

논리회로 설계 및 실험

- 6주차 : 유한상태기계에 대한 이해 -

담당 교수: 권동현 교수

조교: 송수현

컴퓨터보안연구실 : <http://sites.google.com/csl-pnu>

6주차 목표

목표

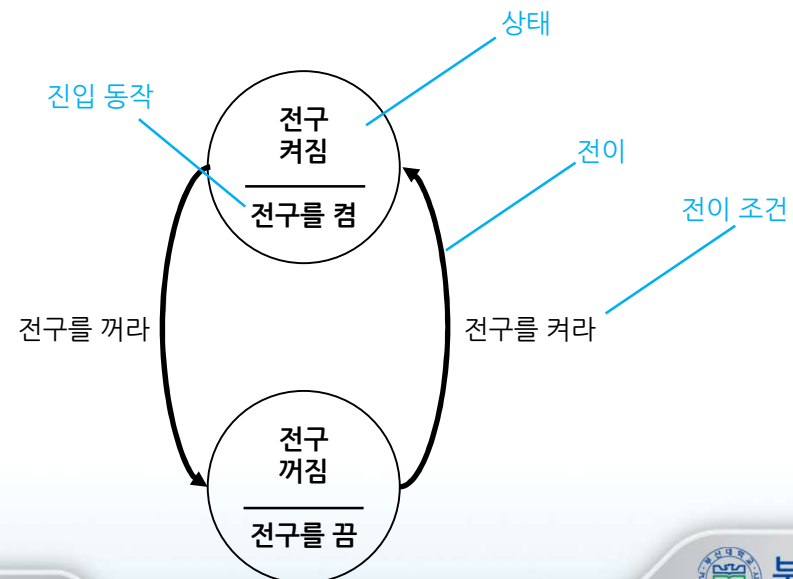
1. 유한상태기계(Finite State Machine, FSM)에 대한 이해
2. FSM을 이용한 컨트롤 유닛(Control Unit, CU) 설계
3. Quartus를 이용한 FSM 설계 학습

유한상태기계

- 프로그램, 논리회로, 정규 표현식 등을 표현하고 설계할 수 있는 수학적 모델
- 유한상태기계의 특성
 - 한번에 하나의 상태만 가짐
 - 어떠한 사건에 의해 현재 상태에서 다른 상태로 변화함
 - 유한상태 변환기 모델로는 무어(Moore) 모델과 밀리(Mealy) 모델이 있음

