# Computer Architecture (CSED 311), Spring 2025
# Lab 1: RTL Design

구현웅 (20210940), 김민서 (20220826)
3/9, 2025

## 1. Introduction

본 랩에서는 컴퓨터구조 설계의 기반이 되는 combinational logic과 sequential logic의 차이점을 이해하고 vending machine을 설계함으로 컴퓨터 구조에 적용하는 방법을 익힌다.

설계 상에는 combinational logic과 sequential logic 부분을 구분하여 표현하였으며 finite state machine으로서 어떤 구조를 가지는지 간단히 나타낼 것이다. 추상적 구조가 실제 베릴로그 코드의 문법으로 어떻게 표현되는지 설명하고 더 나아가 lab 1에서 요구하는 구현 기준 외의 동작에 대해 특이점을 짚으며 본 설계의 발전 방향을 제시하고 마무리한다.

## 2. Design

### 2.1. FSM type: Mealy vs. Moore Machine

Finite state machine은 크게 Mealy machine과 Moore machine으로 설계할 수 있다. Mealy machine은 출력이 상태와 입력에 모두 의존하고, Moore machine은 출력이 상태에만 의존한다. 따라서 Moore machine은 출력이 클럭에 맞추어 동기적으로 변하지만 Mealy machine은 입력 변화에 즉각 반응하여 비동기적으로 출력이 변화한다.

Mealy machine이 출력을 입력과 상태의 조합으로 계산하므로 내부적으로 필요한 상태의 수가 더 적고 모델을 조금 더 유연하게 설계 가능한 장점이 있으므로, 이에 본 랩에서 구현한 vending machine 또한 Mealy machine으로 설계되었다.

### 2.2. Design of Combinational and Sequential Logic

Vending machine 구현은 크게 combinational logic과 sequential logic 부분으로 나눌 수 있다. Sequential logic은 machine 내부의 state를 clock에 맞추어 synchronous하게 업데이트한다. Combinational logic은 외부 입력, 내부 state를 입력받아 외부 출력, 중간값, 그리고 내부 state의 다음 값을 출력한다.

본 랩에서 구현한 vending machine은 크게 세 개의 모듈로 구성된다. 한 모듈은 오직 combinational logic으로만 구성되어, 외부의 입력을 받아서 내부적인 중간값을 조합한다. 이 중간값들은 vending machine의 최종 출력의 후보가 되는 값이다. 다른 모듈은 이 중간값과 외부 입력을 받아서 machine 내부 상태를 갱신하고 다른 보조적인 값을 계산한다. 이 모듈은 내부 상태를 계산하기 위한 sequential logic과 보조 값을 계산하기 위한 combinational logic을 동시에 갖는다. 마지막 모듈은 앞전에 구한 외부 출력의 중간값으로부터 최종 출력을 combinational logic으로 조합해낸다. 최종 출력의 조합을 위해 필요한 내부 상태를 관리하기 위해 sequential logic 또한 갖는다.

## 3. Implementation

본 vending machine은 calculate_intermediate_output, calculate_return_and_wait_time, regularize_output 세 개의 내부 모듈을 포함하고 있다. calculate_intermediate_output에서는 combinational logic으로 입력된 돈, 구매 가능한 물품, 입력된 물품이 실제로 구매가능 한지 등을 계산하고 중간값으로서 다른 모듈에 넘겨주는 역할이다.
calculate_return_and_wait_time은 기기가 입력를 기다리는 동작과 코인의 반환을 구현하기 위한 모듈이다. regularize_output의 다른 모듈에서 만든 중간값을 실제 vending machine의 출력으로 가공하는 모듈이다.

### 3.1. calculate_intermediate_output

각각의 모듈의 구현은 다음과 같다. 먼저 calculate_intermediate_output를 살펴본다. 다은은 현재 선택된 물건 입력으로 그 물건의 총 가격이 얼마인지 계산하고, 각각의 아이템의 가격을 현재 머신의 돈과 비교해 구매가능한 아이템을 업데이트 한다.

```
    for (i = 0; i < `kNumItems; i++) begin
      if (i_select_item[i]) begin
        price_total += item_price[i];
      end
      if (current_total >= item_price[i]) begin
        available_item_1[i] = 1;
      end
    end
```

다음은 input_total에 입력된 코인의 값을 더해 총 입력된 코인의 가격을 저장한다. 그 아래에서는 코인이 반납되는 경우의 값을 만드는 과정으로, return_coin이라는 플래그에 그 동전이 반납이 되는지 안 되는지 결정한다.

```
    for (i = `kNumCoins - 1; i >= 0; i--) begin
      if (i_input_coin[i]) begin
        input_total += coin_value[i];
      end
      if (current_total - sum_coin >= coin_value[i]) begin
        return_coin_1[i] = 1;
        sum_coin += coin_value[i];
      end
    end
```

마지막으로 물건이 구매가 가능한지 판단해 머신이 보유한 돈의 데이터를 저장하는 current_total_nxt 값을 만들어준다.

```
    // Check if the items can be purchased
    if (price_total > current_total) begin
      output_item_nxt   = 0;
      current_total_nxt = current_total + input_total;
    end else begin
```

```
      output_item_nxt    = i_select_item;
      current_total_nxt = current_total + input_total - price_total;
    end
```

