中山大學 SUN YAT-SEN UNIVERSITY

实 验 报 告

姓 名：刘焕新 **学 号：**11331206

院系专业：软件学院11级嵌入式软件系统开发与设计

完成日期：2013 年12 月17日

实验题目：CPU Design

目录

一、实验目的

1、熟练掌握使用 verilog 语言开发技能。
2、熟练掌握流水线设计方法。
3、加深对 CPU 运作的了解。
4、掌握独立设计操作集并实现的能力

二、实验内容

在 Xilinx ISE 设计平台上完成简单 CPU 的设计，包括操作集的设计，流水线设计及简单的结果模拟验证等。

三、实验设计

（1）操作集设计

根据 PPT 给出的操作集进行设计，为每一个操作设计唯一的操作码，由于本次实验设计的操作集总共能够容纳 32 条操作指令，故自己添加以下五条操作指令：

分别是：INC，DEC，NOT，SRR，SLR,由红色标识加以分辨。

每一条指令的详细描述由下表给出

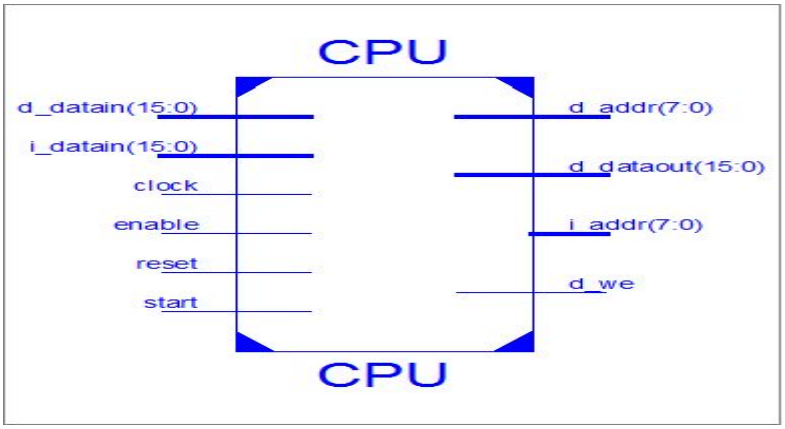
十六位指令格式	15 11	10 8	7 4	3 0
	OP cod	Operand1	Operand2	Operand3
	5 bit	3 bit	4 bit	4 bit
		r1	r2	r3
			val2(4bit)	val3(4bit)

mnemonic	operand1	operand2	operand3	op code	operation
NOP *				00000	no operation
HALT *				00001	halt
LOAD *	r1	r2	val3	00010	gr[r1]<-m[r2+val3]
STORE *	r1	r2	val3	00011	m[r2+val3]<-r1
LDIH	r1	val2	val3	10000	r1<-r1+{val2, val3, 0000_0000} (lower 8’ b0 can be given with ADDI)
ADD *	r1	r2	r3	01000	r1<-r2+r3
ADDI	r1	val2	val3	01001	r1<-r1+{val2, val3}
ADDC	r1	r2	r3	10001	r1<-r2+r3+CF
SUB	r1	r2	r3	10010	r1<-r2-r3
SUBI	r1	val2	val3	10011	r1<-r1-{val2, val3}
SUBC	r1	r2	r3	10100	r1<-r2-r3-CF
INC	r1	r2		10101	r1<-r2+1
DEC	r1	r2		10110	r1<-r2-1
JUMP		val2	val3	11000	jump to {val2, val3}

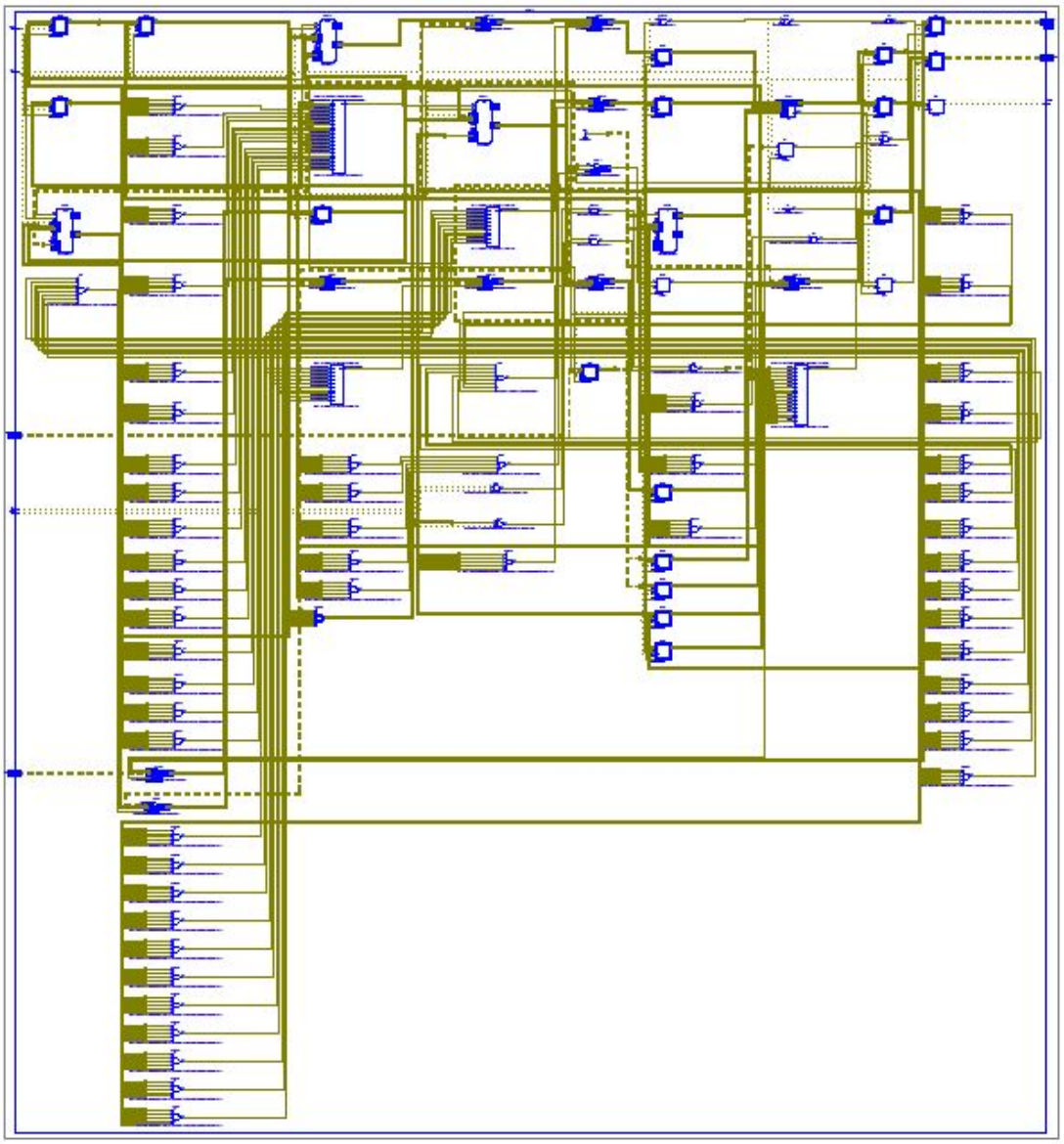
JMPR	r1	val2	val3	11001	jump to r1+{val2, val3}
BZ *	r1	val2	val3	11010	if ZF=1 branch to r1+{val2, val3}
BNZ	r1	val2	val3	11011	if ZF=0 branch to r1+{val2, val3}
BN *	r1	val2	val3	11100	if NF=1 branch to r1+{val2, val3}
BNN	r1	val2	val3	11101	if NF=0 branch to r1+{val2, val3}
BC	r1	val2	val3	11110	if CF=1 branch to r1+{val2, val3}
BNC	r1	val2	val3	11111	if CF=0 branch to r1+{val2, val3}
CMP *		r2	r3	01100	r2-r3; set CF,ZF and NF
AND	r1	r2	r3	01101	r1<-r2 and r3
OR	r1	r2	r3	01110	r1<-r2 or r3
NOT	r1	r2		10111	r1<- not r2
XOR	r1	r2	r3	01111	r1<-r2 xor r3
SLL	r1	r2	val3	00100	r1<-r2 shift left logical (val3 bit shift)
SRL	r1	r2	val3	00110	r1<-r2 shift right logical (val3 bit shift)
SLA	r1	r2	val3	00101	r1<-r2 shift left arithmetical (val3 bit shift)
SRA	r1	r2	val3	00111	r1<-r2 shift right arithmetical (val3 bit shift)
SLR	r1	r2	val3	01010	r1<-r2 Cyclic left shift (val3 bit shift)
SRR	r1	r2	val3	01011	r1<-r2 Cyclic right shift (val3 bit shift)

