Computer Organization (001)

Lab02-RTL and Sequential Logic Report

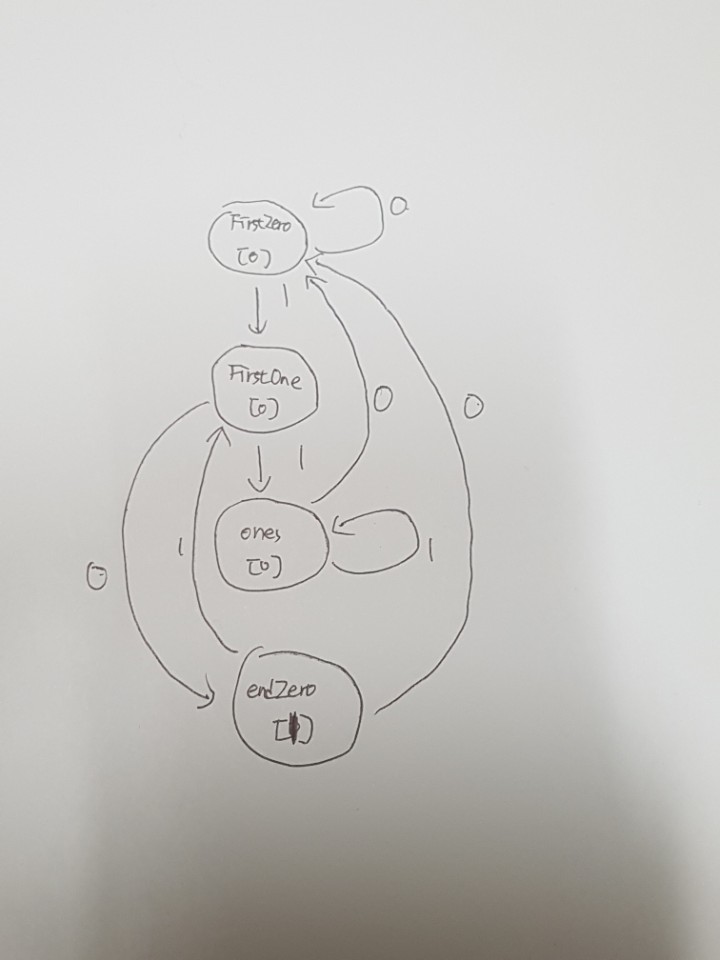
전기정보공학부

2017-17497 김현규

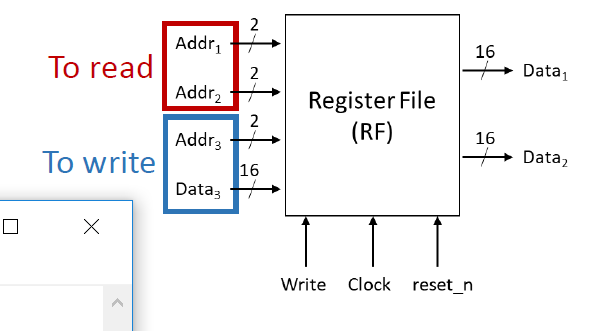
1. Introduction

RTL이 어떠한 것인지 이해하고 Verilog를 통해 sequential logic을 구현하는 방식에 대해 배운다. FSM의 두가지 타입인 moore machine 과 mealy machine을 이해하고 한 방식을 선택하여 010 Detector을 구현한다. 또한 16-bit 2-read/1-write register file을 구현하는 것이 이번 lab의 목표이다.

1. Design
2. 010 Detector

모듈의 인풋으로는 CLK와 in이 있고 아웃풋으로는 하나의 reg인 out이 주어져 있다. 클럭의 posedge마다 FSM의 상태를 변화시키면서 동작하도록 구현되었다.총 4개의 state가 있으므로 2비트 크기의 레지스터를 통하여 상태를 저장하고 case문을 통해 각 state를 올바르게 구현하였다.

FirstZero는 010에서 맨 앞에 있는 0이 등장한 상태이고 endZero는 010에서 마지막 0으로 output이 1이 되는 경우이다. FirstOne의 경우에는 바로 직전0이고 현재는 1인 상태이며 ones는 직전에도 1이 나온 상태를 의미한다. 또한 초기 상태를 Firstzero가 아닌 ones로 설정해야 한다는 것에 주의하도록 디자인하였다.

