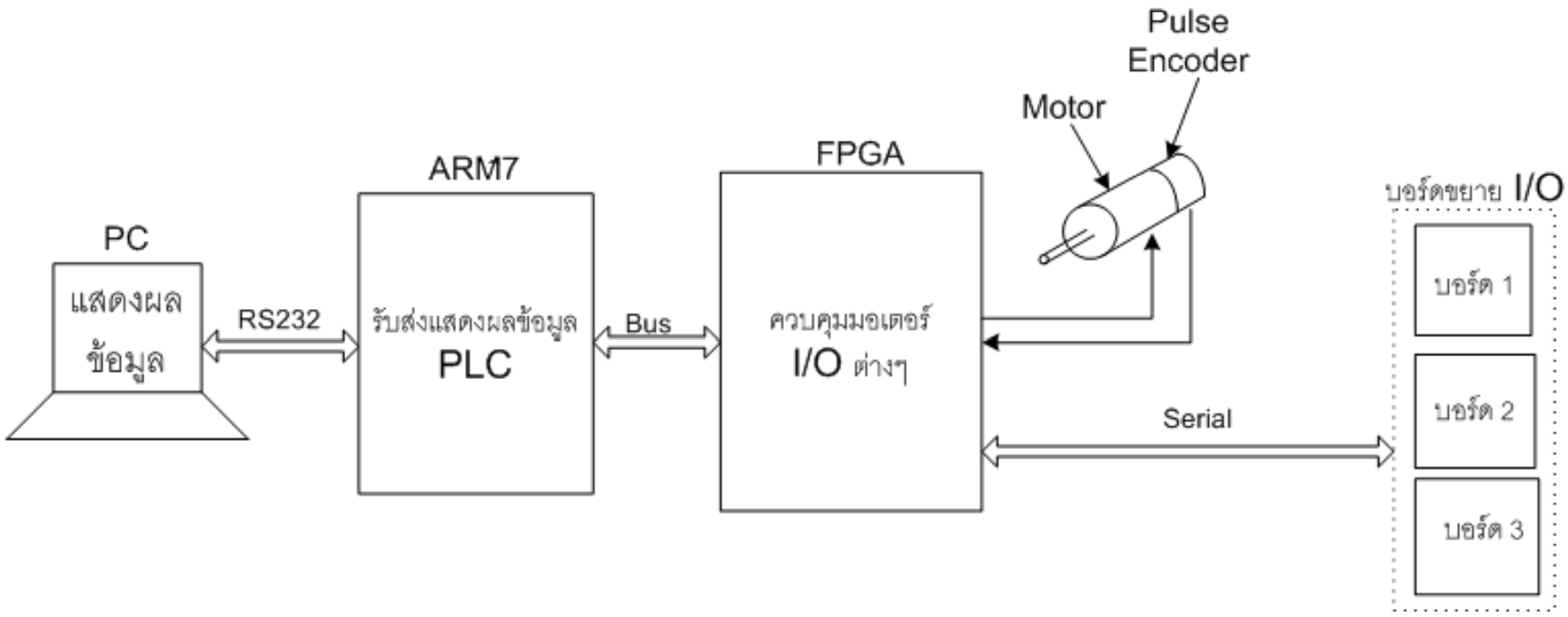
การเพิ่มประสิทธิภาพการทำงาน บนระบบควบคุมมอเตอร์หลายแกนด้วย FPGA

ธีรพงศ์ ฟองจันทร์, วุฒิภัทร คอวนิช, กิตติพงศ์ เอกไชย, อภิสิทธิ์ ตันตระวรศิลป์, อุดม โกมินทร์, ภานุพันธ์ ขวัญสุด, วุฒิกร เชาว์ประมวลกุล, ธีศิษฏ์ ลีลาสวัสดิ์สุข, ชลลดา ธีระวร และ กมลวรรณ ตันไถง

