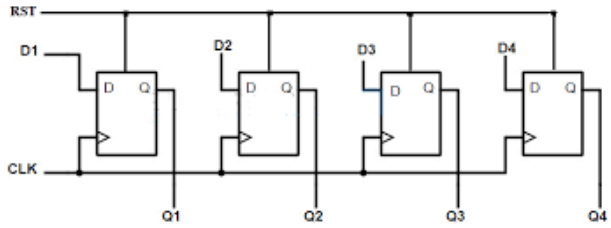
این دستورالعمل ها عبارت اند از:

- جمع (ADD)
- تفریق (SUB)
- ضرب (MUL)
- ذخیره کردن یک عدد (ST)
- Load کردن یک عدد (LD)
- مقایسه دو عدد (CMP)
- NAND
- NOR
- XOR

تعاریف:

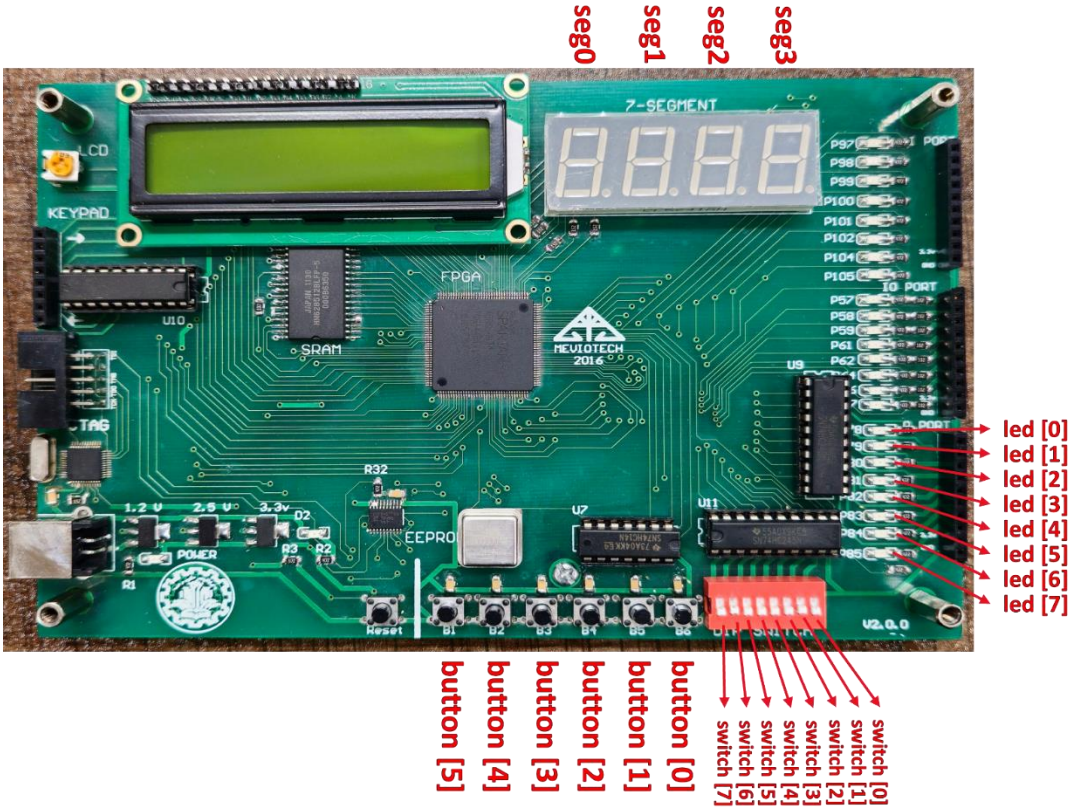
Register : در این CPU یک Register ۴ بیتی عبارت است از ۴ عدد D فلیپ فلاپ در کنار هم که کلاک همگی به هم دیگر متصل است. به عبارت دیگر یک Register ۴ بیتی را برای ذخیره یک داده ۴ بیتی استفاده می‌کنیم. هنگامی که لبه کلاک به فلاپ‌ها اعمال شود، ۴ بیت داده در ورودی، به خروجی منتقل شده و ذخیره می شوند.

همچنین در نظر داشته باشید که این Register ها می‌توانند قابلیت Reset هم داشته باشند.