1. Register File

위와 같은 형태의 register file을 구현하였다. Reset\_n의 값에 따라서 다른 동작을 하는데 이 값이 low인 경우에는 모든 register의 값을 0으로 바꾼다. High 인 경우에는 Addr1, Addr2의 주소에서 값을 읽은 후 Data1, Data2에 저장하고 Write 시그널도 High 인 경우에만 Addr3에 Data3을 쓰는 일을 한다.

Addr이 2bit이므로 RF내부에는 4개의 register가 존재하며 각 레지스터는 16비트이다. Read의 경우에는 asynchronous 해야 하므로 addr1과 addr2이나 reset\_n이 바뀌는 즉시 data가 변하도록 디자인하였다. Write의 경우에는 clock의 다음 posedge에서 data3에 쓰이도록 구현하였다.

1. Implementation
2. 010 Detector

|  |  |  |  |
| --- | --- | --- | --- |
| 상태 | 의미 | 0 | 1 |
| firstZero | in의 값이 0인데endZero상태가 아닐 때이다. | firstZero | firstOne |
| firstOne | in이 값이 1이고 이전에 등장한 값이 0인 상태이다. | endZero | Ones |
| Ones | in의 값이 1이고 이전에 등장한 값이 1인 상태이다. | firstZero | ones |
| endZero | 010에서 마지막으로 등장하는 0을 의미한다. 이때만 out의 값이 1이 된다. | firstZero | firstOne |

lab시간에 사용한 pdf에서 설명한 것과 같은 형태로 구현하였다. 인풋은 clk와 in두개이고 아웃풋은 out 하나로 간단하다. Design에서 설명한 것과 같이 4개의 state가 존재해야 하므로 2비트의 state와 nextState를 선언하였고 state를 ones로 초기화하였다. 이후에는 in이 들어올 때nextState에 다음 상태를 저장하였다. In에 값에 따라서 위의 표들의 정보들을 사용하여 nextState를 결정하였다. 바로 nextSate로 바꾸지 않고 clock의 posedge가 될 때 현재 상태를 다음 상태로 업데이트 해주었다. 그리고 업데이트 해 주면서 out 레지스터에 0 또는 1의 값을 출력하였다. State가 endZero일 때만 1을 반환하고 나머지 경우에는 0을 반환한다.

1. Register File

|  |  |  |
| --- | --- | --- |
| Write | 1bit | 1일 때 Addr3에 Data3를 쓴다. 이 동작은 clock의 다음 posedege에서 일어난다. |
| Clk | 1bit | 주기적으로 0, 1값을 가진다. |
| Reset\_n | 1bit | 1일 때 모든 레지스터의 값을 0으로 바꾼다. 0일때만 read랑 write가 동작하게 된다. |
| Addrx | 2bit | 레지스터의 주소를 저장한다. 2bit 이므로 총 4개의 레지스터를 갖는다. |
| Datax | 16bit | 레지스터에 저장된 값이다. 16bit까지 저장 가능하다. |

RF모듈의 인풋, 아웃풋 값은 위와 같다. Read같은 경우에는 asynchronous 하므로 위 값들 중 하나라도 바뀌면 동작하도록 always @(\*) 블록 내부에 구현하였다. Always 블록 내부에서는 reset\_n의 값에 따라서 0으로 변경시키거나 원하는 주소에서 데이터를 읽어왔다. 데이터를 읽어올 때는 case를 사용하였다.

Write은 read와는 다르게 synchronous해야 하므로 clk의 posedge에서만 동작하도록 always @(posedge clk) 블록을 사용하였다. 이 블록 내부에서 write의 값이 1일 경우에만 addr3에 data3를 쓰게 했다.

1. Discussion

아무래도 verilog에 익숙해 져서인지 저번 lab에 비해서는 조금 수월하게 구현할 수 있었습니다. 특히 fsm이나 moore, mealy machine에 대해서 들어본 적이 없었는데 lab 시간과 pdf에 예시를 통해 자세히 설명이 되어 있어서 좋았습니다. 구현하면서 case문을 많이 사용했는데 이제 보니 코드가 불필요하게 길어지고 가독성이 떨어지는데 이것을 그냥 이중 배열로 구현했으면 더 쉬웠을 거라는 생각이 이제 들었네요.

1. Conclusion

목표였던 010 Detector와 RF파일을 모두 구현하였다. 별 문제는 없었다.