NECTEC (ICA)
Email : theerapong.fongjun@nectec.or.th

NECTEC-ACE 2010
Aeptember 23, 2010

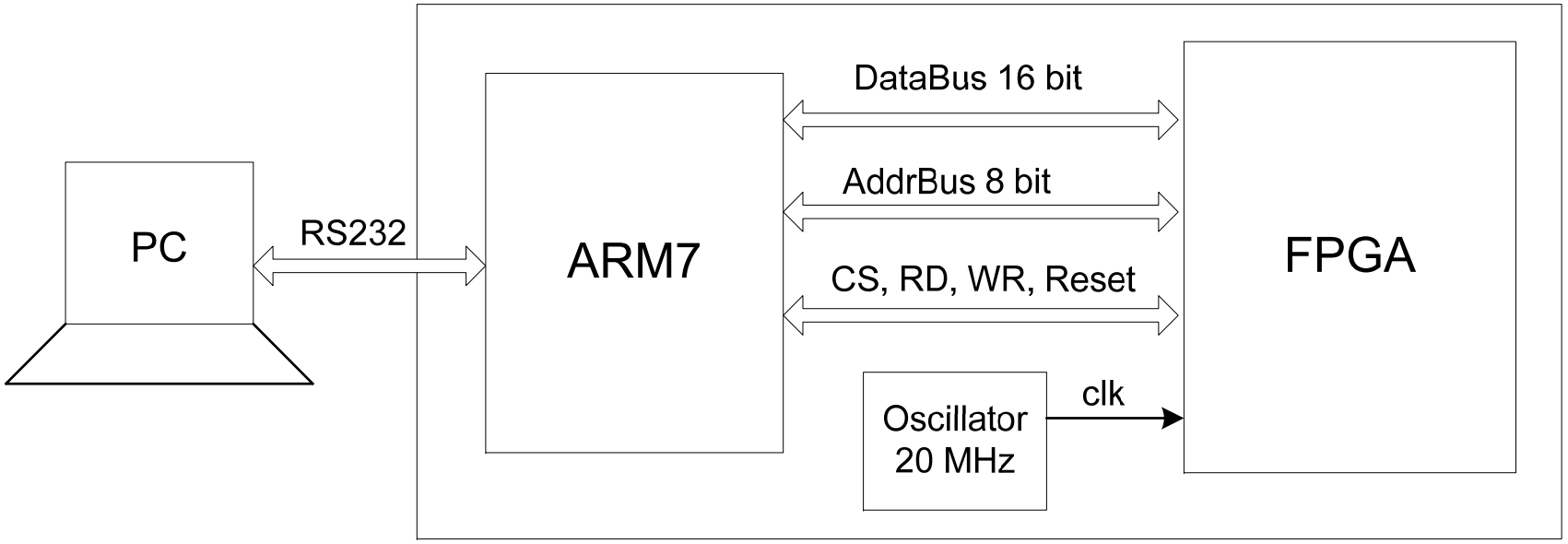
Outline


Pulse
Encoder
Motor
FPGA
ARM7
บอร์ดขยาย I/O
PC
แสดงผล
ข้อมูล
RS232
รับส่งแสดงผลข้อมูล
PLC
Bus
ควบคุมมอเตอร์
I/O ต่างๆ
Serial
บอร์ด 1
บอร์ด 2
บอร์ด 3

Outline

- โครงสร้างของระบบ
- การกรองสัญญาณรบกวน
- การอ่านตำแหน่งของมอเตอร์จากสัญญาณพัลส์เอ็นโคดเดอร์
- ตัวควบคุมพีไอดี
- การเพิ่มจำนวนช่องสัญญาณ I/O ด้วยการเชื่อมต่อแบบอนุกรม
- หน้าที่การทำงานอื่น ๆสำหรับควบคุมเครื่องจักร CNC
- ผลการทดลอง
- สรุป

โครงสร้างของระบบ



- FPGA (XC3S400-4tq144)
- ARM7 (LPC2148)
- CS, RD, WR active Low

โครงสร้างของระบบ

```
 1  RDcs <= RD or CS;
 2  WRcs <= WR or CS;
 3  DataBusWR <= DataBus;
 4
 5  DataBus2Direct:process(Reset,RDcs,DataBusRD)
 6  begin
 7   if(Reset = '1')then
 8    DataBus  <= (others=>'Z');
 9   elsif(RDcs = '0')then
10    DataBus  <= DataBusRD;
11   else
12    DataBus  <= (others=>'Z');
13   end if;
14  end process DataBus2Direct;
```

- Data Bus ใช้งานร่วมกับ module อื่นๆ
- High Impedance Data Bus เพื่อป้องกันการชนของข้อมูล

โครงสร้างของระบบ

```
 1  signal tempData16bit : std_logic_vector(15 downto 0);
 2
 3  WRtoFPGA:process(Reset,WRcs,AddrBus,DataBusWR)
 4  begin
 5   if(Reset='1')then
 6    tempData16bit <= (others=>'0');
 7    Ref           <= (others=>'0');
 8   elsif(WRcs'event and WRcs='1')then
 9    if(AddrBus = AddrRefLo)then
10     tempData16bit(15 downto 0) <= DataBusWR(15 downto 0);
11    elsif(AddrBus = AddrRefHi)then
12     Ref(31 downto 0)    <= DataBusWR(15 downto 0)
13                         & tempData16bit(15 downto 0);
14    end if;
15   end if;
16  end process WRtoFPGA;
```

