제어 IC 및 이를 이용한 자동문(Auto Door) 제작

권경민* · 허창우*

Automatic Door Using Control IC

Gyoung-min Kwon*, Chang-wu Hur*

요 약

본 논문은 Pyroelectric Sensor의 신호를 입력으로 받아 Stepping Motor 제어신호를 출력하는 Controller를 VHDL을 이용하여 ALTERA사의 EPM7128SLC84-7 CPLD칩으로 구현하여 자동문을 제작하였다. 동작은 센서부에서 사람을 감지하여 출력한 신호를 CPLD에서 받아들여 Stepping Motor 구동부 회로의 제어신호로 사용하여 문을 동작시켰다.

Ⅰ. 서 론

현대에는 기술의 발달로 모든 것이 자동화되어가고 있는 추세이다. 문도 마찬가지로 자동화되어 사람이 손으로 열지않고 사람의 유무를 판단하여 자동으로 열리고 닫히는 문이 제작되고 있다.

요즘은 모든 분야가 세분화 되어 가고 있다. 따라서, 각각의 목적에 부합되는 컨드롤러 제작에 있어서 표준 집적회로나 마이크로 집적회로를 사용하여 제작하는 데는 많은 시간과 엄청난 비용문제가 발생한다. 이를 개선하기 위해 본 작품. 자동문을 만들어 봄에 있어 Stepping Motor를 제어할수 있는 IC를 빠른 설계와 구현이 가능하고 설계상의 문제도 쉽게 수정할 수 있는 사용자 프로그램 IC(Programmable Logic Device)인 CPLD를 이용하고 Program Language는 VHDL (Very high speed integrated circuit Hardware Description)을 사용하였다. VHDL은 1980년대 초부터 개발이 시작되어 1986년에 IEEE 표준 VHDL이 탄생하였다. VHDL은 우리가 흔히 사용하고 있는 고급 프로그래밍 언어와 같은 원리에 기초를 두고 있으며 하드웨어적인 특징을 추가하여 표현 능력이 뛰어나며 시스템 레벨에서부터 게이트 레벨까지의 하드웨어의 동작적 및 구조적 기술이 가능하다. 하지만 아직까지 Programmable Logic Device가 고가라는 단점이 있다. 자동문 구현은 Pyroelectric 센서와 스테핑 모터를 사용하여 문을 동작시켰다.

Ⅱ. 구현 방법 및 시뮬레이션

1. VHDL로 표현

1) VHDL Graphic Editer 표현

센서의 출력값을 입력으로 받아 Stepping Motor와 7segment 6개를 제어할 수 있는 Controller를 설계하였다.

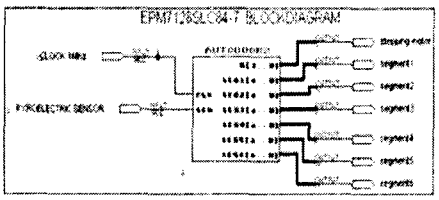


그림1. 자동문 컨트롤러 블록도

그림 1.은 Graphic Editer을 이용한 컨트롤러의 표현이다. 그림에서 보듯이 Clock과 센서의 입력을 가지고 원하는 출력값을 발생시키는 기능을 하는 IC를 구현하였다.

2) VHDL PROGRAM

프로그램의 기본적인 알고리즘은 센서에서 출력이 high이면 IC에서는 순방향으로 모터를 구동시키는 출력비트를 일정시간동안 출력하고 low이면 역방향으로 모터를 구동시키는 출력비트를 출력한다. 프로그램에서는 5개의 PROCESS문으로 구성하였는데 주파수 분주부, DOOR 상태 결정부, OPEN TIME 표시부, DOOR 상태 표시부, 모터 제어 비트 발생부로 나누어서 동기를 맞추에 출력값을 발생하도록 프로그램을 구현하였다.

아래 그림과 같이 구성되는 다섯 부분의 블록의 프

로그램은 다음과 같다.

