EE312 Computer Architecture

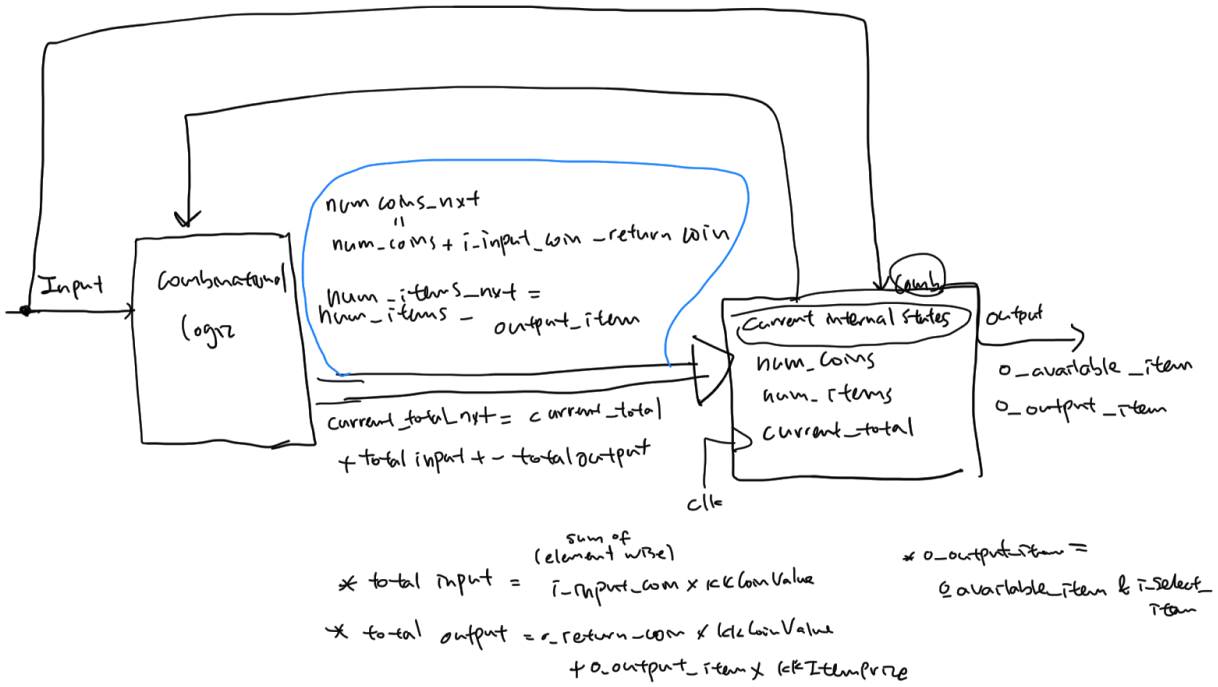
LAB2 Report

20160542 이현지, 20160707 형준하

1. Introduction
    - 이번 랩에서는 verilog를 이용해 vending machine의 모듈을 작성하는 것이 목표였다. 실재 vending machine과 같이 동전을 넣고, 물품을 고르고, 혹은 환불 버튼을 누르고, 그에 따른 output이 생기게 된다. 이를 작성하기 위해서 가장 먼저 FSM에 대한 이해가 필요한데, 우리는 현재 state와 input을 통해 output이 나오는 Mealy Machine을 설계했다.

2. Design

- 하나의 vending machine module을 사용하였다.



- Combinational logic에서 현재 state 와 input(과 output)을 가지고 next state을 계산한다
- 또 다른 combinational logic 으로 input과 current stae 을 이용해 Output(o_avaialble_output, o_output_item)을 계산한다.
- Sequential logic 으로 clk의 positive edge에서 next state의 값을 current state로 넣어준다.