- Data Register 32 bit
- 2 times Data Writing via Data bus 16 bit
- ป้องกันการทำงานที่ผิดพลาดอันเกิดจากการแยกกันของข้อมูล
- Sending 32 bit data TO 32 bit register : Low Data Word to Data Bus, THEN High Data Word to Data Bus

โครงสร้างของระบบ

```
 1   MultiplexRDfromFPGA:process(Reset,RDcs,AddrBus)
 2   begin
 3    if(Reset = '1')then
 4          DataBusRD <= x"0000";
 5    elsif(RDcs'event and RDcs='0')then
 6     if(AddrBus    = AddrEncLo)then
 7      DataBusRD(15 downto 0)     <= ENC(15 downto 0);
 8          tempData16bit(15 downto 0) <= ENC(31 downto 16);
 9     elsif(AddrBus = AddrEncHi)then
10      DataBusRD(15 downto 0) <= tempData16bit(15 downto 0);
11     end if;
12    end if;
13   end process MultiplexRDfromFPGA;
```

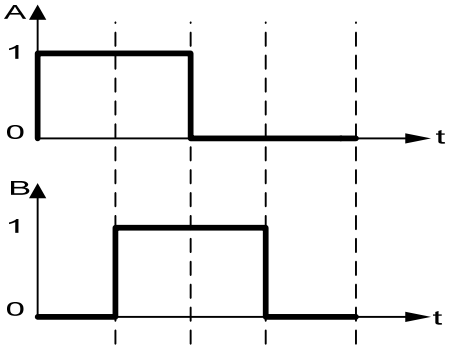
- Data Register 32 bit
- 2 Times Data Reading via Data Bus 16 bit
- ป้องกันการอ่านข้อมูลผิดพลาด อันเกิดจากการอ่านข้อมูลแยกกัน ในขณะที่ข้อมูลมีการเคลื่อนไหวตลอดเวลา
- Sending 32 bit data FROM 32 bit register : Low Data Word to Data Bus, THEN High Data Word to Data Bus

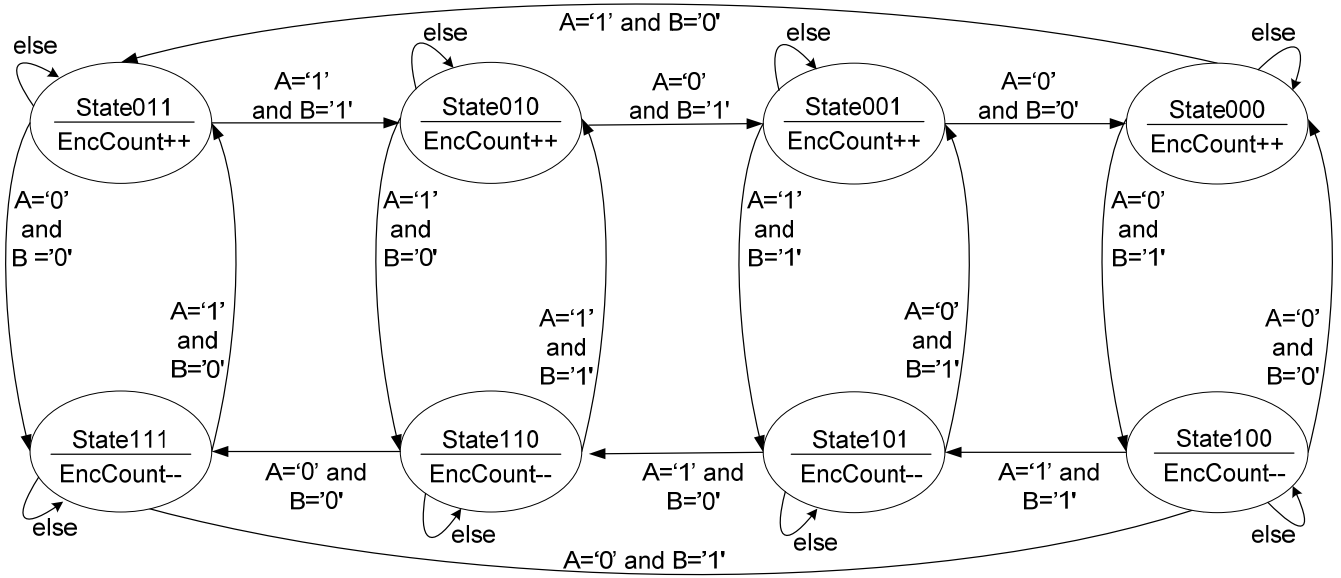
การกรองสัญญาณรบกวน

```
 1  FilterENC:process(clk)
 2  variable temp: std_logic_vector(0 to 8);
 3  begin
 4   if (clk'event and clk = '1') then
 5    temp(1 to 8) := temp(0 to 7);
 6    temp(0) := CHin;
 7    if temp(1 to 8) = "00000000" then
 8     filt <= '0';
 9    end if;
10    if temp(1 to 8) = "11111111" then
11     filt <= '1';
12    end if;
13   end if;
14  end process FilterENC;
```

