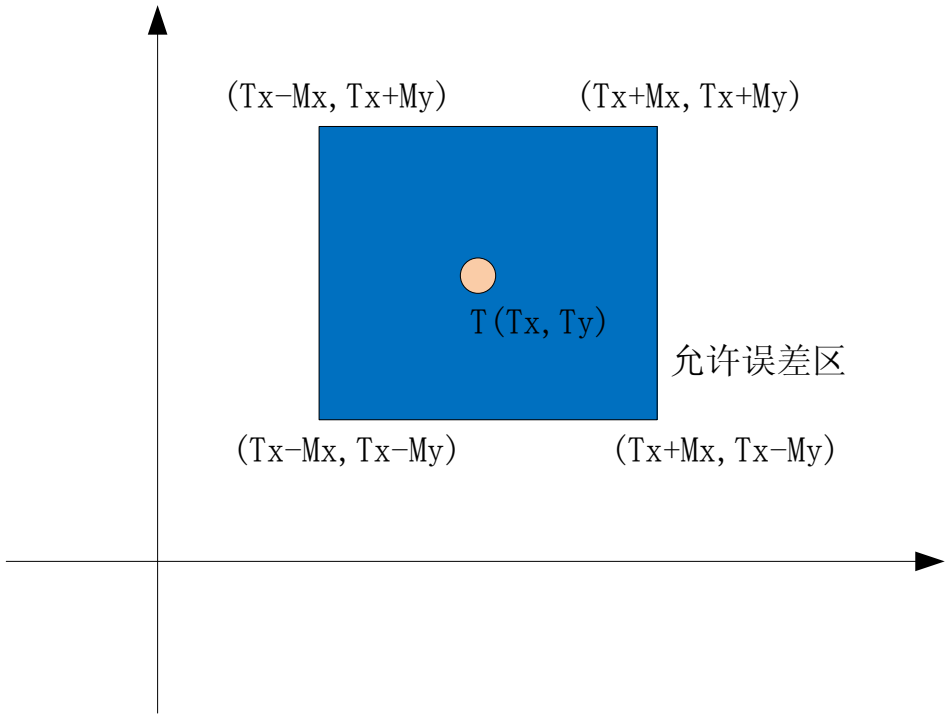
二维位置比较输出设计

版本	日期	修订
1.0	2014-8-6	黄廉真

目录

第一部分：二维比较输出原理

设 A（Xa，Ya）为目标点，OFFSET(Fx,Fy)为偏移量,T（Tx,Ty)为校准后的目标点。其中T=A+OFFSET=（Xa+Fx，Ya+Fy）。MAXERR(Mx，My）为最大允许位置误差（误差区）。如下图所示，T 为校准后的目标点，当系统运行至蓝色区域，则认为已经进入目标区域，T 点已经进入。



当轨迹进入 T 点后，系统会进一步等待并寻找最近 T 点，再输出 IO 控制。

1.1 最优点寻找算法

设系统的实际位置 R(Rx,Fy)，误差评估函数为

E(R)=Abs(Rx-Tx)+Abs(Ry-Ty) .....(1)

如下图所示，随着实际位置的运动，实际位置会越来越 T，进而 E 趋向于零或者等于 0，然后变大。

情况 1：系统能够达到 T，则最小 E 为 0，此时立刻输出 IO 控制。此时误差为 0。

情况 2：系统去不到 T，则 E 达到最小值 E0，然后再会变大。此时认为最优点找到，然后在 E1 出输出 IO，误差为 E1。

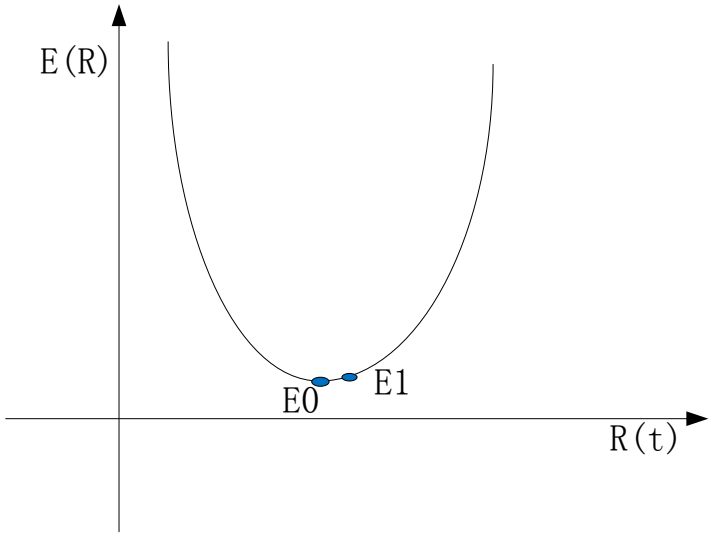
为了防止需找的 E1 为非最小值。采用以下方式，设当前误差为 E，最小 E 为 E0(E0 初始化为(Mx+My)，然后

if E<E0 then E0 = E;

else if E>E0+TH then IO out;

else E0 = E0;