（2）RTL 电路图

总体概要电路图为：



内部结果电路图为：

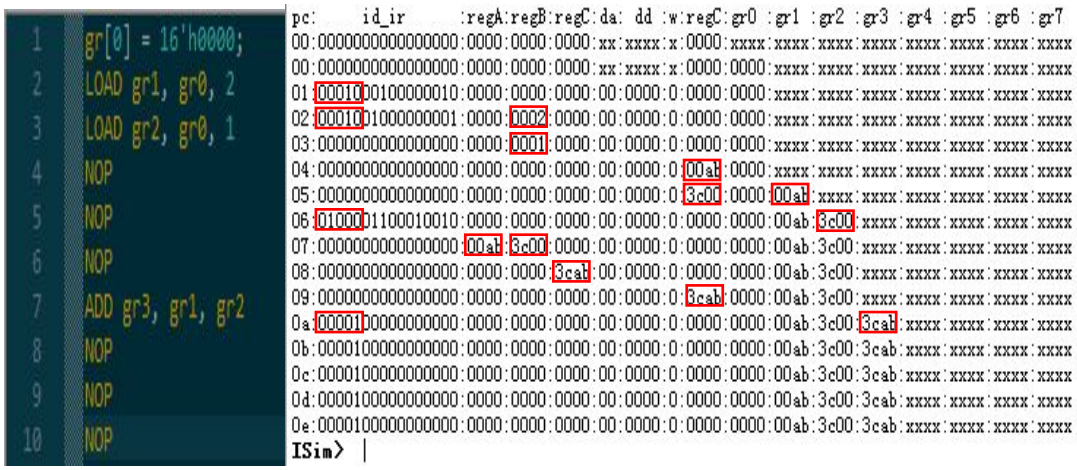


（3）模拟运行验证

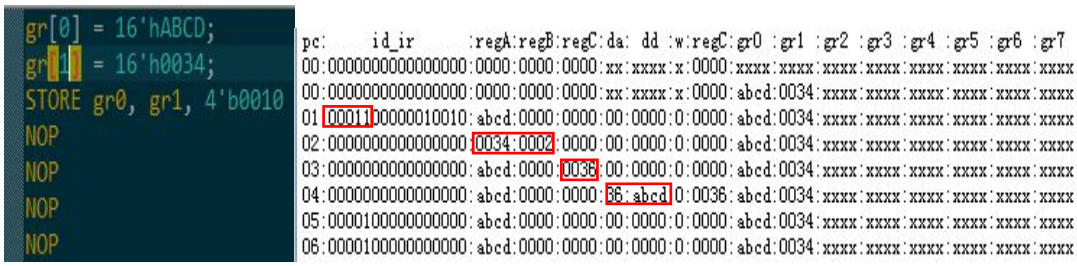
验证每一条指令的测试文件文件均上传 FTP，以下只列出验证结果指令验证：左端为验证伪代码，右端为模拟运行结果

NOP，HALT，ADD 指令验证

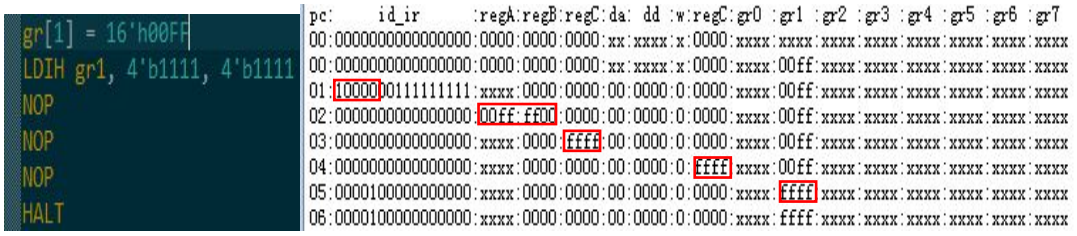
LOAD 指令能够正确将数值加载，并且 ADD 指令能够正确将寄存器中的数值相加并赋值到指定寄存器 gr3。验证正确。



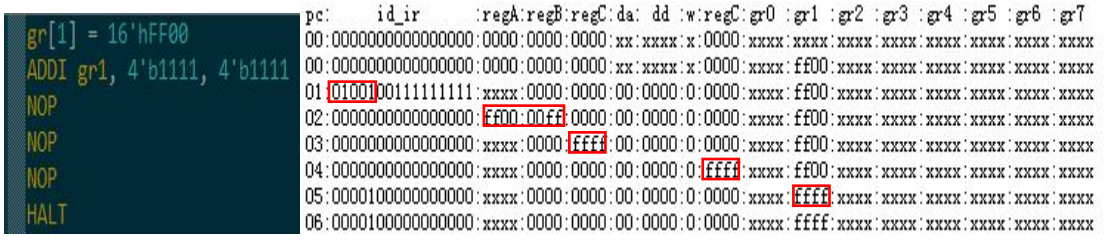
STORE 指令验证：正确将 gr0 中的值赋给 m[gr1+val3]