- Low Pass Filter
- Shift Register
- Comparing data with Filter noise

การอ่านตำแหน่งของมอเตอร์จากสัญญาณพัลส์เอ็นโคดเดอร์





- Initial State from Input Data
- Moore Finite-State Machine
  - ตัดรูปแบบของสัญญาณ input อื่นๆที่ไม่ถูกต้องออกไป ไม่ให้มีผลต่อการนับค่าพัลส์เอ็นโคดเดอร์
  - ลดความผิดพลาดในการนับค่าพัลส์เอ็นโคดเดอร์ อันเนื่องมาจากสัญญาณรบกวน

ตัวควบคุมพีไอดี

$$u(n) = k_p e(n) + k_i \sum_{j=0}^{n} e(j) + k_d [e(n) - e(n-1)]$$

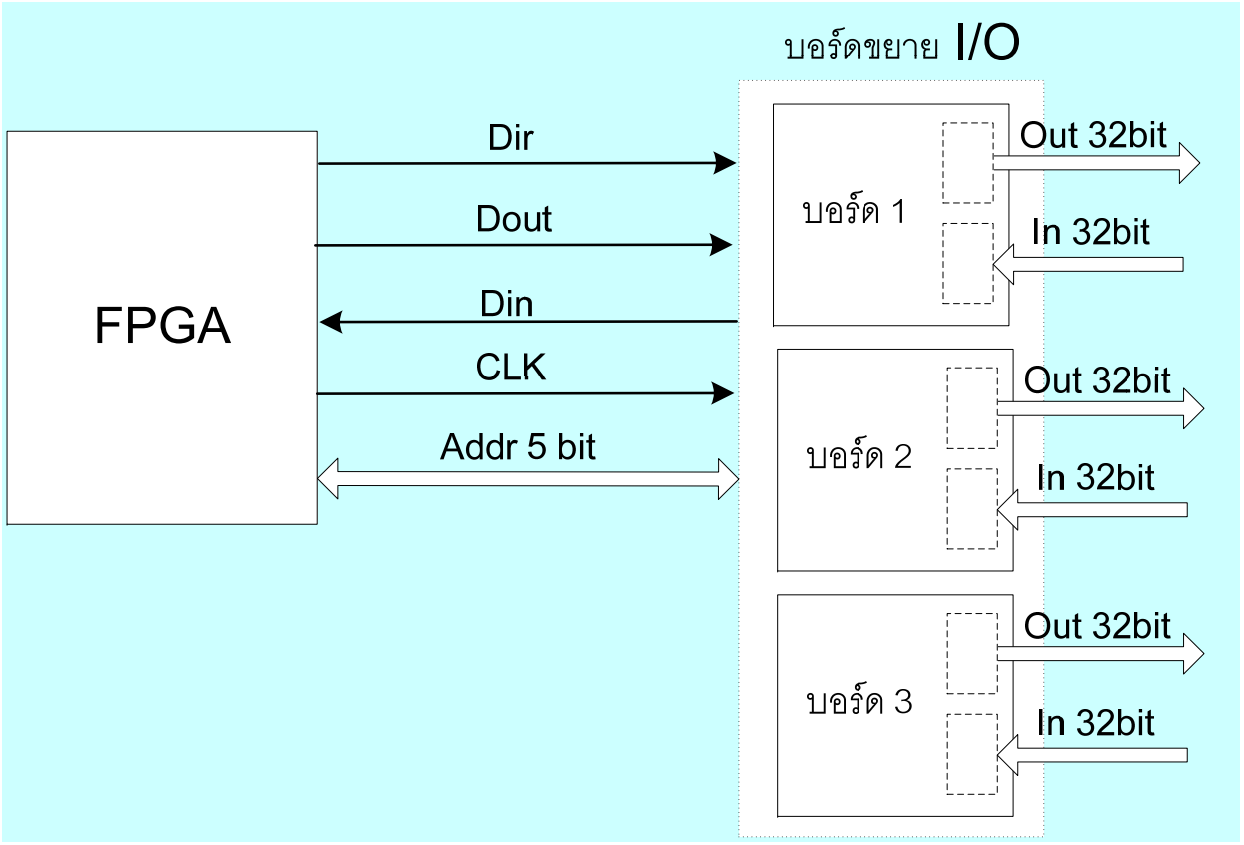
$$+ k_{vf} [r(n) - r(n-1)]$$

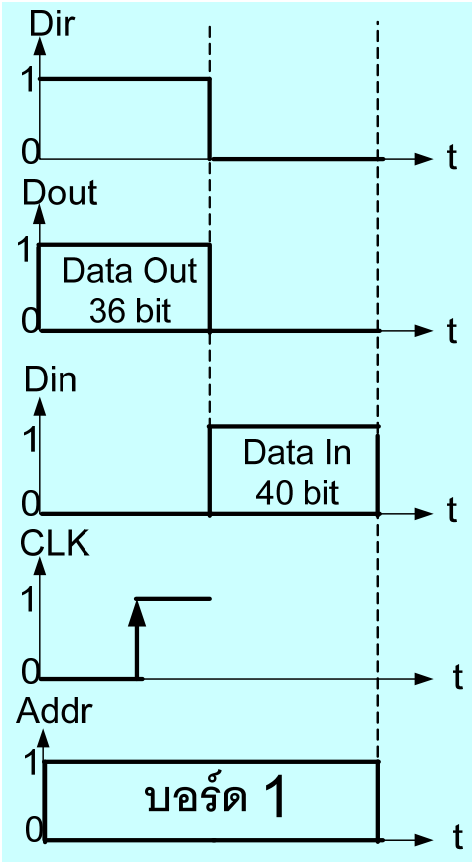
$$e(n) = r(n) - EncCount(n)$$

- PID-Feedforward , velocity feedforward
- Gain register 16 bit, Error register 16 bit, Ref register 32 bit

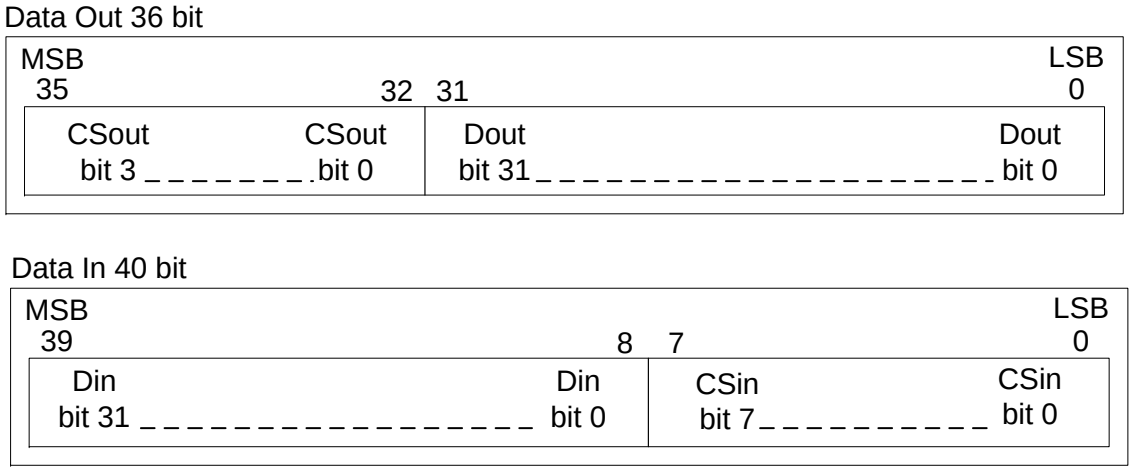
  Control register 12 bit
- ตัวควบคุม PID เขียนโดยคำนึงถึงความเร็วของการทำงานเป็นหลัก ไม่คำนึงถึงปริมาณเกทที่ใช้

การเพิ่มจำนวนช่องสัญญาณ I/O ด้วยการเชื่อมต่อแบบอนุกรม





การเพิ่มจำนวนช่องสัญญาณ I/O ด้วยการเชื่อมต่อแบบอนุกรม



- Data Out => FPGA send Data to I/O Board
- Data In => FPGA get Data from I/O Board
- First bit from MSB

การเพิ่มจำนวนช่องสัญญาณ I/O ด้วยการเชื่อมต่อแบบอนุกรม

```
 1  CSout(0)  := Dout(28) xor Dout(24) xor
 2               Dout(20) xor Dout(16) xor
 3               Dout(12) xor Dout(8)  xor
 4               Dout(4)  xor Dout(0);
 5  CSout(1)  := Dout(29) xor Dout(25) xor
 6               Dout(21) xor Dout(17) xor
 7               Dout(13) xor Dout(9)  xor
 8               Dout(5)  xor Dout(1);
 9  CSout(2)  := Dout(30) xor Dout(26) xor
10               Dout(22) xor Dout(18) xor
11               Dout(14) xor Dout(10) xor
12               Dout(6)  xor Dout(2);
13  CSout(3)  := Dout(31) xor Dout(27) xor
14               Dout(23) xor Dout(19) xor
15               Dout(15) xor Dout(11) xor
16               Dout(7)  xor Dout(3);
```

