**논리설계 프로젝트**

2018-13627 김기덕

1. 구현방식

우선 스위치와 시간입력을 Debouncer, PulseGenerator, ClockModulator로 입력을 분명하게 바꾸어 실제로 사용되는 input값의 역할을 했다.

프로젝트에서 필요한 기능을 크게 시계, 알람, 스탑워치로 분류를 하였고 우선 각각을 따로 구현하였다. 그 후, 이 셋을 합치기 위해 StateMachine에서 input값에 따른 state를 설정해 특정 기능을 활성화 하도록 설계하였다. 이렇게 input값으로 각각의 기능들은 돌아갈텐데, 마지막으로 각각의 output을 현재 state들과 input값에 따라 선택적으로 원하는 값을 출력할 수 있도록 하는 Output Logic을 구현하였다.

이렇게 뼈대를 구성하고 Output Logic에서 나오는 Output을 보드에 맞는 값으로 바꿔줄 translate모듈을 마지막으로 통과시켜 마무리하였다.

2. 모듈 설명

1) Clock

State :

S0 : 초기화

S1 : 시계 기본상태에서 24시간 표시법

S2 : 시계 기본상태에서 24시간 표시법

S3 : S1에서 set을 눌렀을 때/시 설정/24시간

S4 : S2에서 set을 눌렀을 때/오전오후설정/12시간

S5 : S3에서 set을 눌렀을 때/분 설정/24시간

S6 : S4에서 set을 눌렀을 때/시 설정/12시간

S7 : S6에서 set을 눌렀을 때/분 설정/12시간

이후 S5에서 set을 누르면 S1, S7에서 set을 누르면 S2로 가게 하였다.

그리고 S1과 S2사이는 서로 display가 0일때는 S1, 1일때는 S2로 움직일 수 있게 하였다.

출력값은 유효한 state(시간설정 하는동안)에서 up, down을 눌렀을 때 해당하는 값을 바꾸었고, S1과 S2일때는 시계를 작동시켜 시간이 흐르도록 출력하였다.

2) Alarm

State : Clock과 S0만 빼고 전부 동일, Clock과 달리 reset 이외에도 aclear를 받으면 S0로 간다.

Clock과 거의 똑같지만 Clock의 S0는 딱히 의미가 없는 반면에 Alarm의 S0는 알람이 없을때를 표시해주는 상태를 갖고 있다. 알람 클리어 버튼이 유효할 때 그 버튼을 누를시 S0상태로 돌아가게 된다. 또, 알람에서 맞춰놓은 시간은 흐르는 시간이 아니기 때문에 Clock과는 다르게 항상 시간이 흐르지 않게 했고, 나머지는 Clock과 동일하게 하였다.

3) StopWatch

State :

S0 : 스탑워치가 멈춘 상태

S1 : 스탑워치가 진행되고 있는 상태

S0에서 set을 누르면 S1으로, S1에서 set을 누르면 S0로, 어디서든 clear를 누르면 S0로 이동하며 값을 모두 0으로 초기화하였다.

4) StateMac(State Machine)

Clock, Alarm, StopWatch중 자신이 지금 어떤 모듈을 컨트롤하고 있는지를 State로 나타내었다.

State:

S0 : Clock을 컨트롤 중

S1 : Alarm을 컨트롤 중

S2 : StopWatch를 컨트롤 중

각각의 state에서 mode0, mode1을 누르면 전/후 state로 넘어갈 수 있게 구현을 하였다.

추가로 display 상태를 저장했는데, 내부에 dp 레지스터를 만든 뒤 display버튼에서 입력을 받을 때 그 state에서 그 display버튼이 DigitalWatch의 display변화를 의도했는지를 판단해서 dp를 변화시켰다. 그 이외의 경우(StopWatch를 작동중이거나 시간설정 중인경우)는 dp값을 변경하지 않았다. 이렇게 저장한 dp값을 output과 연결시켜 Clock, Alarm에서 받아 상태를 표시할 수 있게 했다.

5) OutputLogic(Output Logic)

앞서말한 4개의 모듈의 state, output인 시/분/초와 같은 값을 받아들이고, 그와 더불어 인풋들을 추가로 받아 최종 Output을 결정하는 Logic을 말한다.

Clock, Alarm, StopWatch모두 각각의 상태에 따른 output이 있을텐데, 우리가 보고있는 화면은 하나이므로 StateMac의 state에 따라 활성화된 모듈을 판단하여 그 값을 10진수와 AM/PM등으로 각 segment가 어떤 값을 가져야 하는지를 출력하였다. 또, 점멸 레지스터를 선언해 각 모듈의 state에 따라 점멸이 필요하면 필요한 위치에 점멸을 알리는 output도 같이 내었다.

6) Translate(Final Display)

OutputLogic에서 받은 최종 output과 점멸상태를 받아와서 그 값에 맞게 sevensegment에 출력을 하였다.

7) DigitalWatch(Union 모듈)

clk을 ClockModulator에, 스위치 input값을 Debouncer, PulseGenerator에 통과시켜 인풋값의 노이즈가 없도록 조정한 뒤 위에서 구현한 6개의 모듈을 넣어 연결시켰다.

3. 추가구현

1) 밝기

DIP스위치 첫번째 값을 밝기로 받아 스위치를 위로 올리면 쭉 1을 출력하던 것을 1반 0반으로 끊어서 짧은 주기로 출력을 해서 밝기를 절반가량으로 줄였다. 이 기능은 Translate모듈에서 light인풋을 받아 light가 1이면 위와 같이 실행되도록 하였다.

2) 비밀번호

우선 핸드폰의 잠금처럼 on/off버튼을 DIP 스위치 두번째 값으로 받고 DigitalWatch가 켜져있을 때 on/off버튼을 내려 off상태로 바꾸면 화면에 LOCKED가 뜨며 스위치를 눌러도 반응하지 않는 상태가 된다. 이 시계의 LOCK을 해제하려면 on버튼을 올리며 비밀번호를 눌러야 되는데 비밀번호는 우선 DIP세번째 스위치로 하였다. On 버튼을 올리고 이 스위치를 올려야 LOCK이 해제가 된다. 비밀번호를 한자리로 해서 그렇지만 여러자리로 하고 if문으로 엮으면 여러자리도 쉽게 확장할 수 있다. 이를 구현하기 위해 우선 보드의 on/off 상태를 정해야 하는데 이는 on\_input과 비밀번호를 input으로 받아 on\_input이 0이 되면 off로, on\_input이 1이고 비밀번호가 맞으면 on으로 출력이 되는 모듈을 만들었다. 그 후 여기서 나온 On과 input을 받았을 때, on이 1이면 input을 그대로, 0이면 0을 출력하는 모듈을 만든 뒤(mux) 각 input에 스위치의 값을 mux에 같이 넣은 뒤 모듈에 실제로 유효할 input 값을 뽑아내었다. 그리고 Translate에서 on/off값을 입력받아 off면 차례대로 L O C K E D를 표현하도록 구현을 하였다.