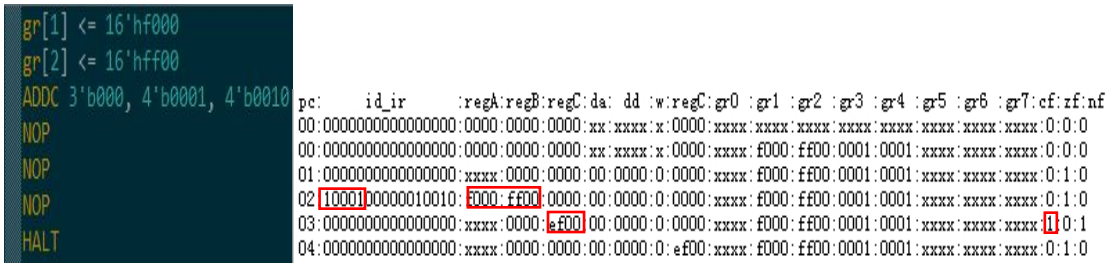
LDIH 指令验证：正确执行 gr1<-gr1+{val2, val3, 0000_0000}



ADDI 指令验证：正确执行 r1<-r1+{val2, val3}



ADDC 指令验证：正确执行 r1<-r2+r3+CF



SUB 指令验证：正确执行 r1<-r2-r3

```
gr[1] = 16'h2222
gr[2] = 16'h1111
SUB gr0, 4'b0001, 4'b0010
NOP
NOP
NOP
HALT
```

```
pc:     id_ir      :regA:regB:regC:da: dd :w:regC:gr0 :gr1 :gr2 :gr3 :gr4 :gr5 :gr6 :gr7:cf:zf:nf
00:0000000000000000:0000:0000:0000:xx:xxxx:x:0000:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:0:0:0
00:0000000000000000:0000:0000:0000:xx:xxxx:x:0000:xxxx:2222:1111:xxxx:xxxx:xxxx:xxxx:xxxx:0:0:0
01:1001000000010010:xxxx:0000:0000:00:0000:0:0000:xxxx:2222:1111:xxxx:xxxx:xxxx:xxxx:xxxx:0:1:0
02:0000000000000000:2222:1111:0000:00:0000:0:0000:xxxx:2222:1111:xxxx:xxxx:xxxx:xxxx:xxxx:0:1:0
03:0000000000000000:xxxx:0000:1111:00:0000:0:0000:xxxx:2222:1111:xxxx:xxxx:xxxx:xxxx:xxxx:0:1:0
04:0000000000000000:xxxx:0000:0000:00:0000:0:1111:xxxx:2222:1111:xxxx:xxxx:xxxx:xxxx:xxxx:0:0:0
05:0000100000000000:xxxx:0000:0000:00:0000:0:0000:1111:2222:1111:xxxx:xxxx:xxxx:xxxx:xxxx:0:1:0
06:0000100000000000:1111:0000:0000:00:0000:0:0000:1111:2222:1111:xxxx:xxxx:xxxx:xxxx:xxxx:0:1:0
```

SUBI 指令验证：正确执行 r1<-r1-{val2, val3}

```
gr[1] = 16'h2222
SUBI gr1, 4'b0010, 4'b0010
NOP
NOP
NOP
HALT
```

```
pc:     id_ir      :regA:regB:regC:da: dd :w:regC:gr0 :gr1 :gr2 :gr3 :gr4 :gr5 :gr6 :gr7:cf:zf:nf
00:0000000000000000:0000:0000:0000:xx:xxxx:x:0000:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:0:0:0
00:0000000000000000:0000:0000:0000:xx:xxxx:x:0000:xxxx:2222:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:0:0:0
01:1001100100100010:xxxx:0000:0000:00:0000:0:0000:xxxx:2222:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:0:1:0
02:0000000000000000:2222:0022:0000:00:0000:0:0000:xxxx:2222:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:0:1:0
03:0000000000000000:xxxx:0000:2200:00:0000:0:0000:xxxx:2222:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:0:1:0
04:0000000000000000:xxxx:0000:0000:00:0000:0:2200:xxxx:2222:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:0:0:0
05:0000100000000000:xxxx:0000:0000:00:0000:0:0000:xxxx:2200:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:0:1:0
06:0000100000000000:xxxx:0000:0000:00:0000:0:0000:xxxx:2200:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:0:1:0
```

SUBC 指令验证：正确执行 r1<-r2-r3-CF

```
gr[1] <= 16'h0080
gr[2] <= 16'h0082
SUBC 3'b000, 4'b0001, 4'b0010
NOP
NOP
NOP
HALT
```

```
pc:     id_ir      :regA:regB:regC:da: dd :w:regC:gr0 :gr1 :gr2 :gr3 :gr4 :gr5 :gr6 :gr7:cf:zf:nf
00:0000000000000000:0000:0000:0000:xx:xxxx:x:0000:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:0:0:0
00:0000000000000000:0000:0000:0000:xx:xxxx:x:0000:xxxx:0080:0082:xxxx:xxxx:xxxx:xxxx:xxxx:0:0:0
01:0000000000000000:xxxx:0000:0000:00:0000:0:0000:xxxx:0080:0082:xxxx:xxxx:xxxx:xxxx:xxxx:0:1:0
02:1010000000010010:0080:0082:0000:00:0000:0:0000:xxxx:0080:0082:xxxx:xxxx:xxxx:xxxx:xxxx:0:1:0
03:0000000000000000:xxxx:0000:fffe:00:0000:0:0000:xxxx:0080:0082:xxxx:xxxx:xxxx:xxxx:xxxx:1:0:1
04:0000000000000000:xxxx:0000:0000:00:0000:0:fffe:xxxx:0080:0082:xxxx:xxxx:xxxx:xxxx:xxxx:0:1:0
05:0000000000000000:xxxx:0000:0000:00:0000:0:0000:fffe:0080:0082:xxxx:xxxx:xxxx:xxxx:xxxx:0:1:0
```

INC 指令验证：正确执行 r1<-r2+1

```
gr[2] = 16'h2222
INC gr1, 4'b0010, 4'b0000
NOP
NOP
NOP
HALT
```

```
pc:     id_ir      :regA:regB:regC:da: dd :w:regC:gr0 :gr1 :gr2 :gr3 :gr4 :gr5 :gr6 :gr7:cf:zf:nf
00:0000000000000000:0000:0000:0000:xx:xxxx:x:0000:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:0:0:0
00:0000000000000000:0000:0000:0000:xx:xxxx:x:0000:xxxx:xxxx:2221:xxxx:xxxx:xxxx:xxxx:xxxx:0:0:0
01:1010100100100000:xxxx:0000:0000:00:0000:0:0000:xxxx:xxxx:2221:xxxx:xxxx:xxxx:xxxx:xxxx:0:1:0
02:0000000000000000:2221:0000:0000:00:0000:0:0000:xxxx:xxxx:2221:xxxx:xxxx:xxxx:xxxx:xxxx:0:1:0
03:0000000000000000:xxxx:0000:2222:00:0000:0:0000:xxxx:xxxx:2221:xxxx:xxxx:xxxx:xxxx:xxxx:0:1:0
04:0000000000000000:xxxx:0000:0000:00:0000:0:2222:xxxx:xxxx:2221:xxxx:xxxx:xxxx:xxxx:xxxx:0:0:0
05:0000100000000000:xxxx:0000:0000:00:0000:0:0000:xxxx:2222:2221:xxxx:xxxx:xxxx:xxxx:xxxx:0:1:0
06:0000100000000000:xxxx:0000:0000:00:0000:0:0000:xxxx:2222:2221:xxxx:xxxx:xxxx:xxxx:xxxx:0:1:0
```