其中 TH 为抖动阈值。例如 T=（0,0），R 序列为
(-3,0), (-3,0), (-2,0), (-2,1), (-2,2), (-1,1), (-1,0), (-1,0),(0,0),(1,0)……
当 TH=0，则在以上的(-2,1),机会触发 IO 输出，而当 TH=2 时，则以上的序列就不会再触发 IO。



为了简化设计，让 Mx=My。因此比较算法的参数为 TH 和 Mxy，OFFSET。

第二部分：二维比较输出应用接口

2.1 寄存器定义

2.1.1 reg_ctrl（0x0）只写

bits	变量	读写性	复位值	备注
15	on	只写	0	写 1 使能比较，在使能比较的时候，系统将会以启动时刻的位置为参考原点，然后进行后续的比较。如果需要重新设置参考点，则必须关闭模块，再打开。（必须注意这一点）。 当使能该功能后，比较输出 IO 口的控制权将自动交付给二维比较输出。一维比较输出将不能控制比较输出 IO。否则控制权在一维比较输出处。
14	off	只写	0	写 1 关闭比较。
13	mode_pulse			写 1 脉冲模式（默认）
12	mode_level			写 1 电平模式，
11~9	axi_x_sel		0	X 轴选择（从 8 个通道中选择）
8~6	axi_y_sel		0	Y 轴选择（从 8 个通道中选择）
5	axi_src_valid			0：屏蔽参数 x_src 和 y_src (不修改) 1：修改参数 x_src 和 y_src
4	cmp_src_sel_enc			使用编码器作为比较源
3	cmp_ src_sel_pfr			使用规划位置作为比较源
2	out_ reverse_on			使能输出取反
1	out_ reverse_off			输出取反
0	flag_rst	只写		写 1 复位状态标志位和清空 fifo。

2.1.2 pulse_width（0x1）只写

bits	变量	读写性	复位值	备注
15~0	pulse_width		16	用于脉冲模式下（us）

2.1.3 offset_x_low（0x2）只写

bits	变量	读写性	复位值	备注
15~0	offset_x[15:0]		0	X 轴偏移量低 16bits

2.1.4 offset_x_high（0x3）只写

bits	变量	读写性	复位值	备注
15~14	io_set[1:0]			
13	io_set_valid			
12~8	保留			
7~0	offset_x[23:16]		0	X 轴偏移量高 8bis

2.1.5 offset_y_low（0x4）只写

bits	变量	读写性	复位值	备注
15~0	offset_y[15:0]		0	Y 轴偏移量低 16bits

2.1.6 offset_y_high（0x5）只写

bits	变量	读写性	复位值	备注
15~8	保留		0	
7~0	offset_y[23:16]		0	Y 轴偏移量高 8bis

2.1.7 err（0x6）只写

bits	变量	读写性	复位值	备注
15	threshold_valid		0	0：屏蔽参数 threshold (不修改) 1：修改参数 threshold
14:10	threshold[4:0]		4	用于最优点计算。范围 0~31
9	max_err_valid		0	0：屏蔽参数 max_err (不修改) 1：修改参数 max_err
8:0	max_err[8:0]		256	最大允许位置误差（误差区），范围：0~511。如果设备实际轨迹与规划轨迹误差比较大，（主要由于设备同步性不好，或者各个滤波的滤波导致轨迹变化两个原因导致），那么该值需要舍得比较大。 建议 max_err 同时小于相邻两个点的“X 轴增量的 1/3”和“Y 轴增量的 1/3”。

