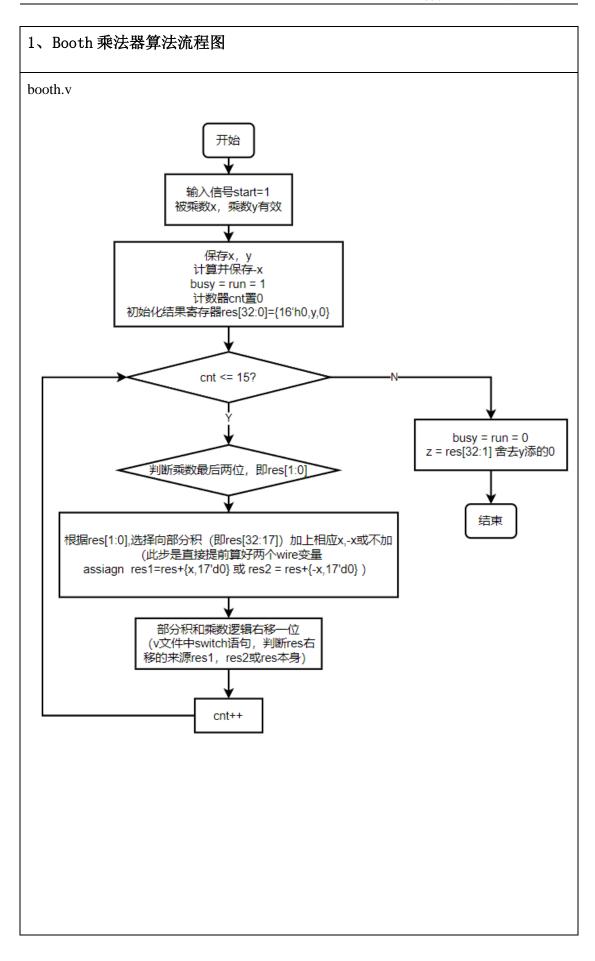


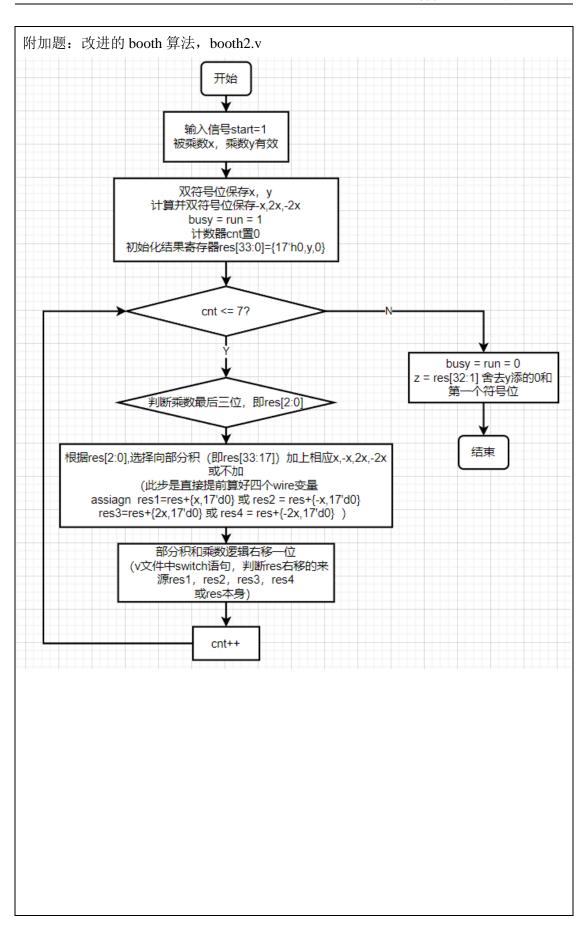
## 哈爾濱Z業大學 (深圳) HARBIN INSTITUTE OF TECHNOLOGY

## 实验作业

开课学期:	2022 春季
课程名称:	计算机组成原理(实验)
实验名称:	Booth 乘法器设计
实验性质:	综合设计型
实验学时:	4 地点: T2507
学生班级:	 20 级 8 班
学生学号:	200210231
学生姓名:	王木一
作业成绩:	