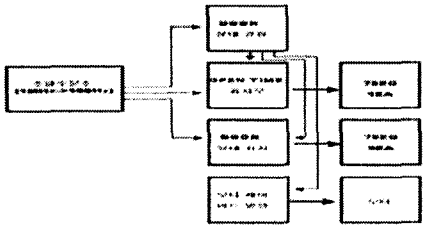


그림2. 프로그램 블록도

(1) 주파수 분주부

```
process(clk, cnt)
 begin
    if clk'event and clk='1' then
       cnt <= cnt + 1;
       if cnt = 10000 then
          clken <= '1';
       else
          clken <= '0';
       end if;
    end if;
end process;
```

위 process구문에서 사용되는 신호로 port 구문에서 선언한 외부 주파수 입력 신호인 clk와 분주의 내부 신호인 cnt를 사용하여 cnt를 외부 clock마다 count하여 다른 process 구문에서 사용하기 위한 새로운 clock 신호인 clken을 발생시켰다. 위와 같이 주파수 분주를 하는 이유는 Stepping Motor에 너무 빠른 제어 신호가 입력되면 Stepping Motor가 회전을 하지 못하고 제자리에서 진동하기 때문에 주파수 분주를 하였다. 그림3은 주파수 분주부의 simulation 결과이다. 단. simulation에서는 카운터값을 프로그램에서의 1에서 10,000까지가 아닌 편의상 1에서 10까지 하였다.

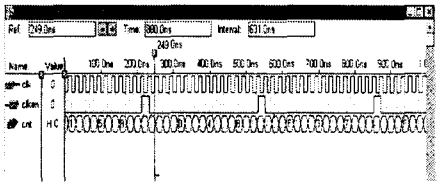


그림3. 주푸수 분주 Simulation

(2) DOOR 상태 결정부

센서에서 출력값이 계속 유지가 되지 않으므로 door의 open 상태와 close 상태를 유지하는데 필요한 새로운 신호를 만들어 모터를 제어하는 i 값을 제어한다.

```
process(stop_count, clken, close2, sen, i,
door_count, op, clken, close, segment)
   begin
```

위에서 설명한 것과 같이 내부 신호로 door를 open하는 신호인 op, door의 open 상태에서 정지상태를 유지하는 신호인 close, door를 close하는 신호인 close2를 설정하였다.

```
if clken'event and clken='1' then
   if sen='1' then
      op <= '1';
      close2 <= '0';
      close <= '0';
    segment <= '1';
```

센서의 값에 따라 door가 열리는 동작을 하도록 op 신호를 high로 만들어주고 닫히는 동작을 하는 close2와 door가 정지되어있는 동작을 제어하는 close는 low로 만들어준다. door 상태를 표시하는 7segment의 상태를 정의한다.

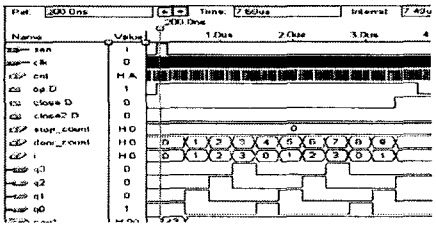


그림4. Door Open 동작의 Simulation

```
elsif close = '1' then
   op <= '0';
   if stop_count = 450 then
      stop_count <= 0;
      close2 <= '1';
      close <= '0';
    segment <= '0';
   else
      stop_count <= stop_count + 1;
   end if;
end if;
```

door가 열린 상태에서 일정시간 동안 멈추어져 있는 상태를 제어하는 구문으로 정수값을 count하여 그 변수값이 만족하였을 경우 door를 닫기 위하여 close2 신호를 high로 만들어준다. door의 상태를 표시하는 7segment 신호를 변화시킨다.

stop_count는 문이 열린상태로 정지해 있는 시간을

유지하기 위하여 count 해주고 count가 종료되면 close2값을 high로 만들어 주고 열리는 시간을 제어하는 close신호를 low로 만들어 준다.