DEC 指令验证：正确执行 r1<-r2-1

```
gr[2] = 16'h2223
DEC gr1, 4'b0010, 4'b0000
NOP
NOP
NOP
HALT
```

```
pc:     id_ir      :regA:regB:regC:da: dd :w:regC:gr0 :gr1 :gr2 :gr3 :gr4 :gr5 :gr6 :gr7:cf:zf:nf
00:0000000000000000:0000:0000:0000:xx:xxxx:x:0000:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:0:0:0
00:0000000000000000:0000:0000:0000:xx:xxxx:x:0000:xxxx:xxxx:2223:xxxx:xxxx:xxxx:xxxx:xxxx:0:0:0
01:1011000100100000:xxxx:0000:0000:00:0000:0:0000:xxxx:xxxx:2223:xxxx:xxxx:xxxx:xxxx:xxxx:0:1:0
02:0000000000000000:2223:0000:0000:00:0000:0:0000:xxxx:xxxx:2223:xxxx:xxxx:xxxx:xxxx:xxxx:0:1:0
03:0000000000000000:xxxx:0000:2222:00:0000:0:0000:xxxx:xxxx:2223:xxxx:xxxx:xxxx:xxxx:xxxx:0:1:0
04:0000000000000000:xxxx:0000:0000:00:0000:0:2222:xxxx:xxxx:2223:xxxx:xxxx:xxxx:xxxx:xxxx:0:0:0
05:0000100000000000:xxxx:0000:0000:00:0000:0:0000:xxxx:2222:2223:xxxx:xxxx:xxxx:xxxx:xxxx:0:1:0
06:0000100000000000:xxxx:0000:0000:00:0000:0:0000:xxxx:2222:2223:xxxx:xxxx:xxxx:xxxx:xxxx:0:1:0
```

JUMP 指令验证：正确执行 jump to {val2, val3}

```
JUMP 3'b000, 4'b0010, 4'b0001
NOP
NOP
NOP
HALT
```

```
pc:     id_ir      :regA:regB:regC:da: dd :w:regC:gr0 :gr1 :gr2 :gr3 :gr4 :gr5 :gr6 :gr7:cf:zf:nf
00:0000000000000000:0000:0000:0000:xx:xxxx:x:0000:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:0:0:0
01:1100000000100001:xxxx:0000:0000:00:0000:0:0000:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:0:1:0
02:0000000000000000:0000:0021:0000:00:0000:0:0000:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:0:1:0
03:0000000000000000:xxxx:0000:0021:00:0000:0:0000:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:0:1:0
21:0000000000000000:xxxx:0000:0000:00:0000:0:0021:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:0:0:0
```

JMPR 指令验证：正确执行 jump to r1+{val2, val3}

```
gr[1] = 16'h0001
JMPR 3'b001, 4'b0010, 4'b0001
NOP
NOP
NOP
HALT
```

```
pc:     id_ir      :regA:regB:regC:da: dd :w:regC:gr0 :gr1 :gr2 :gr3 :gr4 :gr5 :gr6 :gr7:cf:zf:nf
00:0000000000000000:0000:0000:0000:xx:xxxx:x:0000:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:0:0:0
00:0000000000000000:0000:0000:0000:xx:xxxx:x:0000:xxxx:0001:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:0:0:0
01:1100100100100001:xxxx:0000:0000:00:0000:0:0000:xxxx:0001:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:0:1:0
02:0000000000000000:0001:0021:0000:00:0000:0:0000:xxxx:0001:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:0:1:0
03:0000000000000000:xxxx:0000:0022:00:0000:0:0000:xxxx:0001:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:0:1:0
22:0000000000000000:xxxx:0000:0000:00:0000:0:0022:xxxx:0001:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:0:0:0
23:0000100000000000:xxxx:0000:0000:00:0000:0:0000:xxxx:0001:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:0:1:0
```

BZ 指令验证：正确执行 if ZF=1 branch to r1+{val2, val3}

```
gr[1] = 16'h0000
BZ 3'b001, 4'b0000, 4'b0000
NOP
NOP
NOP
HALT
```

```
pc:     id_ir      :regA:regB:regC:da: dd :w:regC:gr0 :gr1 :gr2 :gr3 :gr4 :gr5 :gr6 :gr7:cf:zf:nf
00:0000000000000000:0000:0000:0000:xx:xxxx:x:0000:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:0:0:0
00:0000000000000000:0000:0000:0000:xx:xxxx:x:0000:xxxx:0000:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:0:0:0
01:0000000000000000:xxxx:0000:0000:00:0000:0:0000:xxxx:0000:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:0:1:0
02:0000000000000000:0000:0000:0000:00:0000:0:0000:xxxx:0000:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:0:1:0
03:1101000100000000:xxxx:0000:0000:00:0000:0:0000:xxxx:0000:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:0:1:0
00:0000000000000000:xxxx:0000:0000:00:0000:0:0000:xxxx:0000:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:0:1:0
01:0000000000000000:xxxx:0000:0000:00:0000:0:0000:xxxx:0000:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:0:1:0
```

BNZ 指令验证：正确执行 if ZF=0 branch to r1+{val2, val3}

```
gr[1] = 16'h0002
BNZ 3'b001, 4'b0010, 4'b0001
NOP
NOP
NOP
HALT
```

```
pc:     id_ir      :regA:regB:regC:da: dd :w:regC:gr0 :gr1 :gr2 :gr3 :gr4 :gr5 :gr6 :gr7:cf:zf:nf
00:0000000000000000:0000:0000:0000:xx:xxxx:x:0000:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:0:0:0
00:0000000000000000:0000:0000:0000:xx:xxxx:x:0000:xxxx:0002:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:0:0:0
01:0000000000000000:xxxx:0000:0000:00:0000:0:0000:xxxx:0002:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:0:1:0
02:0000000000000000:0002:0021:0000:00:0000:0:0000:xxxx:0002:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:0:1:0
03:1101100100100001:xxxx:0000:0023:00:0000:0:0000:xxxx:0002:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:0:0:0
23:0000000000000000:xxxx:0000:0000:00:0000:0:0023:xxxx:0002:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:0:1:0
24:0000000000000000:xxxx:0000:0000:00:0000:0:0000:xxxx:0002:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:0:1:0
```