实验与创新实践教育中心制 2022 年 4 月





## 2、调试报告

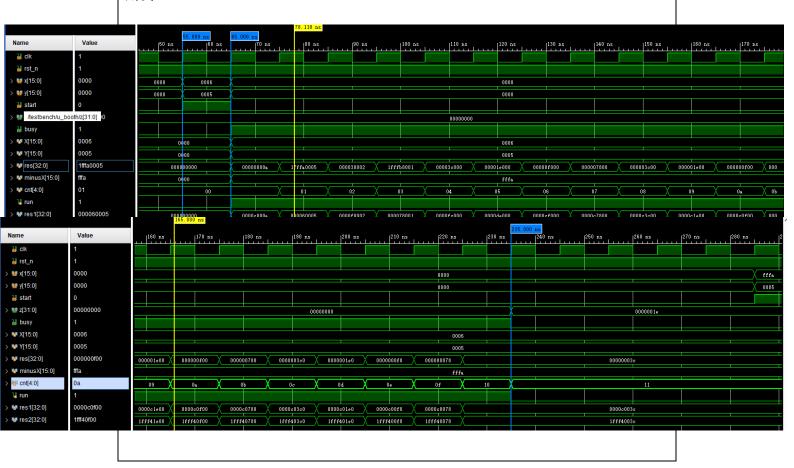
(仿真截图及时序分析,要求分析最少3次乘法运算)

Booth.v

内部信号解释:

信号	功能 (数据以补码表示)
X[15:0]	保存被乘数 x
Y[15:0]	保存乘数 y
minusX[15:0]	保存被乘数-x
cnt[4:0]	计数器
run	同 busy
res[32:0]	其高 32 位为最终结果。初始时高 16 位为 0, 低 17 位为尾部添 0 的乘数 y

用例 1: 0006\*0005 (6\*5)



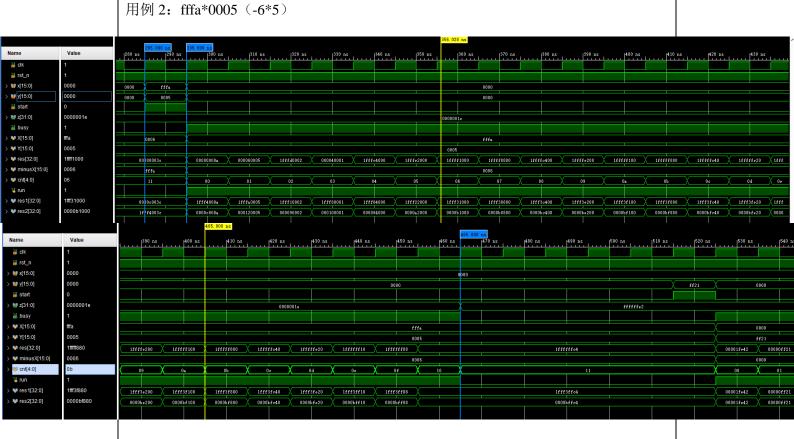
55ns; start=1; x 与 y 分别有信号, 为 0006 和 0005

65ns: start=0, busy=1, 计数器 cnt 从 0 开始计数, 乘法器开始计算。

X[15:0]	0006
Y[15:0]	0005
-X[15:0]	fffa

cnt[4:0]	res[32:0]	res 备注
0	00000000a	res 初始化,高 16 位为 0,低 17 位为尾部添 0 的被乘数
1	1fffa0005	res 最低 2 位为 10, res 高 16 位加-x, 逻辑右移 1 位后的
		结果
2	000030002	res 最低 2 位为 01, res 高 16 位加 x, 逻辑右移 1 位后的
		结果
3	1fffb8001	res 最低 2 位为 10, res 高 16 位加-x, 逻辑右移 1 位后的
		结果
4	00003c000	res 最低 2 位为 01, res 高 16 位加 x, 逻辑右移 1 位后的
		结果
5	00001e000	res 最低 2 位为 00,直接逻辑右移 1 位后的结果
6	00000f000	res 最低 2 位为 00,直接逻辑右移 1 位后的结果
7	000007800	res 最低 2 位为 00,直接逻辑右移 1 位后的结果
8	000003c00	res 最低 2 位为 00,直接逻辑右移 1 位后的结果
9	000001e00	res 最低 2 位为 00,直接逻辑右移 1 位后的结果
a	000000f00	res 最低 2 位为 00,直接逻辑右移 1 位后的结果
b	000000780	res 最低 2 位为 00,直接逻辑右移 1 位后的结果
c	0000003c0	res 最低 2 位为 00,直接逻辑右移 1 位后的结果
d	0000001e0	res 最低 2 位为 00,直接逻辑右移 1 位后的结果
e	0000000f0	res 最低 2 位为 00,直接逻辑右移 1 位后的结果
f	00000078	res 最低 2 位为 00,直接逻辑右移 1 位后的结果
10	00000003c	res 最低 2 位为 00,直接逻辑右移 1 位后的结果