یک Register ۴ بیتی

در ادامه جزئیات پروژه توضیح داده می‌شود. پیش از آن به نکات زیر دقت داشته باشید:



- تمامی ورودی و خروجی دستورالعمل ها و تمامی عملیات‌ها ۴ بیتی می باشند.
- این CPU شامل ۸ Register ۴ بیتی می باشد که نام آن ها را R0, R1, R2, R3, R4, R5, R6, R7 یا به اختصار در تعریف به زبان وریلاگ reg R[7:0]; می‌نامیم.
- با فشرده شدن button[5] بایستی تمامی Register ها صفر شوند.
- هر دستورالعمل کد خاص خود را دارد (که در ادامه توضیح داده خواهد شد) و با استفاده از ۴ بیت switch ورودی (switch[3:0]) نوع دستور العمل مشخص می شود سپس با فشردن button[0] به hardware خود این تاییدیه را می‌دهید که می‌خواهید این دستورالعمل را اجرا کنید. با این کار مشخص می‌کنید که به عنوان مثال اگر دستور ADD دارای دو ورودی ۴ بیتی x و y است، hardware پس از فشرده شدن هر بار button[0]، آماده دریافت یک ورودی ۴ بیتی x یا y باشد. (به ازای هر بار فشردن button[0]، یک ورودی دریافت می کنید)
- هر دستورالعمل با یک یا چند عدد ۴ بیتی در ورودی و خروجی خود سر و کار دارد. هر کدام از این اعداد باید ابتدا در Register ذخیره و سپس پردازش شوند.

- برای برخی دستورالعمل ها نیاز است تا مقدار یک عدد ۴ بیتی را به عنوان ورودی تعیین کنید. برای این کار از switch های ۴ تا ۷ (switch[7:4]) استفاده کنید.
- بر روی برد FPGA یک نمایشگر LCD و یک 7segment وجود دارد که بایستی نوع دستورالعمل و هر اتفاقی که می‌افتد را بر روی LCD نمایش دهید.
- پس از اتمام عملیات هر دستورالعمل، کد شما بایستی این قابلیت را داشته باشد که نتیجه نهایی را هم بر روی LCD و هم 7segment نمایش دهد. به صورت پیش فرض، نتیجه نهایی هر دستورالعمل بایستی بر روی LCD نمایش داده شود اما اگر button[1] فشرده شد، باید این نتیجه بر روی 7segment نیز نمایش داده شود.
- دقت داشته باشید که بایستی پین های LCD را از پیوست شماره ۶ دستورکار موجود در سامانه درس استخراج کنید و به فایل ucf داده شده برای آزمایش ۴، این pin assignment ها را اضافه کنید.

شرح جزئی:

کد هر دستورالعمل در جدول زیر آمده است:

Instruction	Switch[3:0]
ADD	0001
SUB	0011
MUL	0111
ST	1111
LD	1110
CMP	1100
NAND	1000
NOR	1001
XOR	1011

مثال: زمانی که مقدار switch[3:0] را 0001 می‌دهید، یعنی برد باید عملیات جمع c=a+b را انجام دهد. حال button[0] را فشار دهید تا ورودی switch[3:0] را بخواند (که در اینجا همان 0001 است). پس از فشردن این کلید، برد شما متوجه می شود که میخواهید عملیات جمع را انجام دهید؛ زیرا در کد وریلاگ خود، کد دستورالعمل 0001 را به عنوان عملیات جمع تعریف کرده اید. سپس برای اینکه بتوانید ورودی a را دریافت کنید، (در اینجا

مثلا می‌خواهیم مقدار a را 0101 یا همان ۵ بدهیم) مقدار switch[7:4] را بر روی 0101 تنظیم می کنیم و سپس button[0] را فشار می دهیم تا عدد ۵ یا همان 0101، برای پردازش پس از دریافت ورودی ها، در یکی از Register ها ذخیره شود. برای دریافت b نیز مانند a عمل می کنیم. حال a و b هر کدام به صورت جداگانه بر روی Register ذخیره شده است. حال می توانید حاصل c=a+b را محاسبه کنید و c را در یکی از Register ها ذخیره کنید. پس از ذخیره کردن c، می‌توانید آن را نمایش دهید.

تذکر: در انجام برخی دستورالعمل ها ممکن است عدد حاصل از ۴ بیت بیشتر شود. با توجه به اینکه Register های ما ۴ بیتی هستند، پس باید هر حاصلی که بیشتر از ۴ بیت شد، در بیش از یک Register ذخیره شود.