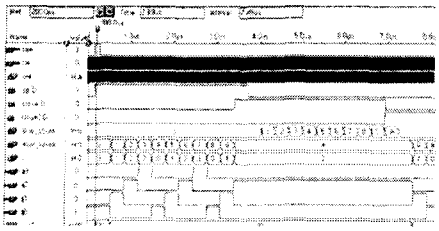


그림5. Open 상태 유지 Simulation

(3) DOOR OPEN부

```
if op='1' then
   i <= i + 1;
door_count <= door_count + 1;
      if door_count = 128 then
         close <= '1';
         i <= i;
         door_count <= 128;
      end if;
```

door를 open시키라는 신호인 op신호가 high가 되었을 때 door를 열기 위하여 stepping motor의 구동비트인 i를 count하여 발생시킨다. door의 이동거리를 조정하기 위해서는 door_count라는 내부 신호를 count하여 거리를 조정한다.

(4) DOOR CLOSE부

```
      elsif close2='1' then
         i <= i - 1;
         door_count <= door_count - 1;
         if door_count = 0 then
            i <= i;
            door_count <= 0;
            close2 <= '0';
```

- 생 략 -

door를 close하는 구문으로 close2신호가 high가 되었을 때 door가 open될때의 i값을 역으로 count하여 stepping motor 구동비트를 반대로 발생시킨다. 이동거리는 open 동작때와 마찬가지로 door_count 값으로 제어하여 같게 만들고 door의 상태를 제어하는 close2 신호를 low로 만들어 다음 상태에 대기한다.

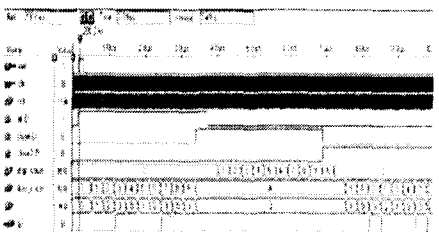


그림6. 상태신호에 따른 i값 변화 Simulation

(5) OPEN TIME 표시부

```
process(stop_count, clken)
begin
 if (clken'event and clken = '1') then
      if stop_count = 0 then
            seg6 <= "1000000";
      elsif stop_count = 410 then
            seg6 <= "0001001";
      elsif stop_count = 320 then
            seg6 <= "0101100";
      elsif stop_count = 230 then
```

- 생략 -

위 구문은 door가 open 상태에서 정지하여 있는 시간을 7segment로 표시하여 주는 구문으로 stop_count 값에 따라 5에서부터 1까지 숫자를 7segment 하나로 표시하였다. 시간은 우리가 사용하는 시간이 아닌 stop_count 값에 동기를 맞추었다.

아래의 Simulation에서 보듯이 close 신호가 high로 되었으때 stop_count 신호가 count 되고 count 값에 따라 7segment에 정지되어있는 시간을 나타내는 값을 구문에 따라 seg6에 출력하였다. close 신호값이 high로 되었을때 stop_count 값이 바로 변화하지 않고 약간의 지연이 발생하였다.

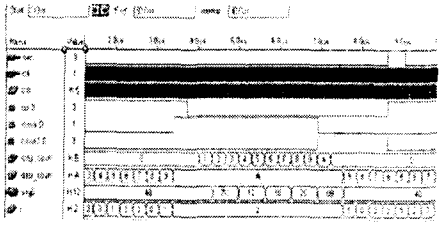


그림7. Open Time Simulation

(6) DOOR 상태 표시부

```
process(segment, clken)
begin
if (clken'event and clken = '1') then
```

```
        if segment = '0' then
        seg1 <= "1000011";
-생략-
        elsif segment = '1' then
        seg1 <= "1000000";
-생략-
        else
        seg1 <= "1111111";
-생략-
```

door 상태를 표시하는 구문으로 segment 신호의 상태에 따라 다섯 개의 7segment에 "open", "close"라는 글씨를 표현하였다. 마지막 else구문은 초기값이나 segment 상태가 불안정하였을 경우 를 위하여 작성하였다.