2.1.8 point_x_low (0x8) 只写

bits	变量	读写性	复位值	备注
15~0	point_x [15:0]		0	目标点 X 轴 16bits

2.1.9 point_x_high（0x9）只写

bits	变量	读写性	复位值	备注
15~8	保留		0	
7~0	point_x [23:16]		0	目标点 X 轴高 8bis

2.1.10 point_y_low (0xA) 只写

bits	变量	读写性	复位值	备注
15~0	point_y [15:0]		0	目标点 Y 轴 16bits
			0	

2.1.11 point_y_high（0xB）只写

bits	变量	读写性	复位值	备注
15~14	io_ctrl[1:0]		0	
13	y_mask		0	1：屏蔽 Y 轴的比较，认为 Y 不需要对比 0：使能 Y 轴比较
12	x_mask		0	1：屏蔽 X 轴的比较，认为 X 不需要对比 0：使能 X 轴比较
11~8	保留			
7~0	point_y [23:16]		0	目标点 Y 轴高 8bits

2.1.12 reg_status(0x0)只读

bits	变量	读写性	复位值	备注
15	cmp_on	只读	0	0：处于关闭状态 1：处于打开状态
14	cmp_status	只读	0	0：cmp_on 关闭，或者正在正常工作 1：停止工作，fifo 空，指令执行完成了。
13	cmp_model	只读	0	0：脉冲模式（默认） 1：电平模式
12	cmp_src	只读	0	0：编码器（默认） 1：规划位置
11	out_reverse			0:输出不取反 1:输出取反
10	fifo_overflow	只读	0	fifo 溢出，需要 flag_rst 清除。
9~0	cmp_fifo_num	只读	0	当前 fifo 区中有的指令个数，0~512。512 表示已经满了

2.1.13 cmp_cnt(0x1)只读

bits	变量	读写性	复位值	备注
15~0	cnt	只读	0	模块在从关闭状态或者 flag_rst 时清除。 每正常触发一次或者强制触发一次（通过寄存器 offset_x_high）,该值+1。如果强制触发和正常触发同时发生，则只会+1。

写入注意：

offset_x, offset_y 的写入

offset_x, offset_y 这两个数都是 24bits，一次写完成不了，需要两次写操作。为了保证数据的一致性。只有 offset_x_low 写入，整个 offset_x 才有效。

两个写入可以插入其他操作。

简而言之，数据写 offset_x_high 只是暂存，不会更新。写入 offset_x_high 才会更行整个 offset_x。**先写高位，再写低位.**

比如需要往 offset_x 写 0x123456;

offset_x_high = 0x12;

offset_x_low = 0x3456;

至此,整个写入完成。

如果需要修改参数，且高位至一样，则可以不需要写高位，只需要写低位数据即可。

如当前 offset_x=0x123456，需要修改 0x128765，那么可以简化操作为

//offset_x_high = 0x12;

offset_x_low = 0x8765;

offset_y 的操作也是如此。

比较点的写入

一个比较点由 4 个 16bits 组成。往 point_x_low，point_x_high，point_y_low 写入数据时，这些数据都只会存在缓存中，只有 point_y_high 写入数据，整个 4 个 16bits 的数据才会同时压入 fifo 中。

因此，当需要写入一个比较点时，必须先保证 point_x_low，point_x_high，point_y_low 的写入完成，然后在写入 point_y_high。

4 个写入可以插入其他操作。

如果 point_x_low，point_x_high，point_y_low 的数据和旧的数据相同，可以不修改。但是 point_y_high 必须写入。

先写低位，再写高位

如需要压入以下点：

1、在（0x123456,0x202233，触发 IO 状态为 0x3）

2、在（0x123456,0x212233，触发 IO 状态为 0x1）

可以如下

point_x_low = 0x3456;

point_x_high = 0x0012;

point_y_low = 0x2233;

point_y_high= 0x3020;

//////////////////////////

//由于 point_x 不变，且 point_y_low 也不变，只需要写入 point_y_high 就可。
//point_x_low = 0x3456;
//point_x_high = 0x0012;
//point_y_low = 0x2233;
point_y_high= 01021;