235ns: cnt 计数计到 11,计算结束 busy=0,最终结果 z 截取 res 的高 32 位,即 z=0000001e=30



285ns; start=1; x 与 y 分别有信号, 为 fffa 和 0005

X[15:0]	fffa
Y[15:0]	0005
-X[15:0]	0006

cnt[4:0]	res[32:0]	res 备注
0	00000000a	res 初始化,高 16 位为 0,低 17 位为尾部添 0 的被乘数
1	00006005	res 最低 2 位为 10, res 高 16 位加-x, 逻辑右移 1 位后的
		结果
2	1fffd0002	res 最低 2 位为 01, res 高 16 位加 x, 逻辑右移 1 位后的
		结果
3	000048001	res 最低 2 位为 10, res 高 16 位加-x, 逻辑右移 1 位后的
		结果
4	1fffc4000	res 最低 2 位为 01, res 高 16 位加 x, 逻辑右移 1 位后的
		结果
5	1fffe2000	res 最低 2 位为 00,直接逻辑右移 1 位后的结果
6	1ffff1000	res 最低 2 位为 00,直接逻辑右移 1 位后的结果
7	1ffff8800	res 最低 2 位为 00,直接逻辑右移 1 位后的结果

8	1ffffc400	res 最低 2 位为 00,直接逻辑右移 1 位后的结果
9	1ffffe200	res 最低 2 位为 00,直接逻辑右移 1 位后的结果
a	1fffff100	res 最低 2 位为 00,直接逻辑右移 1 位后的结果
b	1fffff880	res 最低 2 位为 00,直接逻辑右移 1 位后的结果
c	1fffffc40	res 最低 2 位为 00,直接逻辑右移 1 位后的结果
d	1fffffe20	res 最低 2 位为 00,直接逻辑右移 1 位后的结果
e	1ffffff10	res 最低 2 位为 00,直接逻辑右移 1 位后的结果
f	1ffffff88	res 最低 2 位为 00,直接逻辑右移 1 位后的结果
10	1ffffffc4	res 最低 2 位为 00,直接逻辑右移 1 位后的结果

465ns: cnt 计数计到 11,计算结束 busy=0,最终结果 z 截取 res 的高 32 位,即 z=ffffffe2=-30。

用例 3: 0000\*ff21 (0\*(-223))