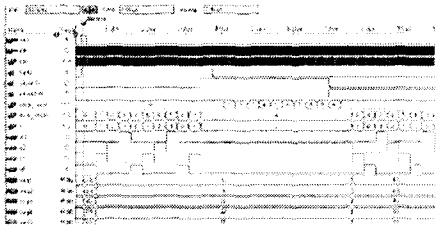


그림8. Door 상태 표시 Simulation

(7) STEPPING MOTOR 제어 비트 발생부

```
process(i)
begin
   case i is
      when 0 => q <= "0001";
      when 1 => q <= "0010";
      when 2 => q <= "0100";
      when 3 => q <= "1000";
      when others => q <= "0000";
   end case;
end process;
```

위 구문에서는 i 신호가 0부터 3까지 count 되면서 Stepping Motor를 제어하는 4비트로 출력된다.

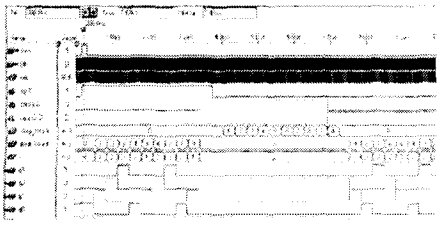


그림9. Stepping Motor 제어 비트 발생 Simulation

2. 하드웨어 제작

전체적인 하드웨어의 회로는 크게 센서부, Controller부, Stepping Motor 구동 회로부, 7segment display부로 나뉘어진다. 회로를 제작할 때는 Controller부와 Stepping motor부를 하나의 기판에 구성하였다.

1) controller제작

위에서 설명한 VHDL로 작성한 프로그램을 AlTERA사의 CPLD인 MAX7000 Series EPM7128SLC84-7을 사용하여 Controller를제작하였다. EPM7128SLC84-7의 제원과 구현시 사용한 사항은 표1과 같고

	제원	사용량
Usable Gates	2,500	1,760
Macrocells	128	88
Logic array blocks	8	
PINS	84	67

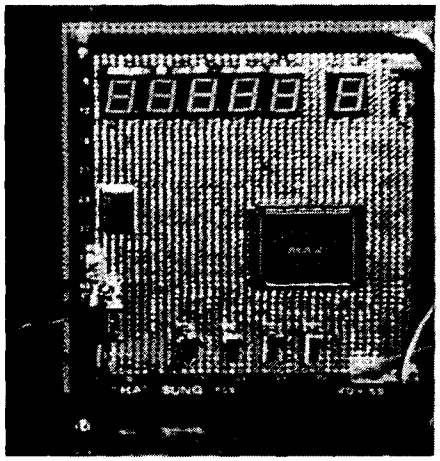


그림10. Controller, Motor 구동, Display회로

그림10에서 보는 것과 같이 하드웨어는 OSC (1MHz)와 EPM7128SLC84-7, 7segment 6EA, Stepping Motor 구동을 위한 Darlington 회로를 사용하여 센서의 출력에 따라 Motor 구동 비트를 발생하는 회로를 구성하였다.

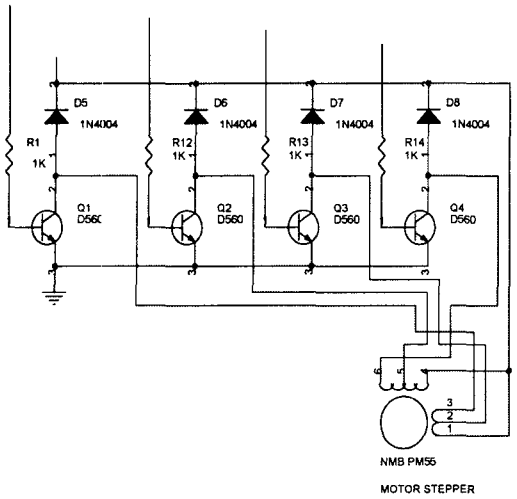


그림.11 Stepping Motor 구동부 회로도

2) 센서부 제작

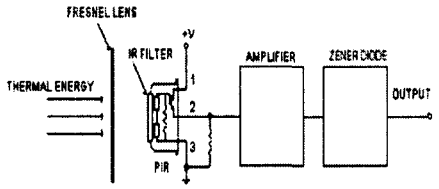


그림12. 센서부 구성도

센서부는 그림12에서 보듯이 크게 3개의 부분으로 나누어져 있다. 센서와 신호 증폭부, 출력전압 안정부로 나뉜다.