BN 指令验证：正确执行 if NF=1 branch to r1+{val2, val3}

```
gr[1] = 16'hf012
BN 3'b001, 4'b0000, 4'b0000
NOP
NOP
NOP
HALT
```

```
pc:     id_ir      :regA:regB:regC:da: dd :w:regC:gr0 :gr1 :gr2 :gr3 :gr4 :gr5 :gr6 :gr7:cf:zf:nf
00:0000000000000000:0000:0000:0000:xx:xxxx:x:0000:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:0:0:0
00:0000000000000000:0000:0000:0000:xx:xxxx:x:0000:xxxx:f012:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:0:0:0
01:0000000000000000:xxxx:0000:0000:00:0000:0:0000:xxxx:f012:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:0:1:0
02:0000000000000000:f012:0000:0000:00:0000:0:0000:xxxx:f012:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:0:1:0
03:1110000100000000:xxxx:0000:f012:00:0000:0:0000:xxxx:f012:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:0:0:1
12:0000000000000000:xxxx:0000:0000:00:0000:0:f012:xxxx:f012:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:0:1:0
13:0000000000000000:xxxx:0000:0000:00:0000:0:0000:xxxx:f012:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:0:1:0
```

BNN 指令验证：正确执行 if NF=0 branch to r1+{val2, val3}

```
gr[1] = 16'h0012
BNN 3'b001, 4'b0000, 4'b0000
NOP
NOP
NOP
HALT
```

```
pc:     id_ir      :regA:regB:regC:da: dd :w:regC:gr0 :gr1 :gr2 :gr3 :gr4 :gr5 :gr6 :gr7:cf:zf:nf
00:0000000000000000:0000:0000:0000:xx:xxxx:x:0000:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:0:0:0
00:0000000000000000:0000:0000:0000:xx:xxxx:x:0000:xxxx:0012:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:0:0:0
01:0000000000000000:xxxx:0000:0000:00:0000:0:0000:xxxx:0012:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:0:1:0
02:0000000000000000:0012:0000:0000:00:0000:0:0000:xxxx:0012:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:0:1:0
03:1110100100000000:xxxx:0000:0012:00:0000:0:0000:xxxx:0012:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:0:0:1
12:0000000000000000:xxxx:0000:0000:00:0000:0:0012:xxxx:0012:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:0:1:0
13:0000000000000000:xxxx:0000:0000:00:0000:0:0000:xxxx:0012:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:0:1:0
```

BC 指令验证：正确执行 if CF=1 branch to r1+{val2, val3}

```
gr[1] <= 16'hff80
BC 3'b001, 4'b1000, 4'b0010
NOP
NOP
NOP
HALT
```

```
pc:     id_ir      :regA:regB:regC:da: dd :w:regC:gr0 :gr1 :gr2 :gr3 :gr4 :gr5 :gr6 :gr7:cf:zf:nf
00:0000000000000000:0000:0000:0000:xx:xxxx:x:0000:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:0:0:0
00:0000000000000000:0000:0000:0000:xx:xxxx:x:0000:xxxx:ff80:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:0:0:0
01:0000000000000000:xxxx:0000:0000:00:0000:0:0000:xxxx:ff80:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:0:1:0
02:1111000110000010:ff80:0082:0000:00:0000:0:0000:xxxx:ff80:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:0:1:0
03:0000000000000000:xxxx:0000:0002:00:0000:0:0000:xxxx:ff80:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:1:0:0
02:0000000000000000:xxxx:0000:0000:00:0000:0:0002:xxxx:ff80:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:0:1:0
```

BNC 指令验证：正确执行 if CF=0 branch to r1+{val2, val3}

```
gr[1] <= 16'h0080
BNC 3'b001, 4'b0000, 4'b0010
NOP
NOP
NOP
HALT
```

```
pc:      id_ir       :regA:regB:regC:da:  dd :w:regC:gr0 :gr1 :gr2 :gr3 :gr4 :gr5 :gr6 :gr7:cf:zf:nf
00:0000000000000000:0000:0000:0000:xx:xxxx:x:0000:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:0:0:0
00:0000000000000000:0000:0000:0000:xx:xxxx:x:0000:xxxx:0080:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:0:0:0
01:0000000000000000:xxxx:0000:0000:00:0000:0:0000:xxxx:0080:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:0:1:0
02:11111:0100000010:0080:0002:0000:00:0000:0:0000:xxxx:0080:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:0:1:0
03:0000000000000000:xxxx:0000:0082:00:0000:0:0000:xxxx:0080:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:0:0:0
82:0000000000000000:xxxx:0000:0000:00:0000:0:0082:xxxx:0080:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:0:1:0
83:0000000000000000:xxxx:0000:0000:00:0000:0:0000:xxxx:0080:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:0:1:0
```

CMP 指令验证：正确执行 r2-r3; set CF,ZF and NF

```
gr[1] <= 16'h0080
gr[2] <= 16'h0082
CMP 3'b000, 4'b0001, 4'b0010
NOP
NOP
NOP
HALT
```

```
pc:      id_ir       :regA:regB:regC:da:  dd :w:regC:gr0 :gr1 :gr2 :gr3 :gr4 :gr5 :gr6 :gr7:cf:zf:nf
00:0000000000000000:0000:0000:0000:xx:xxxx:x:0000:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:0:0:0
00:0000000000000000:0000:0000:0000:xx:xxxx:x:0000:xxxx:0080:0082:xxxx:xxxx:xxxx:xxxx:xxxx:0:0:0
01:0000000000000000:xxxx:0000:0000:00:0000:0:0000:xxxx:0080:0082:xxxx:xxxx:xxxx:xxxx:xxxx:0:1:0
02:01100:0000010010:0080:0082:0000:00:0000:0:0000:xxxx:0080:0082:xxxx:xxxx:xxxx:xxxx:xxxx:0:1:0
03:0000000000000000:xxxx:0000:0000:00:0000:0:0000:xxxx:0080:0082:xxxx:xxxx:xxxx:xxxx:xxxx:1:1:0
04:0000000000000000:xxxx:0000:0000:00:0000:0:0000:xxxx:0080:0082:xxxx:xxxx:xxxx:xxxx:xxxx:0:1:0
05:0000000000000000:xxxx:0000:0000:00:0000:0:0000:xxxx:0080:0082:xxxx:xxxx:xxxx:xxxx:xxxx:0:1:0
```