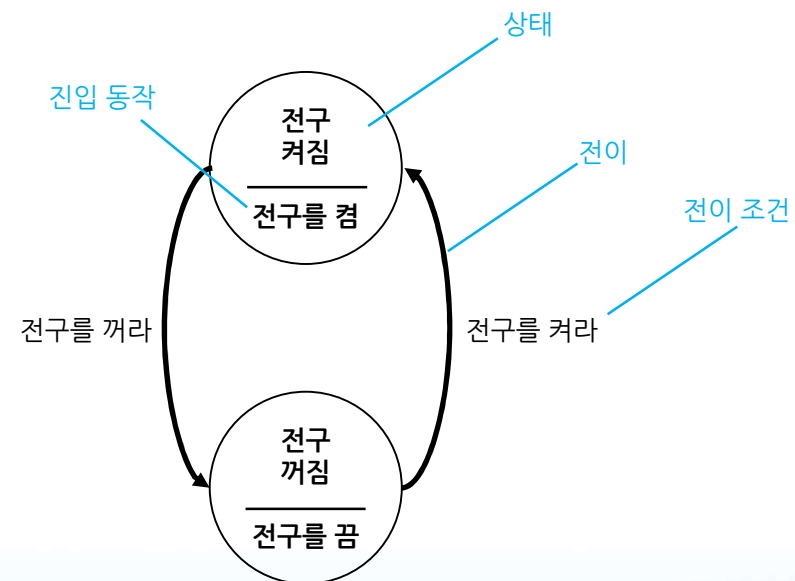
유한상태기계의 예시

- 전구를 켜고 끄는 FSM



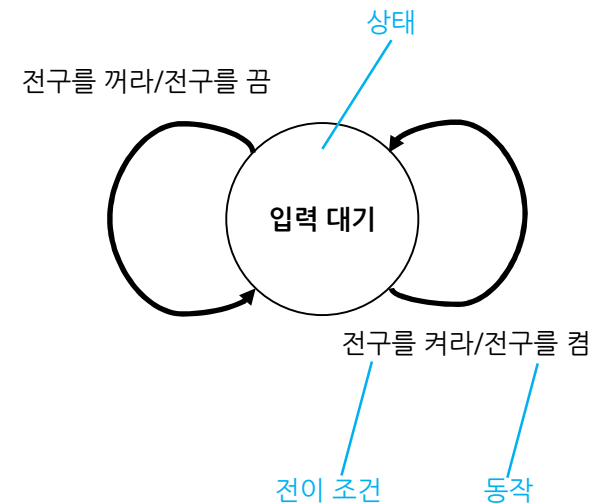
Moore Machine

- 출력이 현재 상태에 따라서 결정됨
- 상태에 진입할 때, 진입 동작을 수행함
- 단순하고 직관적이지만 **상태의 수가 많음**



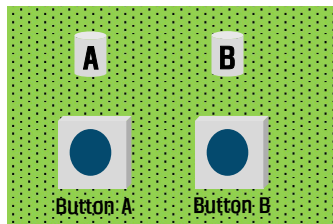
Mealy Machine

- 출력이 현재 상태와 입력에 따라서 결정됨
- 즉, 어떤 입력이 들어올 때 함께 지정된 동작이 동시에 발생함
- 진입 동작은 없음
- 상태의 수를 줄일 수 있으나 **전이 조건 등이 복잡함**

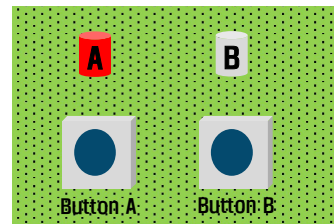


유한상태기계(Finite State Machine, FSM)

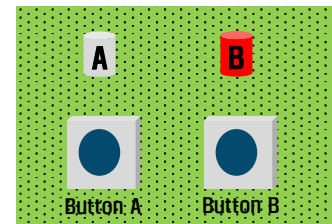
Ex. Button A를 누르면 LED A, Button B를 누르면 LED B가 토글되는 기계