توضیحات دستورالعمل ها:

دستور ADD/SUB:

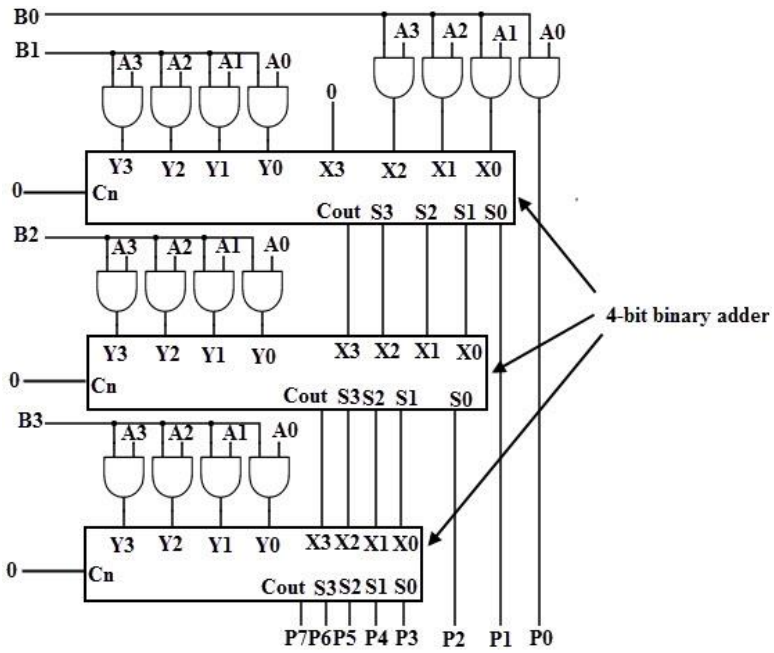
در این دستور شما باید دو عدد ۴ بیتی باینری علامت دار را دریافت کنید و حاصل را محاسبه کنید. دقت کنید که پیش بینی وقوع هر رخدادی مانند overflow، carry out، borrow و ... به عهده شماست و حاصل محاسبه شده در خروجی باید برای هر ورودی صحیح باشد.

دستور MUL:

در این دستور شما دو عدد ۴ بیتی باینری بدون علامت دریافت می کنید و حاصل محاسبه شده را نمایش می دهید. دقت کنید که حاصل ضرب دو عدد n بیتی، یک عدد 2n بیتی است. پس برای ذخیره حاصل ضرب، به ۲ Register نیاز دارید.

مدار زیر ضرب کننده دو عدد ۴ بیتی می باشد.

در صورت نیاز، در مورد فرایند ضرب دو عدد بدون علامت توضیحات بیشتر ارائه خواهد شد.



ضرب کننده ۲ عدد ۴ بیتی بدون علامت

دستور ST:

در این دستور شما باید یک عدد (مثلا z) را در یکی از Register ها ذخیره کنید. ابتدا یک عدد باینری ۴ بیتی را بر روی switch[7:4] ایجاد کنید و button[0] را فشار دهید تا این عدد را نگهداری کنید. این عدد مشخص کننده شماره Register ای است که می‌خواهید عدد z را در آن ذخیره کنید. حال دوباره button[0] را فشار دهید تا عدد z را از switch[7:4] دریافت کنید و در نهایت آن را در Register ای که شماره آن را دریافت کردید، ذخیره کنید.

دستور LD:

در این دستور یک عدد ۴ بیتی باینری (به نام k) را بر روی switch[7:4] مشخص می‌کنید و سپس button[0] را فشار می‌دهید. حال بایستی محتوای عددی که در Register شماره k ذخیره شده را نمایش دهید.

دستور CMP:

با فعال شدن این دستور، دو عدد بدون علامت همانند اعداد a و b در بخش دستور ADD/SUB، دریافت کنید. سپس این دو عدد را مقایسه کنید و اگر a>b بود، G را نمایش دهید. همچنین برای a<b، L و اگر a=b بود، E را نمایش دهید.

دستور NAND/NOR/XOR:

با فعال شدن این دستور، دو عدد همانند اعداد a و b در بخش دستور ADD/SUB، دریافت کنید و سپس بسته به نوع دستور، حاصل NAND/NOR/XOR این دو عدد را در خروجی به صورت **باینری** نمایش دهید.

پیشنهاد می‌شود دوباره نکاتی که در ابتدای پروژه با دایره قرمز مشخص شده اند را مطالعه کنید.

موفق باشید.