### 3.2. calculate_return_and_wait_time

본 모듈의 핵심은 물건 구매를 기다릴 때 얼마나 기다렸는지 확인하는 것이다.
기본적으로 wait_time의 다음 값은 현재 wait_time의 감소로 설정한다.

```
    wait_time_nxt  = wait_time - 1;
```

다음 코드의 첫 번째 if문에서는 wait_time과 i_trigger_return에 대해 코인을 반납해야함을 결정하고 이를 return_changes라는 플래그에 설정한다.
그 아래에서는 wait_time를 reset해야하는 경우로 wait_time_nxt의 값을 바꾸어 주고 있다.

```
    // Return changes on time over or manual return
    if (wait_time == 0 || i_trigger_return) begin
      return_changes = 1;
    end

    // Initialize waiting time on coin insert or item purchased
    if (i_input_coin != 0 || output_item_nxt != 0) begin
      wait_time_nxt = 100;
    end
```

위에서 만든 wait_time의 다음값 실제로 clock에 대해 동기화해준다.

```
  always @(posedge clk) begin
    if (!reset_n) begin
      wait_time <= 0;
    end else begin
      wait_time <= wait_time_nxt;
    end
  end
```

### 3.3. regularize_output

마지막으로 다른 모듈에서 만든 값을 통해 실제 vending machine의 출력을 설정한다.
특이한 부분으로 문제에서 요구하는 o_output_item의 동작을 위해 combinational logic으로 출력을 설정하는 부분과 sequential logic으로 출력을 설정하는 부분이 나누어진다.
다음은 o_available_item과 o_return_coin의 값을 설정해주는 코드로 상황에 맞게 이미 만들어 놓은 중간값을 택하는지, 초기화를 하는지 결정한다.

```
  always @(*) begin
    if (return_changes) begin
```

```
      o_available_item = 0;
      o_return_coin = return_coin_1;
    end else begin
      o_available_item = available_item_1;
      o_return_coin = 0;
    end
  end
```

마지막으로 o_output_item의 경우엔느 clock에 따라 동기화를 해준다. 추가로 current_total의 값도 여기에서 함께 동기화를 한다. o_ouput_item의 경우에는 아이템이 출력되었을 때 벤딩 머신이 소유의 돈의 값도 변하기 때문에 다음과 같이 동기적으로 작동하도록 구현하였다.

```
  always @(posedge clk) begin
    if (!reset_n) begin
      o_output_item <= 0;
      current_total <= 0;
    end else begin
      o_output_item <= return_changes ? 0 : output_item_nxt;
      current_total <= current_total_nxt;
    end
  end
```

## 4. Discussion

Vending machine은 reset 입력이 들어오면 모든 내부 상태의 값을 초기화한다. reset의 구현은 동기적으로도, 비동기적으로도 구현할 수 있다. 이번 구현에서는 클럭이 들어올 때마다 reset 값을 확인하도록 동기적으로 작동한다. 이를 `always @(negedge reset_n)`과 같이 비동기적으로 구현했을 때와 차이점 및 장단점의 비교를 해볼 만 하다.

어떤 입력이 클럭 도달 지점을 경계로 값이 바뀌는 경우, sequential logic에서는 해당 시점 직전의 값을 가져가는지, 또는 해당 시점 직후의 값을 가져가는지 아직 불확실하여 vending machine 구현 과정에서 시행착오가 있었다. 이 부분을 다시 한번 살펴볼 필요가 있다.

Vending machine을 구성하는 각 모듈 또한 하나의 finite state machine로 본다면, 이들 각각이 하나의 Moore machine 또는 Mealy machine일 것이다. 이번 구현에서 vending machine 전체로 보면 Mealy machine이지만, 각 모듈 단위로 한정하면 Moore machine인 모듈이 존재하도록 구현할 수도 있었다.

이번 lab에서는 각 동전의 종류별로 최대 한 개만 입력할 수 있었지만, 이를 확장하여 각 동전의 종류별로 개수를 여러 개 지정할 수도 있을 것이다. 또한, 잔돈 반환 시 각 동전 종류 별 최대 1개씩 선택하는 조합으로 모두 반환할 수 있는 경우만 테스트벤치가 작성되어있는데, 이를 일반화하여 위 제한에 관계 없이 각 종류별 1개 이상의 동전을 사용하여 완전히 반환하도록 하여 구현을 발전시켜볼 수도 있을 것이다.

## 5. Conclusion

본 랩에서는 vending machine이라는 finite state machine을 설계하고 구현해보았다. 그 과정에서 combinational logic과 sequential logic을 이해하고 적용하는 방법을 배웠다. 하나의 상태에서 combinational logic으로 다음 상태의 값을 만들고 clock에 따라 synchronous state를 업데이트 해나가는 과정은 설계의  핵심 아이디어이며, 향후 더 복잡한 시스템의 설계, 예를 들어 single cycle, multi cycle CPU 설계의 기반이 될 것이다.

추가로, vending machine의 요구 사항을 만족하는 것에 그치지 않고 사양외의 동작에 대해 논의해봄으로서 동기화 기기 설계에 대한 개인의 이해도와 능력을 높일 수 있었다.