- CSout คือบิทตรวจสอบความถูกต้องของข้อมูลด้านขาออก
- Dout คือข้อมูลที่ FPGA ส่งให้กับบอร์ดขยาย I/O

การเพิ่มจำนวนช่องสัญญาณ I/O ด้วยการเชื่อมต่อแบบอนุกรม

```
 1  CStemp(0) := Din(8) xor  Din(16) xor Din(24) xor Din(32);
 2  CStemp(1) := Din(9) xor  Din(17) xor Din(25) xor Din(33);
 3  CStemp(2) := Din(10) xor Din(18) xor Din(26) xor Din(34);
 4  CStemp(3) := Din(11) xor Din(19) xor Din(27) xor Din(35);
 5  CStemp(4) := Din(12) xor Din(20) xor Din(28) xor Din(36);
 6  CStemp(5) := Din(13) xor Din(21) xor Din(29) xor Din(37);
 7  CStemp(6) := Din(14) xor Din(22) xor Din(30) xor Din(38);
 8  CStemp(7) := Din(15) xor Din(23) xor Din(31) xor Din(39);
 9  CSin(0) <= CStemp(0) xor RDData40b(0);
10  CSin(1) <= CStemp(1) xor RDData40b(1);
11  CSin(2) <= CStemp(2) xor RDData40b(2);
12  CSin(3) <= CStemp(3) xor RDData40b(3);
13  CSin(4) <= CStemp(4) xor RDData40b(4);
14  CSin(5) <= CStemp(5) xor RDData40b(5);
15  CSin(6) <= CStemp(6) xor RDData40b(6);
16  CSin(7) <= CStemp(7) xor RDData40b(7);
```

- CSin คือบิทตรวจสอบความถูกต้องของข้อมูลด้านขาเข้า
- Din คือข้อมูลด้านขาเข้าจากบอร์ดขยาย I/O สู่ FPGA
- ตรวจสอบความถูกต้องของข้อมูลด้านขาเข้า
  - ถ้าหาก Csin เป็นศูนย์หมดทุกบิท FPGA จึงจะรับข้อมูลเข้าไปได้

หน้าที่การทำงานอื่น ๆสำหรับควบคุมเครื่องจักร CNC

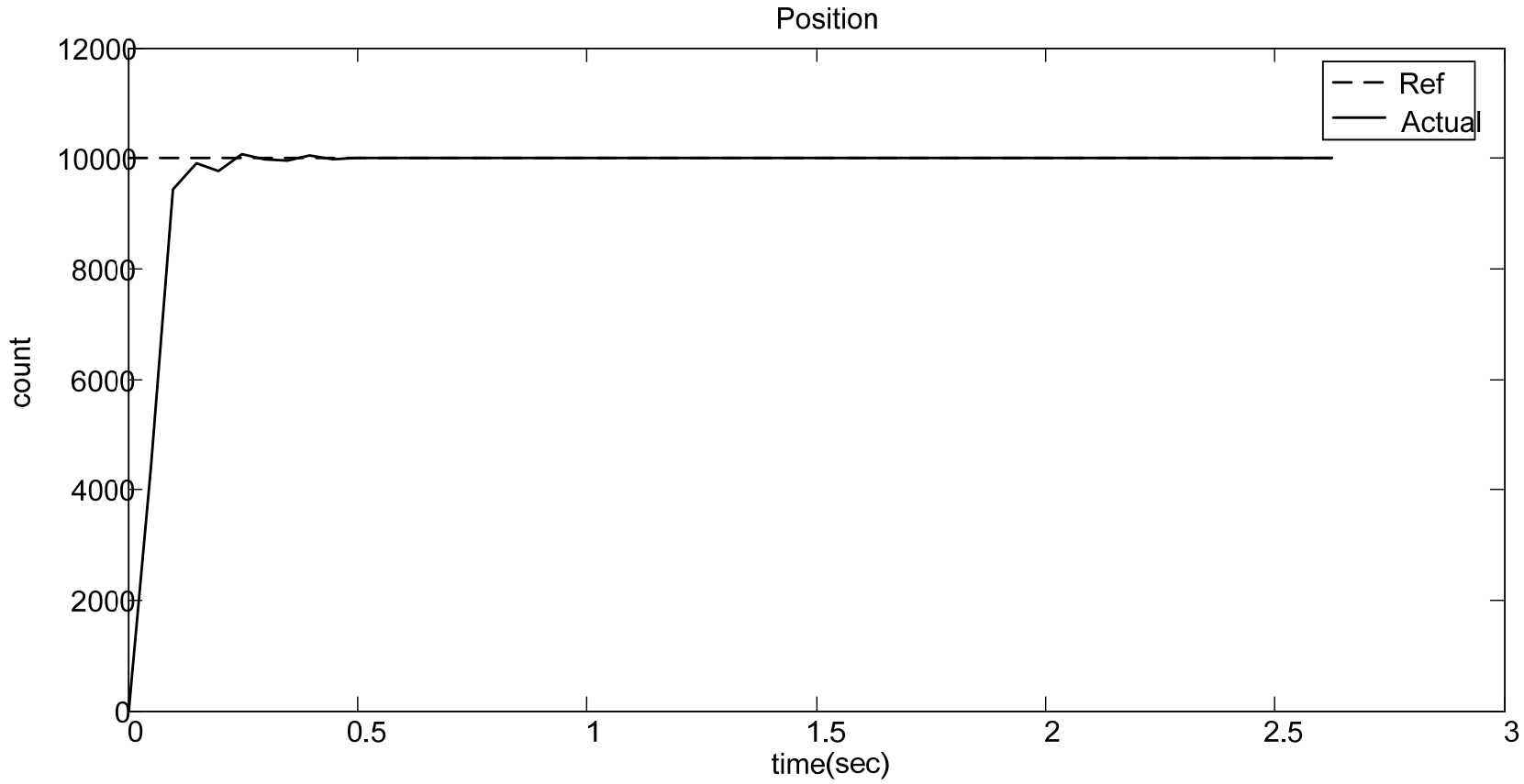
- สัญญาณควบคุมมอเตอร์แบบ PWM
- Capture Encoder by Hard Limit Switch Trigger
- Soft Limit Encoder
- Hard Limit Switch register
- Emergency Switch
- Interrupt Register
- Enable Module
- PID Error Limit

