گزارش بخش اول فاز اول

محمد جوشقانی

۹۵۲۰٤٥٦۹

بخش های اصلی این بخش عبارتست از ٤ ماژول خواسته شده و testbench.v که شامل testbench برای ٤ ماژول خواسته شده است. نتیجه اجرای تست بنچ صحت عملکرد هر بلوک را به شکل زیر مشخص می‌کند:

برای Register File :

در ابتدا از فایل register.dat مقادیر را میخوانیم و به ترتیب A1 و A2 را یکی یکی زیاد می‌کنیم و خروجی را می‌خوانیم. نتیجه این بخش عبارتست از:

```
#reading current registers:
#A1= 2 ,RD1=    65535  ***   A2= 3 ,RD2=4294901760
#A1= 4 ,RD1=4294901760  ***   A2= 5 ,RD2=    65535
#A1= 6 ,RD1=    65535  ***   A2= 7 ,RD2=4294901760
#A1= 8 ,RD1=4294901760  ***   A2= 9 ,RD2=    65535
#A1=10 ,RD1=    65535  ***   A2=11 ,RD2=4294901760
#A1=12 ,RD1=4294901760  ***   A2=13 ,RD2=    65535
#A1=14 ,RD1=    65535  ***   A2=15 ,RD2=4294901760
#A1=16 ,RD1=4294901760  ***   A2=17 ,RD2=    65535
#A1=18 ,RD1=    65535  ***   A2=19 ,RD2=4294901760
#A1=20 ,RD1=4294901760  ***   A2=21 ,RD2=    65535
#A1=22 ,RD1=    65535  ***   A2=23 ,RD2=4294901760
#A1=24 ,RD1=4294901760  ***   A2=25 ,RD2=    65535
#A1=26 ,RD1=    65535  ***   A2=27 ,RD2=4294901760
#A1=28 ,RD1=4294901760  ***   A2=29 ,RD2=    65535
#A1=30 ,RD1=    65535  ***   A2=31 ,RD2=4294901760
```

که مطابق فایل register.dat است و لذا بخش read این بلوک صحیح است

برای چک کردن بخش write ، مقدار A3 از ۱ به بالا زیاد می‌گردد و با کمی تاخیر A2 برابر A3 می‌شود تا مقدار نوشته شده را بخوانیم:

#write and verify write

#A3= 1, WD3= 0 *** A2= 1, RD2= 0

#A3= 2, WD3= 1 *** A2= 2, RD2= 1

#A3= 3, WD3= 2 *** A2= 3, RD2= 2

#A3= 4, WD3= 3 *** A2= 4, RD2= 3

#A3= 5, WD3= 4 *** A2= 5, RD2= 4

#A3= 6, WD3= 5 *** A2= 6, RD2= 5

#A3= 7, WD3= 6 *** A2= 7, RD2= 6

#A3= 8, WD3= 7 *** A2= 8, RD2= 7

#A3= 9, WD3= 8 *** A2= 9, RD2= 8

#A3=10, WD3= 9 *** A2=10, RD2= 9

#A3=11, WD3= 10 *** A2=11, RD2= 10

#A3=12, WD3= 11 *** A2=12, RD2= 11

#A3=13, WD3= 12 *** A2=13, RD2= 12

#A3=14, WD3= 13 *** A2=14, RD2= 13

#A3=15, WD3= 14 *** A2=15, RD2= 14

#A3=16, WD3= 15 *** A2=16, RD2= 15

#A3=17, WD3= 16 *** A2=17, RD2= 16

#A3=18, WD3= 17 *** A2=18, RD2= 17

#A3=19, WD3= 18 *** A2=19, RD2= 18

#A3=20, WD3= 19 *** A2=20, RD2= 19

#A3=21, WD3= 20 *** A2=21, RD2= 20

#A3=22, WD3= 21 *** A2=22, RD2= 21

#A3=23, WD3= 22 *** A2=23, RD2= 22

#A3=24, WD3= 23 *** A2=24, RD2= 23

#A3=25, WD3= 24 *** A2=25, RD2= 24

#A3=26, WD3= 25 *** A2=26, RD2= 25

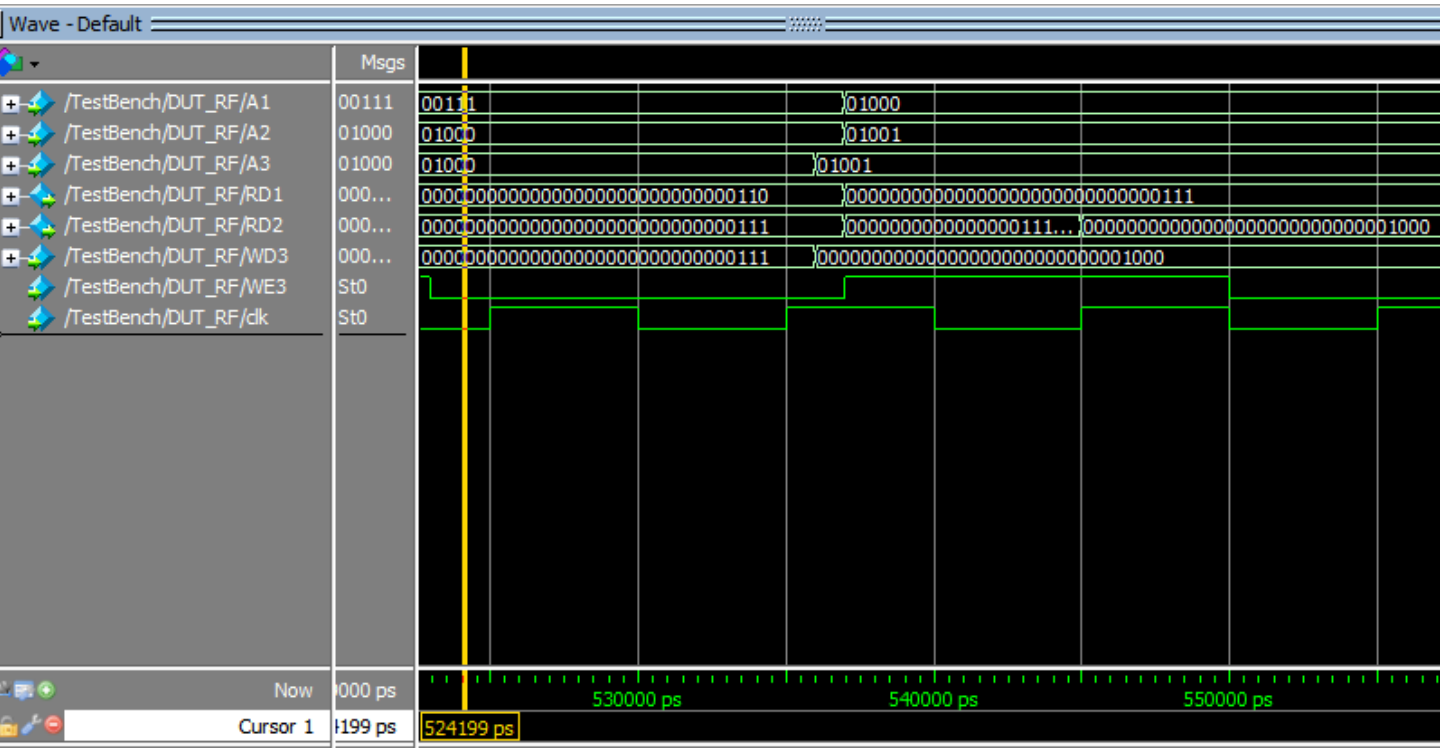
#A3=27, WD3= 26 *** A2=27, RD2= 26

#A3=28, WD3= 27 *** A2=28, RD2= 27

#A3=29, WD3= 28 *** A2=29, RD2= 28

مطابق انتظار مقادیر با هم برابر می‌باشند.