正常触发和强制触发

在模块时能后，有两种触发：一种是运行至指定位置触发。另一是通过寄存器操作地址“0x3”的 io_set，进行强制触发。

强制触发的优先级最高。如果两种触发同时发生，则执行强制触发的行为。

强制触发和正常触发都是引发触发次数+1。但是如果同时发生，则只会+1。

正常触发和强制触发都会对外引发一个触发信号，外部可以用该信号作为捕获源。

位置比较需要用的变量如下表 2。图 5 是位置比较输出的实现框图。

表 2 比较输出变量表

变量	取值	备注
cmp_state	0：正常 1：错误，fifo 没有为空，读到不到数据 该错误在关闭 cmp_on 后自动清除。	状态
cmp_on	0：关闭 cmp 1：打开 cmp 当模块进入错误的时候，需要关闭 cmp_on，然后重新打开才能重新工作	控制量
axi_x_sel	x 轴编码器选择	
axi_y_sel	y 轴编码器选择	
cmp_io	专用输出管脚	状态
reverse	0：正常输出 1：输出取反	控制量
cmp_mode	0：脉冲模式。需要输出有效的时候，专用输出管脚输出一个脉冲。脉冲宽度由 pulse_width 控制 1：电平模式。需要输出有效的时候，专用输出管脚输出高电平；当需要输出无效的时候，专用输出管脚输出低电平	控制量
pulse_width[15:0]	x(us)	控制量
cmp_valid	每比较完成一次发生一个脉冲	状态位
cmp_sel (0)	比较的输入选择 0：编码器作为输入 1：规划脉冲输出作为输入	控制量
cmp_offset	在启动 cmp_on 时，将该值装载在计数模块中，作为初值。	控制量
fif_din	上层输入的比较数据格式如下： {is_cmp, axi_sel, io_ctrl[1:0], delta_data[11:0]}	
fifo_num	当前 fifo 里存有的比较数据个数。上位机保证写不溢出。	控制量

如图 5 所示，位置比较输出模块有一个 fifo，用于存储一系列需要比较的数据（fifo_din）。整个处理流程如图 6 所示，而状态机如图 7 所示。

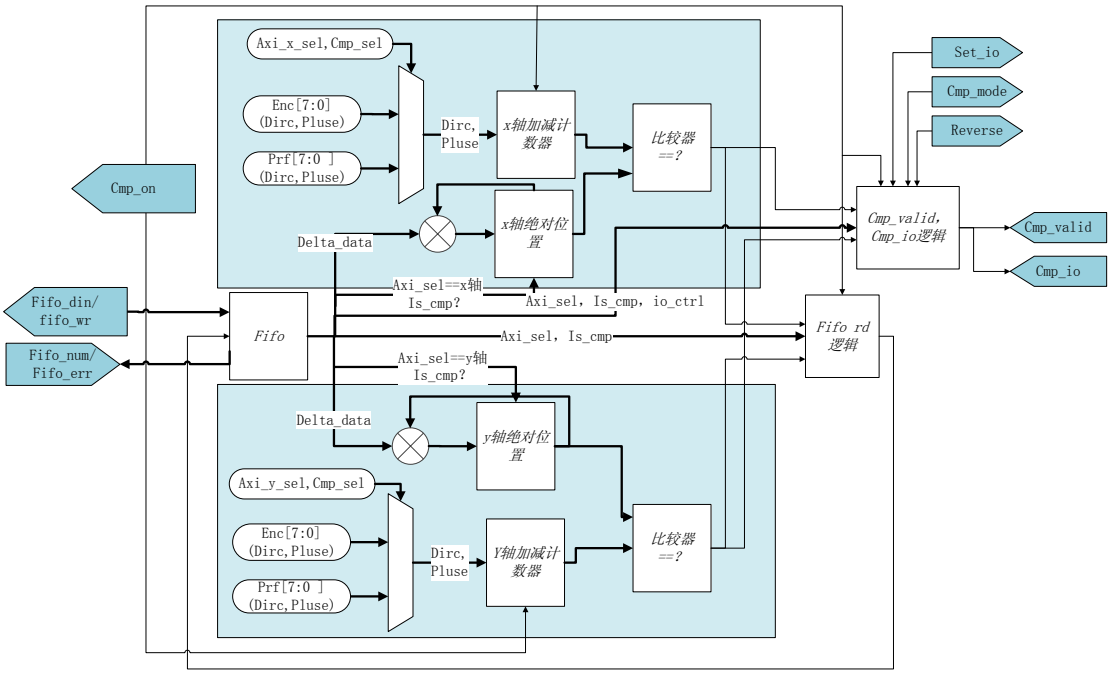

Axi_x_sel,Cmp_sel
Enc[7:0]
(Dirc,Pluse)
Prf[7:0]
(Dirc,Pluse)
Dirc,
Pluse
x轴加减计数器
比较器
==?
Set_io
Cmp_mode
Reverse
Cmp_on
x轴绝对位置
Delta_data
Cmp_valid,
Cmp_io逻辑
Cmp_valid
Cmp_io
Fifo_din/
fifo_wr
Axi_sel==x轴
Is_cmp?
Axi_sel, Is_cmp, io_ctrl
Fifo
Fifo_num/
Fifo_err
Axi_sel, Is_cmp
Fifo rd
逻辑
Axi_sel==y轴
Is_cmp?
Delta_data
y轴绝对位置
Axi_y_sel,Cmp_sel
Enc[7:0]
(Dirc,Pluse)
Prf[7:0]
(Dirc,Pluse)
Dirc,
Pluse
Y轴加减计数器
比较器
==?