3. Implementation
    - 가장 먼저 initial begin을 통해서 값들을 초기화 시켰다. Stopwatch의 경우 4'b1111로 초기값을 세팅해두었다.

- trigger input을 일시적으로 저장시키는 reg_trigger와 stopwatch를 저장하는 reg_stopwatch 레지스터를 만들었다. 이를 만든 이유는, 인풋 트리거 값이 한 사이클 동안만 켜져있어도 동전이 모두 나올때까지 트리거값을 유지시키기 위해서이다. 스톱워치의 경우 인풋이 들어오지 않으면 1씩 작아지고, 무엇인가 들어오면 다시 4b'1111로 돌아가도록 설계했고, 0이 되면 동전을 돌려주도록 했다. 이때 트리거와 마찬가지로 동전이 모두 나올때까지 reg_stopwatch를 0으로 유지시키도록 했다.
- 그 다음, always begin 안에 combinational circuit을 구현하였다. 첫째 always begin안에는 next state값들을 계산하는 역할을 한다. Num_coins_nxt의 경우 현재 num_coins 에서 input, output으로 들어오고/나가는 코인을 더하고/빼서 계산하였고, Num_item_next는 현재 num_item에서 output으로 나간 아이템을 뺌으로서 구현하였다. Current_total_nxt 는 current_total에서 들어온 돈과 (input_coin의 개수 * 각 coin 의 value) o_return_coin에 명시된 나가는 돈, 그리고 user가 뽑은 아이템의 가격을 빼서 구했다.
- 두번째 always begin안에는 output 값들을 계산하였다. (available item, output item)을 계산했다. 각 아이템에 대하여 현재 current_total이 그 아이템 가격보다 큰지, 그리고 아이템 재고가 있는지(num_items) 확인을 하고 조건이 충족되면 각각의 bit를 1로 켜서 구현하였다. 그리고 o_output_item의 경우 o_available_item과 i_select_item을 and 시켜 구했다.
- Sequential 부분은 positive clock edge 혹은 reset의 neg edge에 실행되도록 해두었다. reset signal이 진행될때는, initial begin에서와 같이 모든 값을 초기로 세팅하도록 해두었다.
- 그 후, return을 구현하였다. 이 부분 또한 클락 싱크에 맞게 posedge clk 안에서 진행되도록 하였다. 이는 reg_trigger_return =1 이거나, reg_stopwatch=0 일때 실행되도록 하였다. 이것이 실행되면, current total 이 0이 될때까지 동전을 차례대로 뺴내고, 이때마다 o_return_coin 을 세팅한다.

- 일반적인 clock의 positive edge에서는, next state 값들을 current state 값으로 넣어주는 일을 한다.

4. Evaluation

- 13개의 test 를 모두 통과했다.
- 코드에 통일성을 유지하려고 노력했는데(위에 디자인에서 말한 틀을 지키려 노력했다) 헷갈리는 부분이 많기는 했다. 예를 들어 어떤 로직을 combinational logic안에 넣어야 하는지, 아니면 clock edge마다 실행되도록 sequential 안에 넣어야 하는지 등이 많이 헷갈렸다. 그리고 sequential 안에서는 nonblocking assignment를 사용하는 것이 좋다고 되어있던데 우리 조는 그렇게 코드를 작성하지 않았기 때문에 어떤 단점이 생기는 것인가 의문이 들었다.

5. Discussion
    - 이번 랩에서 Verilog를 자체의 문법이나 나오는 에러들이 낯설어서 많이 헤맸습니다.
    - 위에서 언급한대로 어 어떤 로직을 combinational logic안에 넣어야 하는지, 아니면 clock edge마다 실행되도록 sequential 안에 넣어야 하는지 등이 많이 헷갈렸다. 그리고 sequential 안에서는 nonblocking assignment를 사용해야 하는 것인지에 대한 궁금증도 있었다.
    - 과제가 제대로 설명이 안되어있는 부분이 많아 test bench 파일에서 역으로 유추 해야하는 내용들이 있었는데, pdf파일에 좀더 친절한 설명이 있었으면 좋겠습니다.

6. Conclusion
    - 이번 Lab2을 통해 Block 과 non Block의 차이에 대해 정확히 알 수 있었고, sequential model 에 대해서도 자세히 알 수 있었습니다.

- display등을 이용해 디버깅을 잘 해내었습니다.
- FSM에 대해 이해하고, Mealy Machine을 디자인 했습니다.

딜레이 하루 사용합니다!