نمودار این بخش نیز به شکل زیر است:



برای ALU :

برای ALU نیز دو مقدار دلخواه در testbench به ورودی‌ها داده شده است و تمامی عملیات ممکن بر روی آن به صورت زیر چک شده است:

```
**************************************************#
        #Testing ALU
**************************************************#
#checking alu functions:
#and:
#SrcAE=0000f000   ,SrcBE=000000ff   ,ALUCtrlE= 0   ***   ALUOutE=00000000   Zero=1
#OR:
```

#SrcAE=0000f000 ,SrcBE=000000ff ,ALUCtrlE= 1 *** ALUOutE=0000f0ff Zero=0

#add:

#SrcAE=0000f000 ,SrcBE=000000ff ,ALUCtrlE= 2 *** ALUOutE=0000f0ff Zero=0

#slt

#SrcAE=0000f000 ,SrcBE=000000ff ,ALUCtrlE= 3 *** ALUOutE=00000000 Zero=1

#XOR

#SrcAE=0000f000 ,SrcBE=000000ff ,ALUCtrlE= 4 *** ALUOutE=0000f0ff Zero=0

#NOR

#SrcAE=0000f000 ,SrcBE=000000ff ,ALUCtrlE= 5 *** ALUOutE=ffff0f00 Zero=0

#sub

#SrcAE=0000f000 ,SrcBE=000000ff ,ALUCtrlE= 6 *** ALUOutE=0000ef01 Zero=0

#lui

#SrcAE=0000f000 ,SrcBE=000000ff ,ALUCtrlE= 7 *** ALUOutE=00000000 Zero=0

عملکرد این بلوک هم مطابق انتظار بوده است

برای بلوک Data_Memory :

چک کردن این بلوک عینا مشابه Register_File می‌باشد. یعنی ابتدا از فایل memory.dat اطلاعاتی را خوانده و می‌بینیم، و سپس روی خانه‌های حافظه مقادیر خاصی write می‌کنیم و نتیجه را می‌بینیم. داده‌ای که write می‌کنیم برابر آدرس آن خانه حافظه است.:

reading memory.dat***

********** #

#reading all data memory:

address= 1 ======> data= 15

address= 2 ======> data= 240

address= 3 ======> data= 3840

address= 4 ======> data= 61440

address= 5 ======> data= 983040

address= 6 ======> data= 15728640

address= 7 ======> data= 251658240

address= 8 ======> data=4026531840

address= 9 ======> data=4026531840

address= 10 ======> data= 251658240

address= 11 ======> data= 15728640

address= 12 ======> data= 983040

address= 13 ======> data= 61440

address= 14 ======> data= 3840

address= 15 ======> data= 240

address= 16 ======> data= 15

address= 17 ======> data= 15

...

_____

**

write and verify writen data

Adress= 0 =========> data= 0

Adress= 1 =========> data= 1

Adress= 2 =========> data= 2

Adress= 3 =========> data= 3

Adress= 4 =========> data= 4

Adress= 5 =========> data= 5

Adress= 6 =========> data= 6

Adress= 7 =========> data= 7

Adress= 8 =========> data= 8

Adress= 9 =========> data= 9

Adress= 10 =========> data= 10

Adress= 11 =========> data= 11

Adress= 12 =========> data= 12

Adress= 13 =========> data= 13

Adress= 14 =========> data= 14

Adress= 15 =========> data= 15

```
# Adress=      16 =========> data=      16
# Adress=      17 =========> data=      17
# Adress=      18 =========> data=      18
# Adress=      19 =========> data=      19
# Adress=      20 =========> data=      20
…
```

برای Instruction Memory :

عینا مشابه Data Memory است با این تفاوت که write ندارد:

```
 #reading instruction.dat***
************************ #
# reading all instructions in memory :
# address=       0  ======>  instr=4294967295
# address=       1  ======>  instr=4294967295
# address=       2  ======>  instr=4294967295
# address=       3  ======>  instr=4294967295
# address=       4  ======>  instr=4294967295
# address=       5  ======>  instr=4294967295
# address=       6  ======>  instr=4294967295
# address=       7  ======>  instr=4294967295
# address=       8  ======>  instr=4294967295
# address=       9  ======>  instr=4294967295
# address=      10  ======>  instr=4294967295
# address=      11  ======>  instr=4294967295
# address=      12  ======>  instr=4294967295
# address=      13  ======>  instr=4294967295
# address=      14  ======>  instr=4294967295
# address=      15  ======>  instr=4294967295
# address=      16  ======>  instr=4294967295
```

```
# address=       17  ======>  instr=4294967295
# address=       18  ======>  instr=4294967295
```