图 5 位置比较结构框图

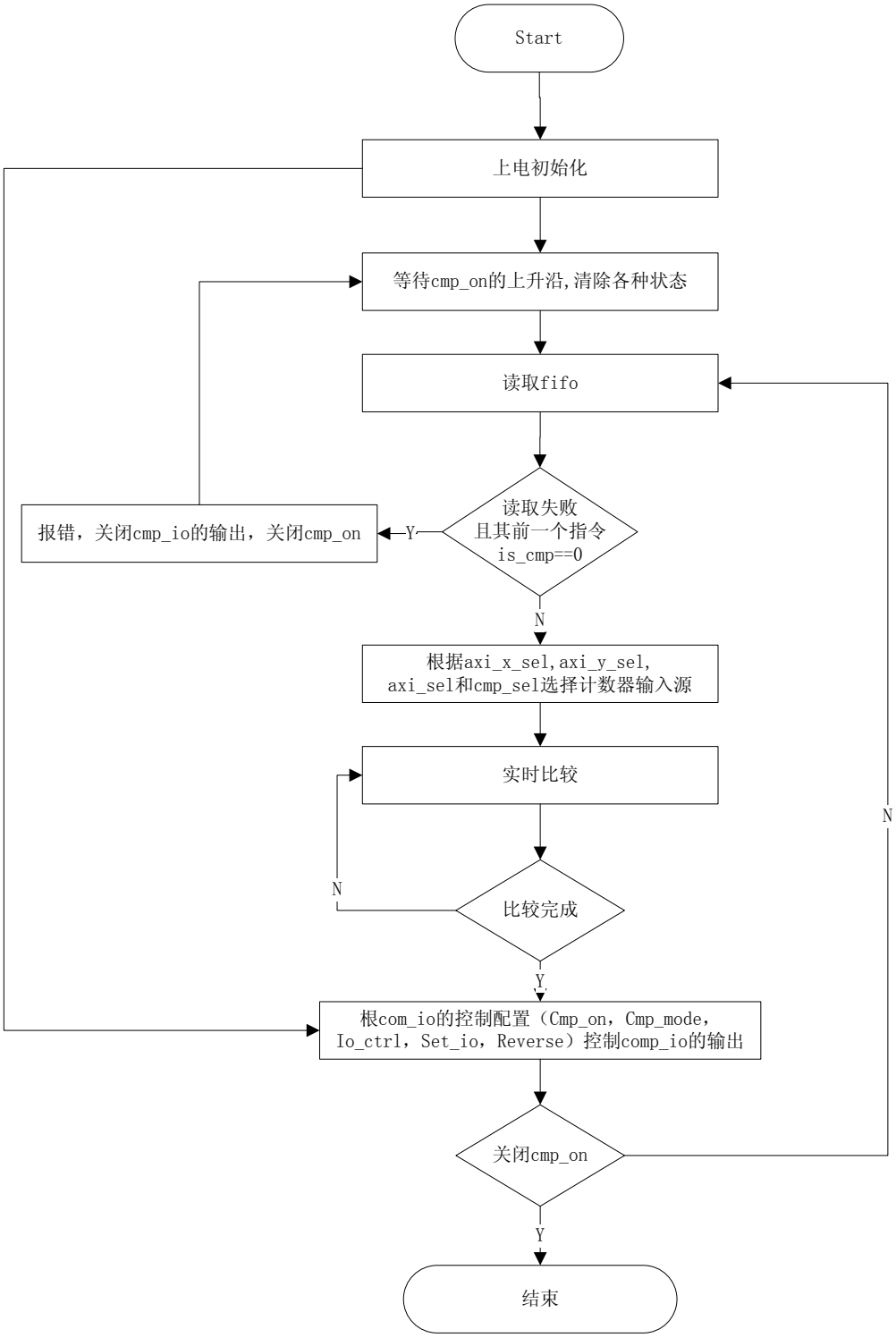

Start
上电初始化
等待cmp_on的上升沿,清除各种状态
读取fifo
读取失败
且其前一个指令
is_cmp==0
Y
报错，关闭cmp_io的输出，关闭cmp_on
N
根据axi_x_sel,axi_y_sel,
axi_sel和cmp_sel选择计数器输入源
实时比较
N
比较完成
Y
根com_io的控制配置（Cmp_on，Cmp_mode，
Io_ctrl，Set_io，Reverse）控制comp_io的输出
关闭cmp_on
N
Y
结束

图 6 比较输出处理流程

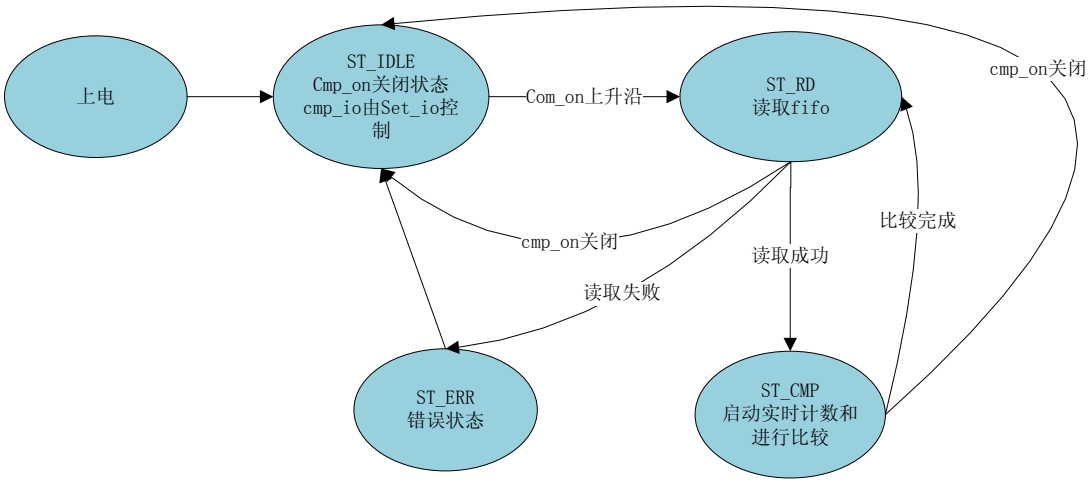


图 7 比较输出状态机

指令实例

如图 8 所示，程序期望在指定输出。压至缓冲区的指令格式为：

{is_cmp, axi_sel, io_ctrl[1:0], delta_data[11:0]}，具体意义见表 3

表 2 比较输出指令解析表

变量	取值	备注
is_cmp	0：不需要进行等待一次比较完成触发，直接读取下一条指令 1：需要完成一次比较完成触发，才能读取下一个指令	状态
axi_sel	0：选择 x 轴，指令的 delta_data 属于 x 轴，与 x 轴比较 1：选择 y 轴，指令的 delta_data 属于 y 轴，与 y 轴比较	控制量
io_ctrl	xx: 在比较完成时，选定 IO 的输出	
delta_data	位置的增量值	

对于样例 1，在启动 cmp_on 时，内部 32bit 的绝对位置 abs_pos_x,abs_pos_y 清零，同时 32 编码器位置 enc_pos_x，enc_pos_y 清零。