			515,000 ns	525, 000 ns		546.673 ns												
Name	Value		520 ns	530 ns	540 ns	550 ns	560 ns	570 ns	580 ns	590 ns	600 ns	610 ns	620 ns	630 ns	640 ns	650 ns	660 ns	670 =
₩ dk	1																	
₩ rst_n	1																	
> 🖬 x[15:0]	0000									0000								
> 👑 y[15:0]	0000	0000	ff21								0000							
start	0																	
> Mr z[31:0]	fffffe2									ffffffe2								
₩ busy	1																	
> 😻 X[15:0]	0000		fffa								0000							
> 🐶 Y[15:0]	ff21		0005								ff21							
> W res[32:0]	000007f90		1ffffff64	00001fe42	00000ff21	000007£90	000003fc8	000001fe4	000000ff2	0000007f9	0000003fo	0000001fe	0000000ff	00000007f	00000003f	00000001f	00000000f	000000007
> W minusX[15:0]	0000		0006								0000							
> W cnt[4:0]	02		11	00	01	02	03	04	05	06	07	08	09	( Oa	ОЪ	00	04	0.
¼ run	1																	
> <b>W</b> res1[32:0]	000007f90		1fff3ffo4	00001fe42	00000ff21	000007£90	000003fc8	000001fe4	000000ff2	0000007£9	0000003fo	0000001fe	0000000ff	00000007£	00000003f	00000001f	00000000£	000000007
> <b>W</b> res2[32:0]	000007f90		0000bffc4	00001fe42	00000ff21	000007£90	000003fc8	000001fe4	000000ff2	0000007£9	0000003fo	0000001fe	0000000ff	00000007£	00000003f	00000001f	00000000£	000000007
						676 000												
						676, 909 ns		695, 000 ns										1
Name	Value		650 ns	660 ns	670 ns	676. 909 ns  680 ns	690 ns		710 ns	720 ns	730 ns	740 ns	750 ns	760 ns	770 ns	780 ns	790 ns	800 :
Name	1		650 ns	660 ns	670 ns		690 ns		710 ns	720 ns	730 ns	740 ns	750 ns	760 ns	770 ns	780 ns	790 ns	800 :
	1		650 ns	860 ns	670 ns		690 ns		710 ns	720 ns	730 ns	740 ns	750 ns	760 ns	770 ns	780 ns	790 ns	800 ±
	1 1 0000		650 ns	660 ns	670 ns		690 ns		710 ns	720 ns	730 ns	740 ns	750 ns	760 ns	770 ns	780 ns	790 ns	800 ±
il dk il rst_n	1 1 0000 0000		650 ns	660 ns	870 ns				710 ns	720 ns	730 ns	740 ns		760 ns	770 ns		790 ns	800 =
	1 1 0000		650 ns	660 ns	670 ns		0000		710 ns	720 ns	730 ns	740 ns	1234	760 ns	770 ns	0000	790 ns	800 =
<pre># clk # rst_n &gt; % x(15:0) &gt; % y(15:0) # start &gt; % z(31:0)</pre>	1 1 0000 0000		650 ns	660 ns	670 ns		0000		710 ns	720 ns	730 ns	740 ns	1234	760 ns	770 ns	0000	790 ns	800 :
월 cik 월 rst_n > 월 x[15:0] > 월 y[15:0] 월 start	1 1 0000 0000 0 fffffe2 1		650 ns	660 ns			0000		710 ns	720 ns	1730 ns	740 ns	1234 1234	760 ns	770 ns	0000	1790 ns	
<pre># clk # rst_n &gt; ** x[15:0] &gt; ** y[15:0] # start &gt; ** z[31:0]</pre>	1 1 0000 0000 0 mmme2						0000		710 ns	720 ns	730 ns	740 ns	1234 1234	780 ns	770 ns	0000	790 ns	1800 =
ii cik ii rst_n  * ** * * * * * * * * * * * * * * * *	1 1 0000 0000 0 fffffe2 1		650 ns				0000	700 ns	710 ns	720 ns	730 ns	740 ns	1234 1234	780 ns	770 ns	0000	790 ns	800 :
# dk # rst_n  > ★ x[15:0]  > ★ y[15:0]  # start  > ★ z[31:0]  # busy  > ★ X[15:0]	1 1 0000 0000 0 1 11112 1 0000			680 ns			0000	7700 ns	710 ns	720 ns	730 ns	740 ns	1234 1234	780 ns	770 ns	0000	730 ns	800 : 
# cik # rst_n  > ** x[15:0]  > ** y[15:0] # start  > ** x[31:0] # busy  > ** x[15:0]  > ** res[32:0]  > ** minus x[15:0]	1 1 0000 0000 0 mme2 1 0000				£11111-2	680 ns	0000	7700 ns	710 ns		730 ns	740 ns	1234 1234			0000 0000 1234 1234		
ii cik ii rst_n  > 10 x[15:0]  > 11 x[15:0] ii start  > 10 x[31:0] ii busy  > 11 x[15:0]  > 12 x[15:0]  > 13 x[15:0]  > 14 x[15:0]  > 15 x[15:0]	1 1 0000 0000 0 1 1 1 0000 1721 000000003				£11111-2	680 ns	0000	7700 ns	710 ns		730 ns	740 ns	1234 1234			0000 0000 1234 1234 00000091a		
# cik # rst_n  > ** x[15:0]  > ** y[15:0] # start  > ** x[31:0] # busy  > ** x[15:0]  > ** res[32:0]  > ** minus x[15:0]	1 1 0000 0000 0 0 ffffe2 1 0000 ff21 000000003	000	0000001f	000000004	fffff*2	000000003	0000	7700 ns	710 ns		730 ns	740 ns	1234 1234	000002488	000001234	0000 0000 1234 1234 00000091a	ledor048d	0091a0246
cik    rst_n  >	1 1 0000 0000 0 0 mme2 1 1 0000 ff21 000000003 0000 of	000	0000001f	000000004	fffff*2	000000003	0000	7700 ns	710 as			740 ax	1234 1234	000002488	000001234	0000 0000 1234 1234 00000091a	ledor048d	0091a0246
li cik ii rst_n  > til (15:0)  > til (15:0)  ii start  > til (21:0)  iii busy  > til (15:0)  > til (16:0)    til (16:0)    til (16:0)	1 1 0000 0000 0 0 mme2 1 0000 m21 000000000000000000000000000	000 )	00000001f	00000000f	######################################	680 ns	0000	7700 ns	710 as	00000001	1730 ns	740 ax	1234 1234	000002488	000001234	0000 0000 1234 1234 0000091a edoc	1edcc048d	0091a0248