ผลการทดลอง

- PID-Feedfoward
- Unit-Step Input, Ref=10,000
- DC motor-Faulhaber 12 v
- Encoder 500 พัลส์ต่อรอบ, อัตราทดเกียร์ 14
  ความละเอียดของตำแหน่งที่ปลายมอเตอร์เมื่อผ่านอัตราทดเกียร์เท่ากับ 28000 ตำแหน่งต่อรอบ
- Sampling Time ของระบบควบคุม เท่ากับ 10 us
- Sampling Time ของการเก็บค่าแสดงผลทุกๆ 49.5 ms
- Gain value

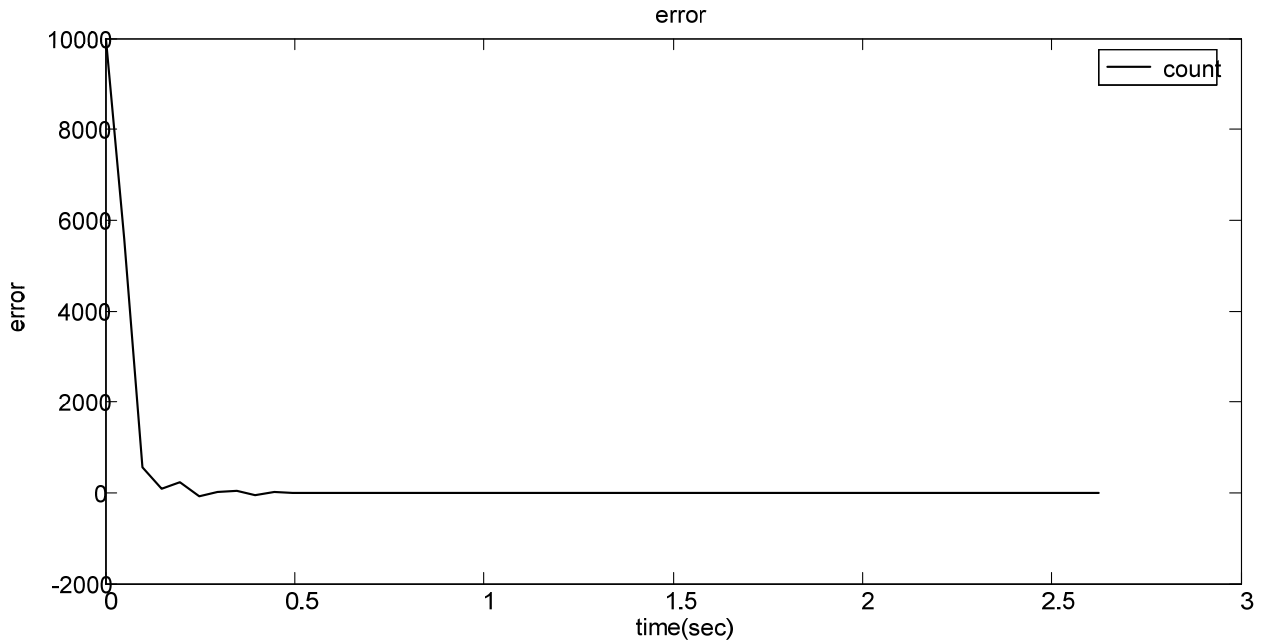
$$k_p = 24000, k_i = 5, k_d = 200, k_{vf} = 100$$