(1) 센서

적외선 센서의 일종인 Pyroelectric센서와 동물과 사람에게서 나오는 9.4mm대의 파장을 감지하기 위하여 필터를 사용하였다.

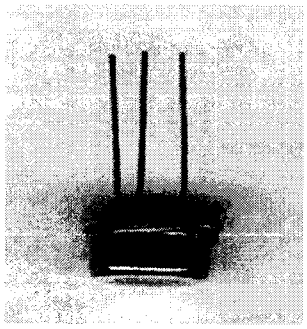


그림13. Pyroelectric Sensor

센서는 적외선을 받았을 때 crystalline에서 표면 전하가 생성되어 전계효과 트랜지스터에 의해 측정되는 구조를 가지고 있다. 그림12에서는 센서의 실물을 보였다.

(2) 신호 증폭부

센서에서 나오는 신호의 전압레벨이 너무 낮아서 OP-AMP를 사용하여 증폭 하였다.

(3) 출력 전압 안정부

센서에서 출력되는 전압이 controller에 입력될 때 과전압 방지를 위하여 제너다이오드를 사용하여 controller의 입력으로 사용하였다.

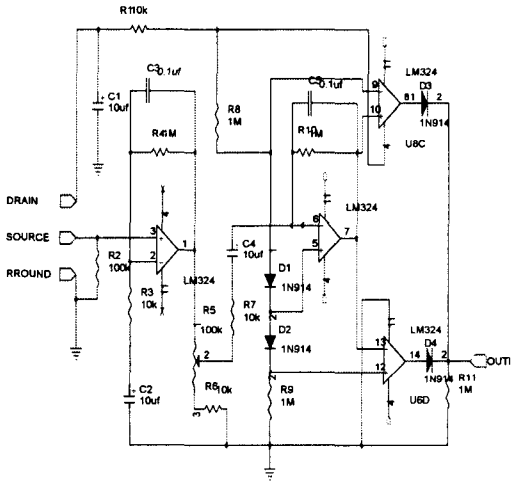


그림14 센서부 회로도

센서의 동작 전압이 3V부터 15V까지 동작하지만 센서의 동작의 상태와 수명을 고려하여 결정하였고 전압을 높이면 출력전압도 높아지므로 controller의 입력 전압 5V에 맞추어 입력을 6V로 하였다.

III. 결 론

Altera사의 CPLD인 EPM7128SCLC84-7을 이용하여 자동문을 제작하였다. 사람이나 동물의 몸에서 방출하는 적외선을 감지하기 위하여 필터를 부착한 Pyroelectric Sensor의 출력값을 CPLD의 입력으로 받아서 Stepping Motor를 구동하였다. 기본 동작은 문이 열리고 일정시간 유지한 후 닫히는 동작하고 닫히는 도중 센서에서 다시 신호가 출력되면 닫히는 동작을 중지하고 다시 닫힌 거리만큼 열리는 동작을 하는 방식을 사용하였다. 문제점으로는 7segment를 State 방

식으로 제어를 하였는데 여기서 CPLD의 PIN을 과다 사용하여 gate수가 늘어남을 볼 수 있었다. 또한 Stepping Motor을 구동하려고 clock을 분주하여 사용하다 보니 주파수 분주 부분에서 또한 많은 gate를 소비하였다. 입력 주파수를 낮은 주파수로 교체하고 7segment수를 줄이는 과정을 통하여 gate수는 2,000 gate 정도 줄일 수 있었다. 이는 최종사용 gate 수보다 많은 양이다. 하지만 앞으로도 7segment 제어 방식이나 door의 이동거리를 count 값으로 제어하지 않고 limit 스위치를 사용하면 count를 하지 않아도 되므로 더 많은 gate 수를 줄일 수 있을 것이다. 이는 gate가 적은 Device를 사용하므로 제작비용에도 영향을 미칠 것이다.