515ns; start=1; x 与 y 分别有信号, 为 0000 和 ff21

X[15:0]	0000
Y[15:0]	ff21
-X[15:0]	0000

cnt[4:0]	res[32:0]	res 备注
0	00001fe42	res 初始化,高 16 位为 0,低 17 位为尾部添 0 的被乘数
1	00000ff21	res 最低 2 位为 10, res 高 16 位加-x, 逻辑右移 1 位后的
		结果
2	000007f90	res 最低 2 位为 01, res 高 16 位加 x, 逻辑右移 1 位后的
		结果
3	000003fc8	res 最低 2 位为 00,直接逻辑右移 1 位后的结果
4	000001fe4	res 最低 2 位为 00,直接逻辑右移 1 位后的结果
5	000000ff2	res 最低 2 位为 00,直接逻辑右移 1 位后的结果
6	0000007f9	res 最低 2 位为 01, res 高 16 位加 x, 逻辑右移 1 位后的
		结果
7	0000003fc	res 最低 2 位为 01, res 高 16 位加 x, 逻辑右移 1 位后的
		结果
8	0000001fe	res 最低 2 位为 00,直接逻辑右移 1 位后的结果
9	0000000ff	res 最低 2 位为 10, res 高 16 位加-x, 逻辑右移 1 位后的
		结果
a	0000007f	res 最低 2 位为 11,直接逻辑右移 1 位后的结果
b	00000003f	res 最低 2 位为 11,直接逻辑右移 1 位后的结果
c	00000001f	res 最低 2 位为 11,直接逻辑右移 1 位后的结果
d	00000000f	res 最低 2 位为 11,直接逻辑右移 1 位后的结果
e	00000007	res 最低 2 位为 11,直接逻辑右移 1 位后的结果
f	000000003	res 最低 2 位为 11,直接逻辑右移 1 位后的结果
10	000000001	res 最低 2 位为 11,直接逻辑右移 1 位后的结果

6955ns: cnt 计数计到 11,计算结束 busy=0,最终结果 z 截取 res 的高 32 位,即 z=00000000=0。

附加题: 改进的 Booth 算法

booth2.v

内部信号解释:

信号	功能 (数据以补码表示)
X[16:0]	保存被乘数 x,两位符号位
Y[16:0]	保存乘数 y,两位符号位
minusX[16:0]	保存被乘数-x,两位符号位
doubleX[16:0]	保存被乘数 2x,两位符号位
minusDoubleX[16:0]	保存被乘数-2x,两位符号位
cnt[3:0]	计数器
run	同 busy
res[33:0]	其[32:1]位为最终结果。初始时高 17 位为 0, 低 17 位为尾部添 0 的乘数 y

用例 1: 0006\*0005 (6\*5)

Value

00000000

00005

1fffa

00003c000

0000fc000

3fff7c000

0001bc000

3ffebc000

₩ x[15:0] ₩ y[15:0] ₩ start ₩ z[31:0]

busy
 X[16:0]
 Y[16:0]
 Y[16:0]

₩ res[33:0]

**W** cnt[3:0]

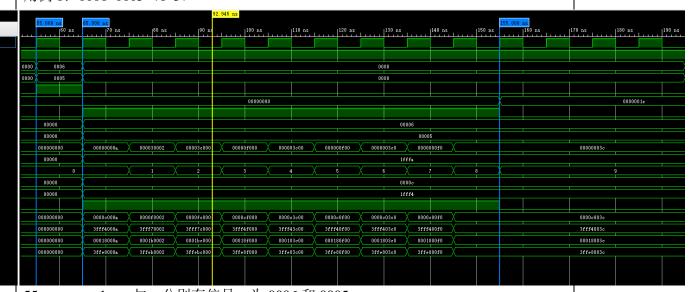
₩ res1[33:0]

₩ res2[33:0]

₩ res3[33:0]

₩ res4[33:0]

₩ minusX[16:0]



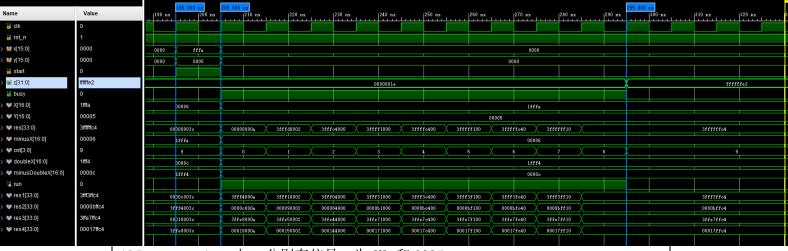
55ns; start=1; x 与 y 分别有信号, 为 0006 和 0005

