به نام خدا

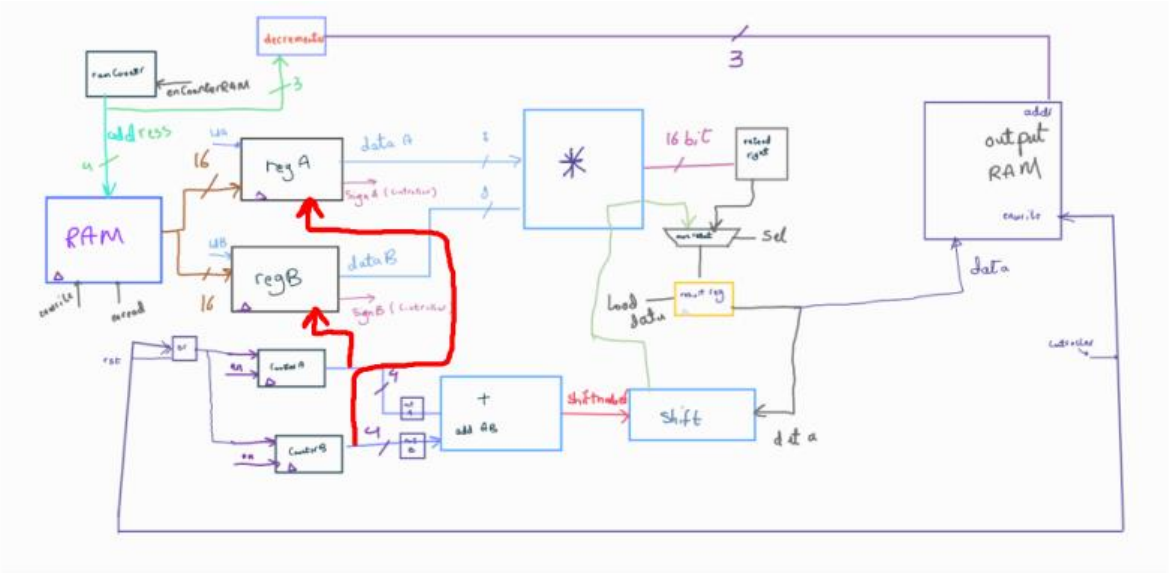
گزارش کار پروژه cad 1

مرضیه موسوی 810101526

مهراد لیویان 810101501

طراحی ضرب کننده تقریبی:

1-دیتاپث:



2-معرفی component ها:

Special reg:

به عنوان ورودی علاوه بر ورودی های معمول مانند rst و clk و load و دیتای ورودی چند ورودی خاص دیگر می گیرد:

Address : پوینتر به یکی از بیت های مقدار موجود در رجیستر.

از address برای جدا کردن 8 بیت خروجی و خروجی دادن بیت ساین که درباره ی آن صحبت خواهیم کرد استفاده می شود.

خروجی این رجیستر شامل 8 بیت خروجی و بیت sign است.بیت sign به کنترلر می رود و اگر یک باشد 8 بیت بعد از آدرس مورد نظر جدا می شود و در غیر این صورت آدرس که خروجی counter است کم می شود و به بیت بعد اشاره می کند.(sign از بیت msb شروع می شود)

:RAM

این قطعه 4 سیگنال به اسم های readfile writefile enwrite و enread می گیرد.

Enwrite و enread اجازه ی خواندن و نوشتن از روی خانه های حافظه را می دهد.

2 سیگنال دیگر از روی فایل می خوانند یا بر روی فایل می نویسند.

Extendright:به تعداد بیت های ورودی 0 به سمت راست دیتای ورودی اضافه می کند.

Upcounter وdowncounter: پیاده سازی این دو شمارنده مانند شمارنده های معمول است.با یک تفاوت.به عنوان ورودی به counter ها می توان offset نیز داد که اگر مثلا با counter 4بیتی تصمیم گرفتیم 8 تا بشماریم اگر دیتای ما به آفست رسید carryout مساوی یک می شود و شمارنده ریست می شود.