AND 指令验证：正确执行 r1<-r2 and r3

```
gr[1] = 16'h0012
gr[2] = 16'h0012
AND 3'b000, 4'b0001, 4'b0010
NOP
NOP
NOP
HALT
```

```
pc:      id_ir       :regA:regB:regC:da:  dd :w:regC:gr0 :gr1 :gr2 :gr3 :gr4 :gr5 :gr6 :gr7:cf:zf:nf
00:0000000000000000:0000:0000:0000:xx:xxxx:x:0000:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:0:0:0
00:0000000000000000:0000:0000:0000:xx:xxxx:x:0000:xxxx:0012:0012:xxxx:xxxx:xxxx:xxxx:xxxx:0:0:0
01:0000000000000000:xxxx:0000:0000:00:0000:0:0000:xxxx:0012:0012:xxxx:xxxx:xxxx:xxxx:xxxx:0:1:0
02:0000000000000000:0012:0012:0000:00:0000:0:0000:xxxx:0012:0012:xxxx:xxxx:xxxx:xxxx:xxxx:0:1:0
03:01101:0000010010:xxxx:0000:0012:00:0000:0:0000:xxxx:0012:0012:xxxx:xxxx:xxxx:xxxx:xxxx:0:0:0
04:0000000000000000:xxxx:0000:0000:00:0000:0:0012:xxxx:0012:0012:xxxx:xxxx:xxxx:xxxx:xxxx:0:1:0
05:0000000000000000:xxxx:0000:0000:00:0000:0:0000:0012:0012:0012:xxxx:xxxx:xxxx:xxxx:xxxx:0:1:0
06:0000000000000000:0012:0000:0000:00:0000:0:0000:0012:0012:0012:xxxx:xxxx:xxxx:xxxx:xxxx:0:1:0
```

OR 指令验证：正确执行 r1<-r2 or r3

```
gr[1] = 16'h0012
gr[2] = 16'h1200
OR 3'b000, 4'b0001, 4'b0010
NOP
NOP
NOP
HALT
```

```
pc:      id_ir       :regA:regB:regC:da:  dd :w:regC:gr0 :gr1 :gr2 :gr3 :gr4 :gr5 :gr6 :gr7:cf:zf:nf
00:0000000000000000:0000:0000:0000:xx:xxxx:x:0000:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:0:0:0
00:0000000000000000:0000:0000:0000:xx:xxxx:x:0000:xxxx:0012:1200:xxxx:xxxx:xxxx:xxxx:xxxx:0:0:0
01:0000000000000000:xxxx:0000:0000:00:0000:0:0000:xxxx:0012:1200:xxxx:xxxx:xxxx:xxxx:xxxx:0:1:0
02:0000000000000000:0012:1200:0000:00:0000:0:0000:xxxx:0012:1200:xxxx:xxxx:xxxx:xxxx:xxxx:0:1:0
03:01110:0000010010:xxxx:0000:1212:00:0000:0:0000:xxxx:0012:1200:xxxx:xxxx:xxxx:xxxx:xxxx:0:0:0
04:0000000000000000:xxxx:0000:0000:00:0000:0:1212:xxxx:0012:1200:xxxx:xxxx:xxxx:xxxx:xxxx:0:1:0
05:0000000000000000:xxxx:0000:0000:00:0000:0:0000:1212:0012:1200:xxxx:xxxx:xxxx:xxxx:xxxx:0:1:0
06:0000000000000000:1212:0000:0000:00:0000:0:0000:1212:0012:1200:xxxx:xxxx:xxxx:xxxx:xxxx:0:1:0
```

NOT 指令验证：正确执行 r1<- not r2

```
gr[1] = 16'hff00
NOT 3'b000, 4'b0001, 4'b0000
NOP
NOP
NOP
HALT
```

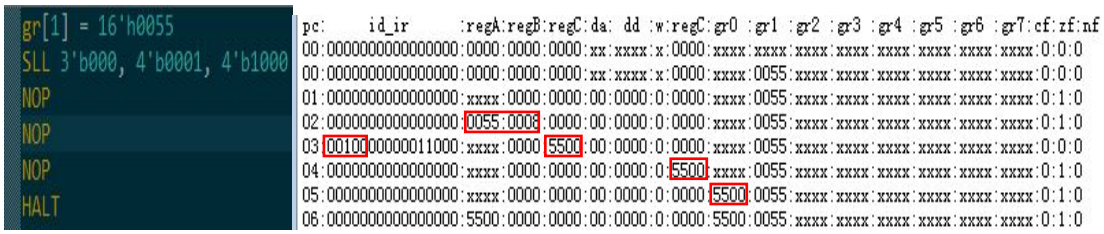
```
pc:      id_ir       :regA:regB:regC:da:  dd :w:regC:gr0 :gr1 :gr2 :gr3 :gr4 :gr5 :gr6 :gr7:cf:zf:nf
00:0000000000000000:0000:0000:0000:xx:xxxx:x:0000:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:0:0:0
00:0000000000000000:0000:0000:0000:xx:xxxx:x:0000:xxxx:ff00:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:0:0:0
01:0000000000000000:xxxx:0000:0000:00:0000:0:0000:xxxx:ff00:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:0:1:0
02:0000000000000000:ff00:0000:0000:00:0000:0:0000:xxxx:ff00:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:0:1:0
03:10111:0000010000:xxxx:0000:00ff:00:0000:0:0000:xxxx:ff00:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:0:0:0
04:0000000000000000:xxxx:0000:0000:00:0000:0:00ff:xxxx:ff00:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:0:1:0
05:0000000000000000:xxxx:0000:0000:00:0000:0:0000:00ff:ff00:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:0:1:0
06:0000000000000000:00ff:0000:0000:00:0000:0:0000:00ff:ff00:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:0:1:0
```

XOR 指令验证：正确执行 r1<-r2 xor r3

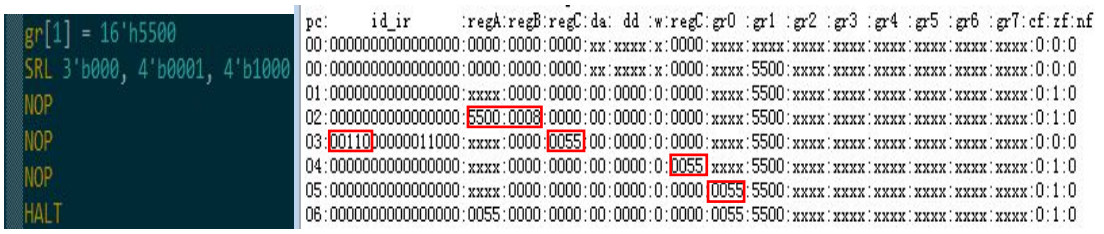
```
gr[1] = 16'h550f
gr[2] = 16'haa0f
XOR 3'b000, 4'b0001, 4'b0010
NOP
NOP
NOP
HALT
```

```
pc:      id_ir       :regA:regB:regC:da:  dd :w:regC:gr0 :gr1 :gr2 :gr3 :gr4 :gr5 :gr6 :gr7:cf:zf:nf
00:0000000000000000:0000:0000:0000:xx:xxxx:x:0000:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:0:0:0
00:0000000000000000:0000:0000:0000:xx:xxxx:x:0000:xxxx:550f:aa0f:xxxx:xxxx:xxxx:xxxx:xxxx:0:0:0
01:0000000000000000:xxxx:0000:0000:00:0000:0:0000:xxxx:550f:aa0f:xxxx:xxxx:xxxx:xxxx:xxxx:0:1:0
02:0000000000000000:550f:aa0f:0000:00:0000:0:0000:xxxx:550f:aa0f:xxxx:xxxx:xxxx:xxxx:xxxx:0:1:0
03:01111:0000010010:xxxx:0000:ff00:00:0000:0:0000:xxxx:550f:aa0f:xxxx:xxxx:xxxx:xxxx:xxxx:0:0:1
04:0000000000000000:xxxx:0000:0000:00:0000:0:ff00:xxxx:550f:aa0f:xxxx:xxxx:xxxx:xxxx:xxxx:0:1:0
05:0000000000000000:xxxx:0000:0000:00:0000:0:0000:ff00:550f:aa0f:xxxx:xxxx:xxxx:xxxx:xxxx:0:1:0
06:0000000000000000:ff00:0000:0000:00:0000:0:0000:ff00:550f:aa0f:xxxx:xxxx:xxxx:xxxx:xxxx:0:1:0
```