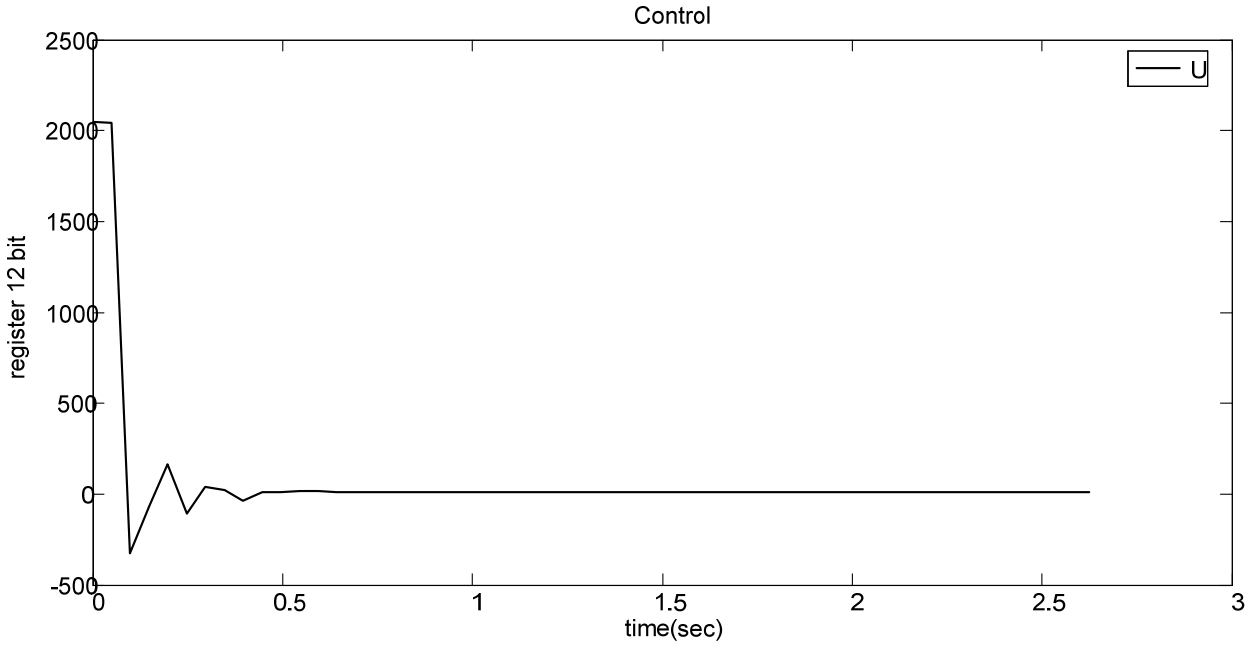
ผลการทดลอง



- Compare Reference vs Actual Value
- เวลาที่ใช้ในการเข้าสู่สภาวะทรงตัวมีค่า 500 ms โดยประมาณ

ผลการทดลอง


error
count
error
time(sec)
10000
8000
6000
4000
2000
0
-2000
0
0.5
1
1.5
2
2.5
3


Control
U
register 12 bit
time(sec)
2500
2000
1500
1000
500
0
-500
0
0.5
1
1.5
2
2.5
3

สรุป

- การควบคุมมอเตอร์โดย FPGA ด้วยตัว
  ควบคุม PID สามารถลดการคำนวณและเวลาในการทำงานของตัว
  ประมวลผลหลักได้
- โมดูลการทำงานต่างๆของ FPGA ทำงานแบบขนาน
  ทำให้ไม่สิ้นเปลืองเวลาในการคำนวณและทำงานด้วยความเร็วสูง
  ทำให้สามารถรองรับการทำงานเครื่องจักร แบบหลายแกนการเคลื่อนที่
  ที่ต้องการความเร็วในการควบคุมระบบที่สูงได้
- โมดูลเพิ่มจำนวนช่องสัญญาณและโมดูลการทำงานอื่นๆใน FPGA
  ช่วยให้งานควบคุมเครื่องจักร CNC มีประสิทธิภาพมากขึ้น