본문에 따른 구현은 자동문이었지만 여러 다른 Controller를 VHDL로 제작한다면 단기간에 적은 비용으로 다른 여러 분야에서 사용되는 전용 IC를 제작할 수 있을 것이다. Pyroelectric Sensor 또한 문이 아니라 방범시스템등에서 요긴하게 사용되어질 것이다.

참고문헌

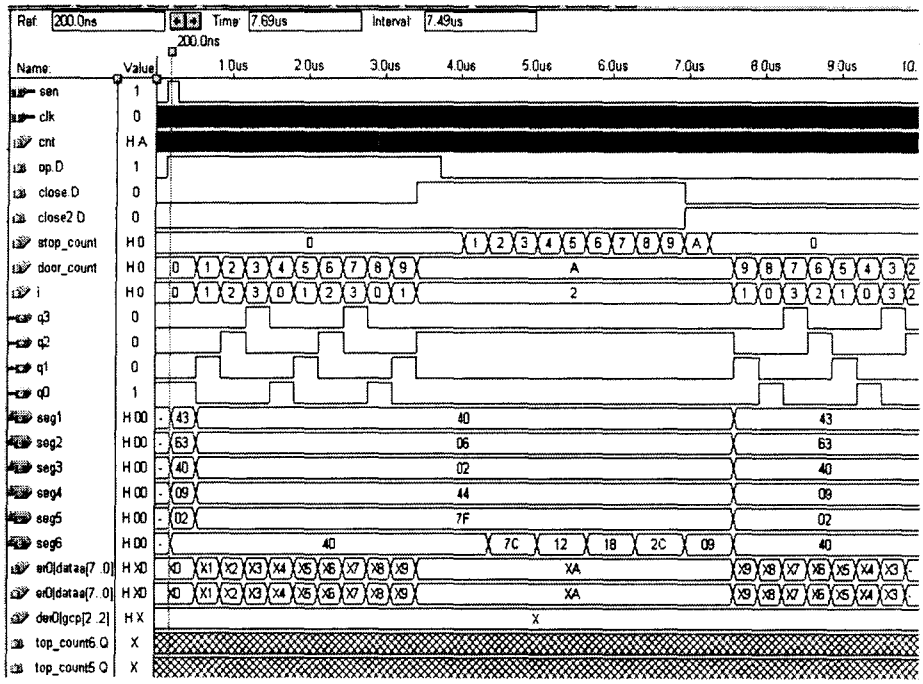
[1] 박세현 "VHDL 기본과 활용" (1998) 그린

[2] 김영철, 정연모, 조중휘, 홍윤식, "디지털 시스템 설계를 위한 VHDL" (1998) 홍릉과학출판사.

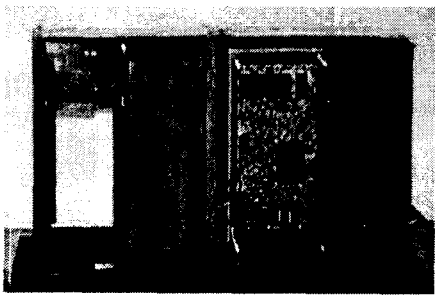
[3] 타카시 젠조, 아키라 스가와라, "스테핑 모터 & 마이컴 제어" (2000) 일진사.

[4] Cohen Ben, "VHDL coding styles methodologies"(1999) Kluwer Academic Publishers.

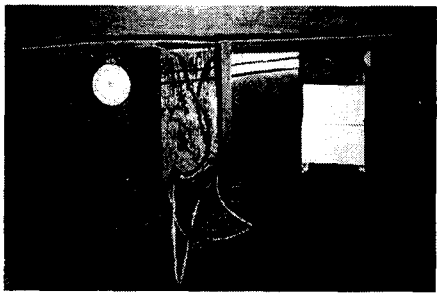
V. 별 첨



전체 Simulation 결과



작품사진(정면)



작품사진(후면)