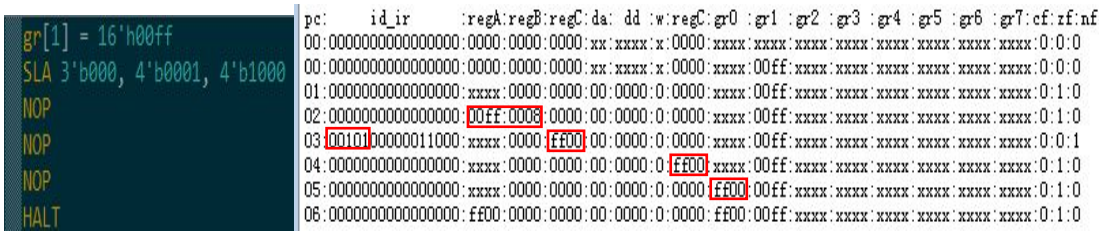
SLL 指令验证：正确执行 r1<-r2 shift left logical (val3 bit shift)



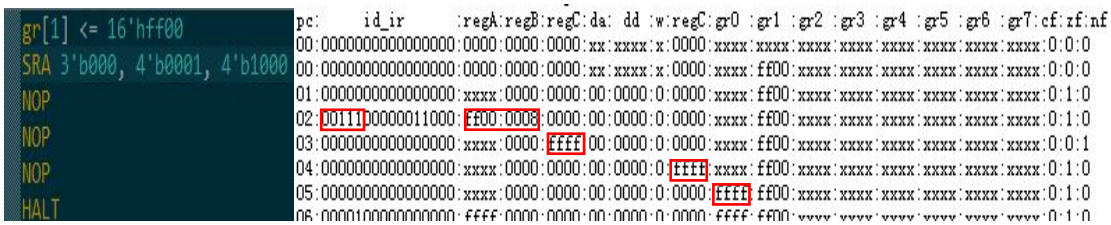
SRL 指令验证：正确执行 r1<-r2 shift right logical (val3 bit shift)



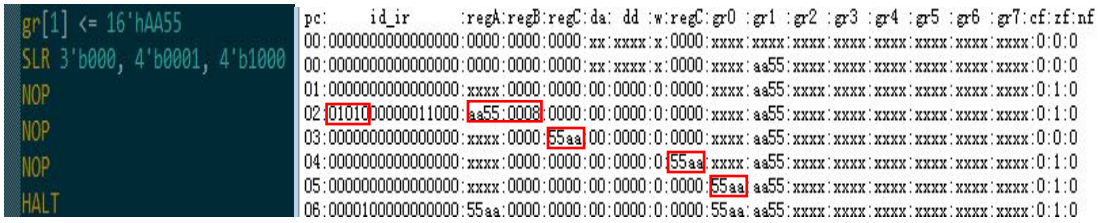
SLA 指令验证：正确执行 r1<-r2 shift left arithmetical (val3 bit shift)



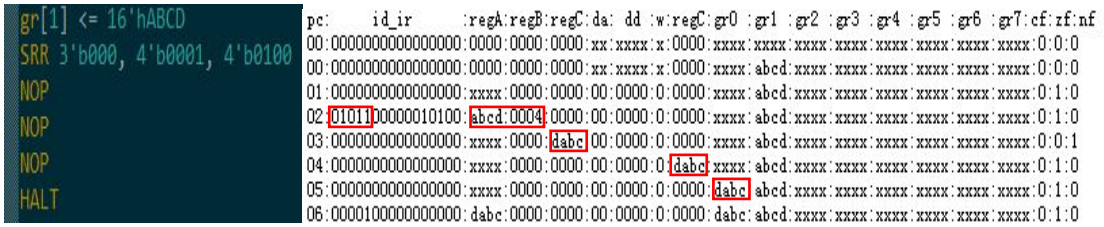
SRA 指令验证：正确执行 r1<-r2 shift right arithmetical (val3 bit shift)



SLR 指令验证：正确执行 r1<-r2 Cyclic left shift (val3 bit shift)



SRR 指令验证：正确执行 r1<-r2 Cyclic right shift (val3 bit shift)



总结：综上，全部设计的 32 条指令均能够正确的执行。

四、性能分析

（1）面积消耗

```
Slice Logic Utilization:
 Number of Slice Registers:              200  out of  18224      1%
 Number of Slice LUTs:                   640  out of   9112      7%
```

从以上数据可以知道，本次实验共使用寄存器 200 个，LUT 单元 640 个。

（2）功耗情况

```
--------------------------------------------------------------------------------
|                          On-Chip Power Summary                               |
--------------------------------------------------------------------------------
|        On-Chip        | Power (mW) |  Used  | Available | Utilization (%) |
--------------------------------------------------------------------------------
| Clocks                |       0.02 |      3 |    ---    |       ---       |
| Logic                 |       0.00 |    576 |      9112 |               6 |
| Signals               |       0.00 |    723 |    ---    |       ---       |
| IOs                   |       0.00 |     61 |       232 |              26 |
| Quiescent             |      14.84 |        |           |                 |
| Total                 |      14.87 |        |           |                 |
--------------------------------------------------------------------------------
```

