Computer Organization (001)

Lab03-RTL Design

전기정보공학부

2017-17497 김현규

1. Introduction

이번 프로젝트에서는 간단한 자판기를 구현하면서 RTL design을 리뷰한다. 자판기는 총 3가지 종류의 동전을 받고 4가지 종류의 물건을 판매하며, 모든 동전과 물건은 무한히 많다고 가정한다.

1. Design

Vending machine의 input은 총 5개, output은 4개로 구현하였으며 모든 동전과 물건이 무한히 많은 상태이므로 state는 자판기에 들어있는 돈으로만 구성된다. 일반적인 RTL design에 맞도록 combinational circuit으로 다음 state와 output을 구한 후 clock에 맞추어서 현재 상태를 변화시키도록 구현하였다. 각 인풋과 아웃풋 값이 의미하는 바는 아래와 같다.

|  |  |  |
| --- | --- | --- |
| clk | 1bit | 클럭, posedge일 때 다음 state로 넘어가는데 사용된다. |
| reset\_n | 1bit | 1일 때 자판기의 상태를 초기화한다. |
| i\_input\_coin | 3bit | i번째 동전들어왔으면 I\_input\_coin[i]가 1이다. |
| i\_select\_num | 4bit | i번째 물건을 구매시도하면 i\_select\_num[i]가 1이다. |
| i\_trigger\_return | 1bit | 반환버튼을 눌렀을 경우 1이다. |

|  |  |  |
| --- | --- | --- |
| o\_available\_item | 4bit | 현재 구매가능한 물건들을 나타낸다. |
| o\_output\_item | 4bit | 구매한 물건을 나타낸다. |
| o\_return\_coin | 10bit | 돈을 반환할 때 나오는 동전들의 수를 의미한다. |
| o\_current\_total | 32bit | 현재 자판기 안에 들어있는 돈을 뜻한다. |

Combinational circuit은 돈이 들어오거나, 물건을 구매시도 하거나, 반환을 시도할 때 동작하도록 디자인하였으며, 순서에 의존하므로 blocking assignment 로 output과 다음 상태를 구하도록 하였다. 특히 available\_item을 구할 경우에는 동전을 넣은 후, 물건을 구매한 이후에 동작해야 함을 유의했다. Sequential circuit에서는 클럭의 posedge 에서만 동작하며, reset\_n의 값에 따라서 current\_total의 값을 0으로 하거나 next\_total로 변경하도록 디자인하였다.

1. Implementation

Combinational circuit은 크게 다음 state를 결정하는 부분과 output을 결정하는 부분으로 나눌 수 있는데 두 부분을 별도로 분리하지 않고 하나로 묶어서 구현하였다. 인풋이 바뀔 때 실행되야 하도록 always 문 내부에 3개의 인풋을 지정하였다. 클럭이 바뀔 때도 실행되게 하면 동전이나 물건을 중복으로 구매하지 않도록 처리해주어야 하므로 구현의 편의를 위해 인풋이 바뀔 때만 always 문 내부가 실행되도록 하였다.

Always문 내부에서는 먼저 output값들을 0으로 초기화 해주고 다음 상태를 의미하는 next\_total을 current\_total로 초기화해주었다. 이후 동전이 들어온 경우에는 해당하는 동전의 금액만큼 next\_total에 더해주었고(for문을 이용하여 구현하였다) 물건을 구매하는 경우에는 돈보다 물건의 가격이 싼 경우에만 구매할 수 있도록 구현하였다. 반환 버튼을 누른 경우에는 while문을 이용하여 비싼 동전부터 하나씩 검사하면서 잔액을 모두 배출하였다. 만약 동전들로 남은 금액을 모두 배출할 수 없는 경우에도 최대한 큰 금액을 반환하도록 구현하였다. 마지막으로 구매 가능한 물건들의 목록을 반환하는데, 이것은 동전을 넣고 물건을 구매하는 동작이 모두 끝난 이후에 실행되도록 하였다.

Sequential circuit내부는 매우 간단하다. 클럭의 posedge마다 실행이 되도록 하였고 reset\_n이 0일 때는 현재 state를 0으로 변경시키고 아닌 경우에는 현재 상태(current\_total)을 다음 상태(next\_total)로 변경시키도록 구현하였다.

1. Discussion

자판기를 만들면서 RTL에 대해서 더 잘 이해하게 되었습니다. 추가 과제도 도전해보려 했지만, 상태를 여러 개 만들어야 하고 테스트 벤치도 직접 구현해야 해서 만들지 못했습니다. 그래도 많은 것들을 배울 수 있는 lab이었습니다.

1. Conclusion

목표였던 vending machine을 구현하였고 테스트 벤치 또한 통과하였다.