Mul:ضرب کننده ی ما که به طور دیفالت 8 بیتی می باشد

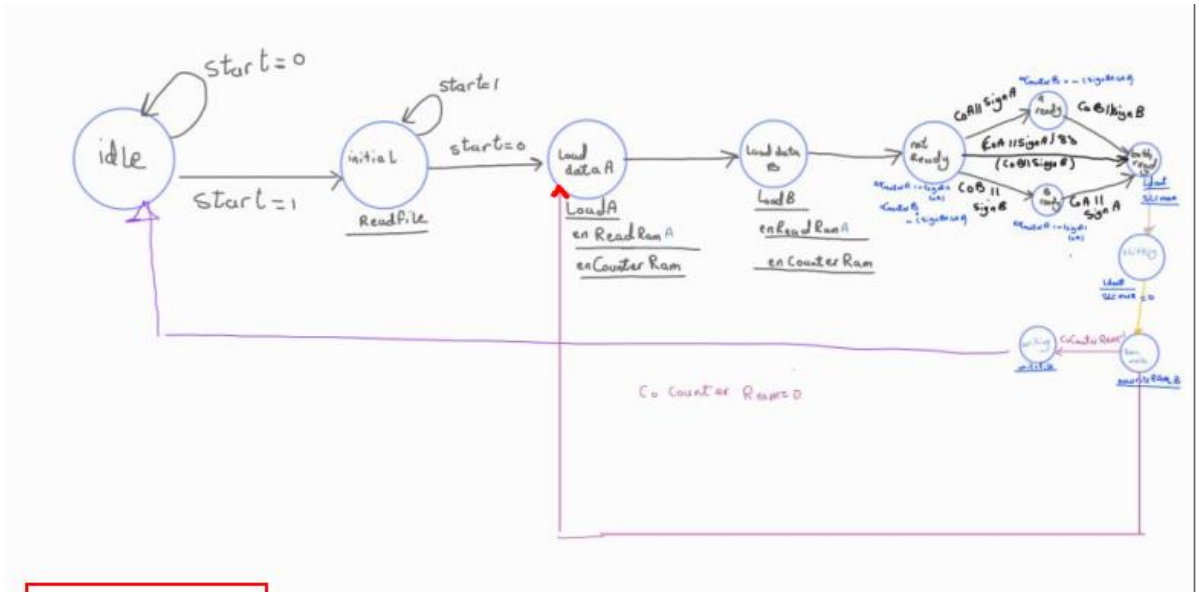
Not:ورودی به صورت not bitwise می شود

Register:رجیستر معمولی با پایه لود و ریست

Shiftright:دیتا را به اندازه آفست به سمت راست شیفت می دهد.

Adder:جمع کننده 4 بیتی برای به دست آوردن bit های worthless

3- controller:



روند کلی کنترلر به صورت زیر است:

ابتدا کنترلر منتظر سیگنال استارت می ماند و وقتی پالس مثبت استارت را گرفت شروع به خواندن فایل می کند. ابتدا ردیف اول را می خوانیم و سپس ردیف دوم.همزمان دو شمارنده مشغول به کارند که پوینتر به بیت های رجیستر A و B هستند و شروعشان از پر ارزش ترین بیت دو عدد است.هر بیتی که پوینتر اشاره می کند به خروجی sign دو رجیستر می رود و به کنترلر می رود.از طرفی خود خروجی شمارنده ها به یک جمع کننده می رود تا بتوانیم بیت های بی ارزش را بشماریم.اگر 8 تا شمردیم یا یک دیدیم(که با سیگنال های sign و co تشخیص داده می شود).از آن به بعد 8 تا می شماریم و آن 8 بیت را به ضرب کننده می دهیم.خروجی 16 بیتی وارد اکستند می شود و 16 بیت از راست اکستند می شود.سپس برای اضافه کردن بیت های بی ارزش از چپ ابتدا مولتی پلکسر خروجی extend right را در رجیستر ذخیره می کند.خروجی جمع کننده را به عنوان عدد شیفت به شیفت دهنده می دهیم و دوباره خروجی را با استفاده از مولتی پلکسر در رم ذخیره می کنیم.در آخر این پروسه در رم می نویسیم.خود رم یک شمارنده دارد که نشان می دهد آیا همه ی خانه ها را خواندیم یا خیر.که اگر هنوز خواندنمان تمام نشده بود دو عدد بعدی را می خوانیم و در غیر این صورت سیگنال done را issue می کنیم.

4تست کردن دو ورودی:

با استفاده از تست کردن دو ورودی، صحت کد خود را امتحان می کنیم.خروجی را در فایل out.txt می نویسیم.

ورودی اول:

```
00000110100000010000000000000000
00010001111000101100000000000000
10110001010011100000000000000000
00101010101010000100000000000000
00101011011000010100000000000000
00001100100100110000000000000000
00011001100101001100000000000000
00111000101101100000000000000000
06810000
11E2C000
B14E0000
2AA84000
2B614000
0C930000
1994C000
38B60000
```

مقایسه خروجی ها:

خروجی ما:

```
// memory data file (do not edit the following line - required for mem load use)
// instance=/TB/UUT/dataPath/outputRam/memory
// format=hex addressradix=h dataradix=h version=1.0 wordsperline=1 noaddress
06810000
11e2c000
b14e0000
2aa84000
2b614000
0c930000
1994c000
38b60000
```

خروجی تست کیس جنریت شده:

```
00000110100000010000000000000000
00010001111000101100000000000000
10110001010011100000000000000000
00101010101010000100000000000000
00101011011000010100000000000000
00001100100100110000000000000000
00011001100101001100000000000000
00111000101101100000000000000000
06810000
11E2C000
B14E0000
2AA84000
2B614000
0C930000
1994C000
38B60000
```

ورودی دوم:

```
0000101000011111
0000000111010100
1110001000011100
1011001111111101
1010100101111111
0011100100100110
1010101000001111
1111111101101000
0011010101011100
0010011111010111
1101000011001111
1110001000011010
1101101011100101
1110110010100011
0011001101011110
1111011000101000
```

خروجی ما:

```
// memory data file (do not edit the following line - required for mem load use)
// instance=/TB/UUT/dataPath/outputRam/memory
// format=hex addressradix=h dataradix=h version=1.0 wordsperline=1 noaddress
00126540
9e060000
25a10000
a9560000
0844b000
b7a00000
c8f80000
313f8000
```

خروجی تست کیس جنریت شده:

```
00000000000100100110010101000000
10011110000001100000000000000000
00100101101000010000000000000000
10101001010101100000000000000000
00001000010001001011000000000000
10110111101000000000000000000000
11001000111110000000000000000000
00110001001111111000000000000000
00126540
9E060000
25A10000
A9560000
0844B000
B7A00000
C8F80000
313F8000
```