从功耗的情况来看，本次实验所用总功耗为 14.87mW。

（3）时延情况

Clock clock to Pad：

Clock clock to Pad

Destination	Max (slowest) clk (edge) to PAD	Process Corner	Min (fastest) clk (edge) to PAD	Process Corner	Internal Clock(s)	Clock Phase
d_addr<0>	7.843(R)	SLOW	3.729(R)	FAST	clock_BUFGP	0.000
d_addr<1>	8.224(R)	SLOW	3.927(R)	FAST	clock_BUFGP	0.000
d_addr<2>	7.919(R)	SLOW	3.790(R)	FAST	clock_BUFGP	0.000
d_addr<3>	7.880(R)	SLOW	3.783(R)	FAST	clock_BUFGP	0.000
d_addr<4>	8.112(R)	SLOW	3.878(R)	FAST	clock_BUFGP	0.000
d_addr<5>	8.051(R)	SLOW	3.854(R)	FAST	clock_BUFGP	0.000
d_addr<6>	7.654(R)	SLOW	3.628(R)	FAST	clock_BUFGP	0.000
d_addr<7>	7.884(R)	SLOW	3.735(R)	FAST	clock_BUFGP	0.000
d_dataout<0>	7.199(R)	SLOW	3.320(R)	FAST	clock_BUFGP	0.000
d_dataout<1>	7.303(R)	SLOW	3.388(R)	FAST	clock_BUFGP	0.000
d_dataout<2>	7.422(R)	SLOW	3.472(R)	FAST	clock_BUFGP	0.000
d_dataout<3>	7.552(R)	SLOW	3.524(R)	FAST	clock_BUFGP	0.000
d_dataout<4>	7.670(R)	SLOW	3.607(R)	FAST	clock_BUFGP	0.000
d_dataout<5>	7.489(R)	SLOW	3.473(R)	FAST	clock_BUFGP	0.000
d_dataout<6>	7.449(R)	SLOW	3.438(R)	FAST	clock_BUFGP	0.000
d_dataout<7>	7.571(R)	SLOW	3.551(R)	FAST	clock_BUFGP	0.000
d_dataout<8>	8.021(R)	SLOW	3.789(R)	FAST	clock_BUFGP	0.000
d_dataout<9>	7.748(R)	SLOW	3.593(R)	FAST	clock_BUFGP	0.000
d_dataout<10>	7.635(R)	SLOW	3.530(R)	FAST	clock_BUFGP	0.000
d_dataout<11>	7.614(R)	SLOW	3.545(R)	FAST	clock_BUFGP	0.000
d_dataout<12>	7.503(R)	SLOW	3.504(R)	FAST	clock_BUFGP	0.000
d_dataout<13>	7.663(R)	SLOW	3.545(R)	FAST	clock_BUFGP	0.000
d_dataout<14>	7.431(R)	SLOW	3.422(R)	FAST	clock_BUFGP	0.000
d_dataout<15>	7.582(R)	SLOW	3.545(R)	FAST	clock_BUFGP	0.000
d_we	7.424(R)	SLOW	3.484(R)	FAST	clock_BUFGP	0.000

Setup/Hold to clock clock

Source	Max Setup to clk (edge)	Process Corner	Max Hold to clk (edge)	Process Corner	Internal Clock(s)	Clock Phase
d_datain<0>	2.426(R)	SLOW	-1.342(R)	FAST	clock_BUFGP	0.000
d_datain<1>	2.587(R)	SLOW	-1.397(R)	FAST	clock_BUFGP	0.000
d_datain<2>	2.415(R)	SLOW	-1.303(R)	FAST	clock_BUFGP	0.000
d_datain<3>	2.242(R)	SLOW	-1.130(R)	FAST	clock_BUFGP	0.000
d_datain<4>	2.484(R)	SLOW	-1.391(R)	FAST	clock_BUFGP	0.000
d_datain<5>	2.351(R)	SLOW	-1.259(R)	FAST	clock_BUFGP	0.000
d_datain<6>	2.210(R)	SLOW	-1.209(R)	FAST	clock_BUFGP	0.000
d_datain<7>	2.153(R)	SLOW	-1.110(R)	FAST	clock_BUFGP	0.000
d_datain<8>	2.426(R)	SLOW	-1.236(R)	FAST	clock_BUFGP	0.000
d_datain<9>	2.536(R)	SLOW	-1.278(R)	FAST	clock_BUFGP	0.000
d_datain<10>	2.625(R)	SLOW	-1.399(R)	FAST	clock_BUFGP	0.000
d_datain<11>	1.684(R)	SLOW	-0.881(R)	FAST	clock_BUFGP	0.000
d_datain<12>	1.924(R)	SLOW	-0.936(R)	FAST	clock_BUFGP	0.000
d_datain<13>	1.759(R)	SLOW	-0.815(R)	FAST	clock_BUFGP	0.000
d_datain<14>	1.865(R)	SLOW	-0.992(R)	FAST	clock_BUFGP	0.000
d_datain<15>	2.136(R)	SLOW	-1.116(R)	FAST	clock_BUFGP	0.000
enable	2.495(R)	SLOW	-1.005(R)	FAST	clock_BUFGP	0.000
i_datain<0>	1.365(R)	SLOW	-0.627(R)	FAST	clock_BUFGP	0.000
i_datain<1>	1.403(R)	SLOW	-0.693(R)	FAST	clock_BUFGP	0.000
i_datain<2>	1.350(R)	SLOW	-0.668(R)	FAST	clock_BUFGP	0.000
i_datain<3>	1.305(R)	SLOW	-0.584(R)	FAST	clock_BUFGP	0.000
i_datain<4>	1.197(R)	SLOW	-0.519(R)	SLOW	clock_BUFGP	0.000
i_datain<5>	1.080(R)	SLOW	-0.443(R)	SLOW	clock_BUFGP	0.000
i_datain<6>	0.970(R)	SLOW	-0.300(R)	SLOW	clock_BUFGP	0.000
i_datain<7>	1.131(R)	SLOW	-0.491(R)	SLOW	clock_BUFGP	0.000
i_datain<8>	0.853(R)	SLOW	-0.057(R)	SLOW	clock_BUFGP	0.000
i_datain<9>	0.888(R)	SLOW	-0.125(R)	SLOW	clock_BUFGP	0.000
i_datain<10>	0.873(R)	SLOW	-0.079(R)	SLOW	clock_BUFGP	0.000
i_datain<11>	2.274(R)	SLOW	-0.267(R)	SLOW	clock_BUFGP	0.000
i_datain<12>	2.108(R)	SLOW	-0.585(R)	SLOW	clock_BUFGP	0.000
i_datain<13>	1.622(R)	SLOW	-0.439(R)	SLOW	clock_BUFGP	0.000
i_datain<14>	1.774(R)	SLOW	-0.284(R)	SLOW	clock_BUFGP	0.000
i_datain<15>	1.301(R)	SLOW	-0.506(R)	SLOW	clock_BUFGP	0.000
reset	4.030(R)	SLOW	-0.182(R)	SLOW	clock_BUFGP	0.000
start	1.925(R)	SLOW	-0.653(R)	FAST	clock_BUFGP	0.000

五、实验感想

1、对流水线设计的方法及实现方法了解得更加清晰。

2、从底层了解简单 CPU 的工作原理并能够进行简单 CPU 的设计。

3、更加深入的了解操作集的有关知识。

4、提高了编写及调试 verilog 程序的能力。

5、学会在调试环境下利用$display 及$monitor 方法进行中间变量的追踪，提高了调试程序的效率。

6、学会在调试环境下进行设置时钟，之前的均是在 initial 模块中设置，本次实验利用 always 在 initial 模块之外设置。