而上位机压下的指令为：

{1’b1, 1’b0, 2’bxx, 12’d200};->abs_pos_x+=200 = 200，需要等待比较

{1’b1, 1’b0, 2’bxx, 12’d200};->abs_pos_x+=200 = 400，需要等待比较

{1’b1, 1’b1, 2’bxx, 12’d100};->abs_pos_y+=100 = 100，需要等待比较

{1’b1, 1’b0, 2’bxx, 12’d-200};->abs_pos_x+=-200 = 200，需要等待比较

{1’b1, 1’b0, 2’bxx, 12’d-200};->abs_pos_x+=-200 = 000，需要等待比较

{1’b1, 1’b1, 2’bxx, 12’d100};->abs_pos_y+=100 = 200，需要等待比较

{1’b1, 1’b0, 2’bxx, 12’d200};->abs_pos_x+=200 = 200，需要等待比较

{1’b1, 1’b0, 2’bxx, 12’d200};->abs_pos_x+=200 = 400，需要等待比较

{1’b1, 1’b1, 2’bxx, 12’d100};->abs_pos_y+=100 = 300，需要等待比较

{1’b1, 1’b0, 2’bxx, 12’d-200};->abs_pos_x+=-200 = 200，需要等待比较

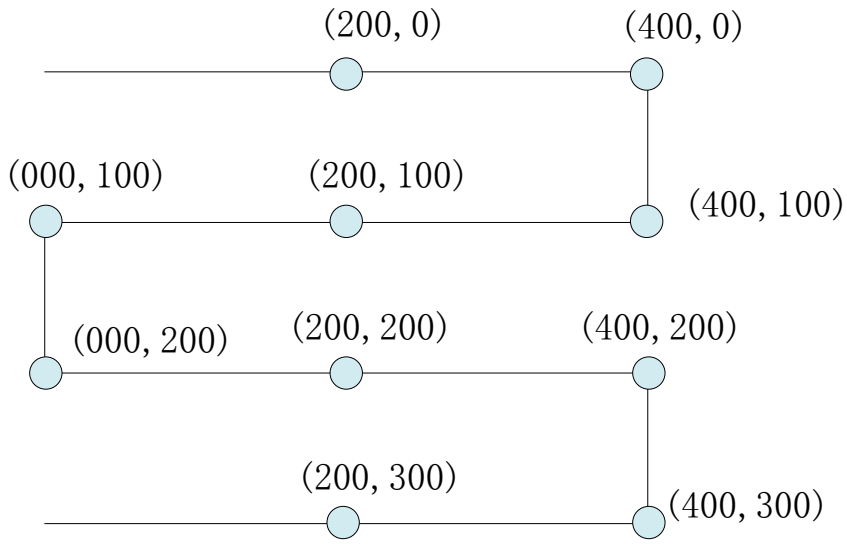


图 8 比较输出样例 1

对于样例 2，在启动 cmp_on 时，内部 32bit 的绝对位置 abs_pos_x,abs_pos_y 清零，同时 32 编码器位置 enc_pos_x，enc_pos_y 清零。

而上位机压下的指令为：

{1'b0, 1'b0, 2'bxx, 12'd2047};->abs_pos_x+=2047 = 2047，不需要等待比较
{1'b0, 1'b0, 2'bxx, 12'd2047};->abs_pos_x+=2047 = 4094，不需要等待比较
{1'b1, 1'b0, 2'bxx, 12'd1906};->abs_pos_x+=1906 = 6000，需要等待比较

{1'b0, 1'b0, 2'bxx, 12'd2047};->abs_pos_x+=2047 = 8047，不需要等待比较
{1'b0, 1'b0, 2'bxx, 12'd2047};->abs_pos_x+=2047 = 1094，不需要等待比较
{1'b1, 1'b0, 2'bxx, 12'd1906};->abs_pos_x+=1906 = 12000，需要等待比较

{1'b0, 1'b1, 2'bxx, 12'd2047};->abs_pos_y+=2047 = 2047，不需要等待比较
{1'b0, 1'b1, 2'bxx, 12'd2047};->abs_pos_y+=2047 = 4094，不需要等待比较
{1'b0, 1'b1, 2'bxx, 12'd2047};->abs_pos_y+=2047 = 6141，不需要等待比较
{1'b0, 1'b1, 2'bxx, 12'd2047};->abs_pos_y+=-2047 = 8188，不需要等待比较
{1'b1, 1'b0, 2'bxx, 12'd2047};->abs_pos_y+=-1812 = 10000，需要等待比较

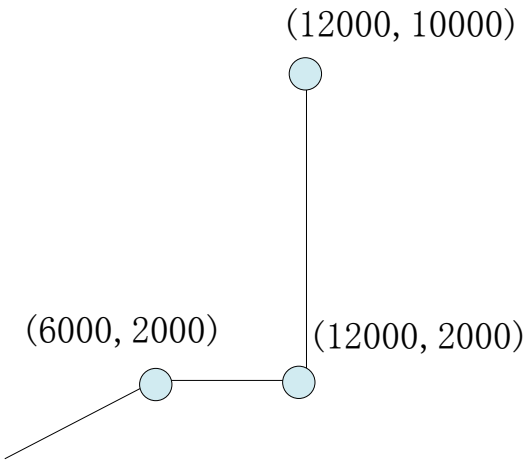


图 9 比较输出样例 2