X[16:0]	00006
Y[16:0]	00005
-X[16:0]	1fffa
2X[16:0]	0000c
-2X[16:0]	1fff4

cnt[4:0]	res[32:0]	res 备注
0	00000000a	res 初始化,高 17 位为 0,低 17 位为尾部添 0 的被乘数
1	000030002	res 最低 3 位为 101, res 高 17 位加-x, 逻辑右移 2 位后
		的结果
2	00003c000	res 最低 3 位为 001, res 高 17 位加 x, 逻辑右移 2 位后
		的结果
3	00000f000	res 最低 3 位为 000, 直接逻辑右移 2 位后的结果
4	000003c00	res 最低 3 位为 000, 直接逻辑右移 2 位后的结果
5	000000f00	res 最低 3 位为 000, 直接逻辑右移 2 位后的结果
6	0000003c0	res 最低 3 位为 000, 直接逻辑右移 2 位后的结果
7	000000f0	res 最低 3 位为 000, 直接逻辑右移 2 位后的结果
8	00000003c	res 最低 3 位为 000, 直接逻辑右移 2 位后的结果

155ns: cnt 计数计到 8,计算结束 busy=0,最终结果 z 截取 res 的[32:1]位,即 z=0000001e=30

用例 2: fffa\*0005 (-6\*5)



195ns; start=1; x 与 y 分别有信号, 为 fffa 和 0005

X[16:0]	1fffa
Y[16:0]	00005
-X[16:0]	00006
2X[16:0]	1fff4
-2X[16:0]	0000c

cnt[4:0]	res[32:0]	res 备注
0	00000000a	res 初始化,高 17 位为 0,低 17 位为尾部添 0 的被乘数
1	3fffd0002	res 最低 3 位为 101, res 高 17 位加-x, 逻辑右移 2 位后
		的结果
2	3fffc4000	res 最低 3 位为 001, res 高 17 位加 x, 逻辑右移 2 位后
		的结果
3	3ffff1000	res 最低 3 位为 000, 直接逻辑右移 2 位后的结果
4	3ffffc400	res 最低 3 位为 000, 直接逻辑右移 2 位后的结果
5	3fffff100	res 最低 3 位为 000, 直接逻辑右移 2 位后的结果
6	3fffffc40	res 最低 3 位为 000, 直接逻辑右移 2 位后的结果
7	3ffffff10	res 最低 3 位为 000, 直接逻辑右移 2 位后的结果
8	3ffffffc4	res 最低 3 位为 000, 直接逻辑右移 2 位后的结果

295ns: cnt 计数计到 8, 计算结束 busy=0, 最终结果 z 截取 res 的[32:1]位, 即 z=ffffffe2=-30

| Sample | S

用例 3: 0000\*ff21(0000\*(-223))

335ns; start=1; x 与 y 分别有信号, 为 0000 和 ff21

X[16:0]	00000
Y[16:0]	1ff21
-X[16:0]	00000
2X[16:0]	00000
-2X[16:0]	00000

cnt[4:0]	res[32:0]	res 备注
0	00001fe42	res 初始化,高 17 位为 0, 低 17 位为尾部添 0 的被乘数
1	000007f90	res 最低 3 位为 010, res 高 17 位加 x, 逻辑右移 2 位后
		的结果
2	000001fe4	res 最低 3 位为 000,直接逻辑右移 2 位后的结果
3	0000007f9	res 最低 3 位为 100, res 高 17 位加-2x, 逻辑右移 2 位后
		的结果
4	0000001fe	res 最低 3 位为 001, res 高 17 位加 x, 逻辑右移 2 位后
		的结果
5	00000007f	res 最低 3 位为 110, res 高 17 位加-x, 逻辑右移 2 位后
		的结果
6	00000001f	res 最低 3 位为 111,直接逻辑右移 2 位后的结果
7	000000007	res 最低 3 位为 111,直接逻辑右移 2 位后的结果
8	000000001	res 最低 3 位为 111,直接逻辑右移 2 位后的结果

435ns: cnt 计数计到 8,计算结束 busy=0,最终结果 z 截取 res 的[32:1]位,即 z=00000000=0