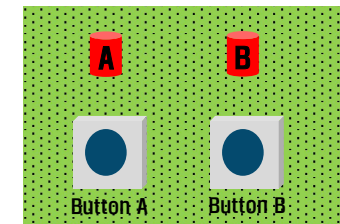
S0



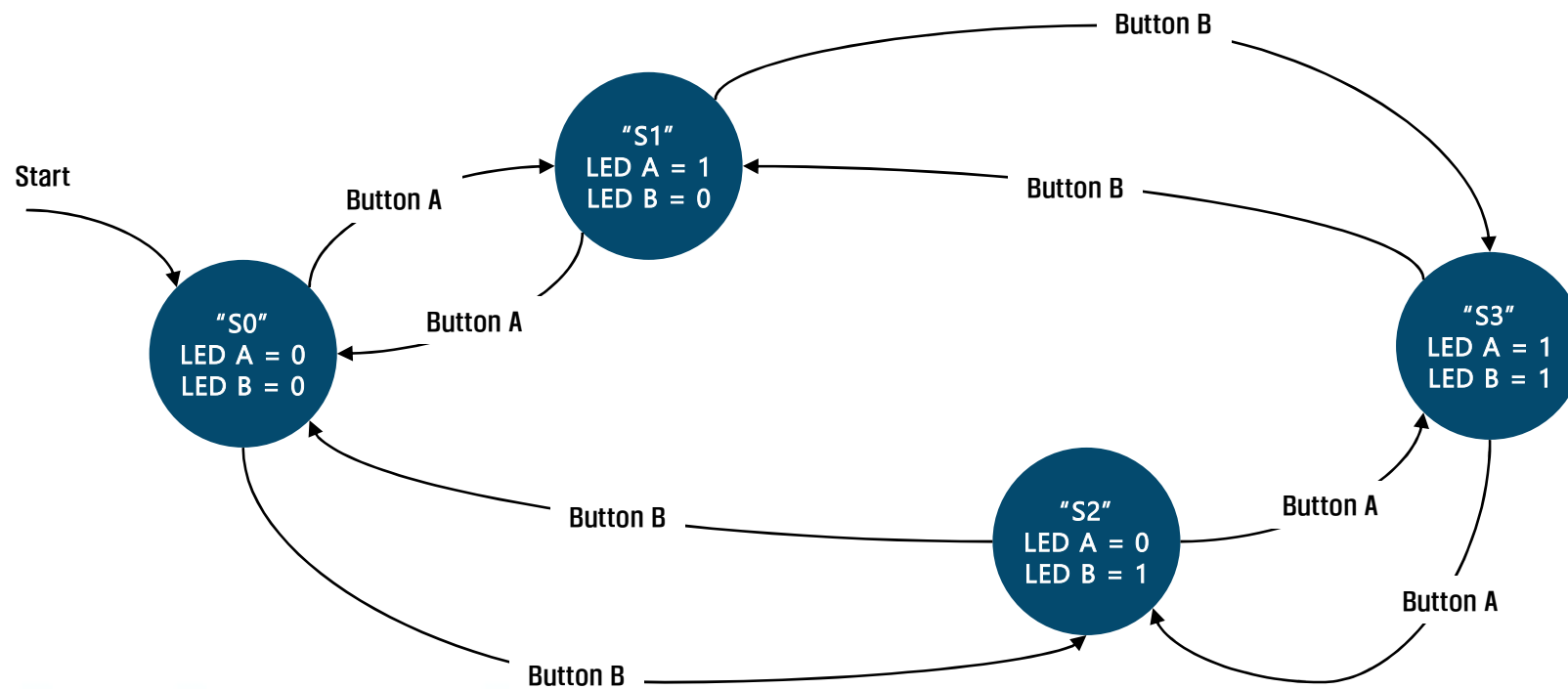
S1



S2

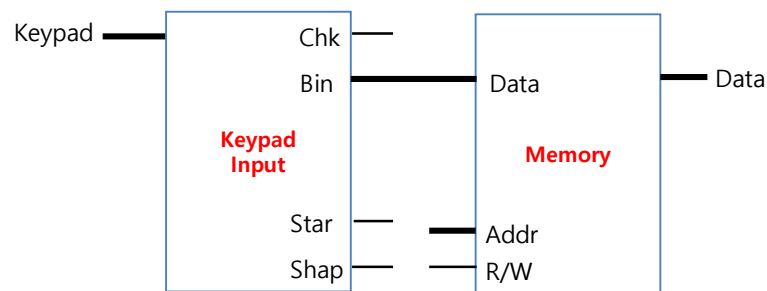


S3

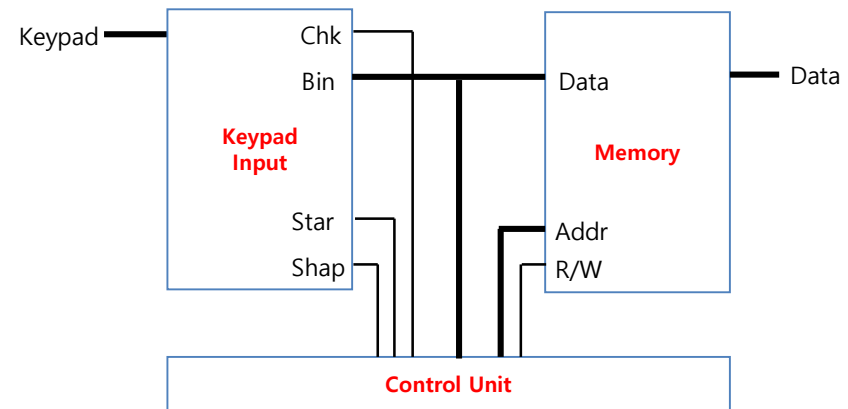


Control Unit

- 데이터 처리(연산, 저장, 쉬프트 등)를 할 수 있도록 제어 신호를 공급함
 - 지난 실습에서 메모리의 Addr, R/W나 레지스터의 Ce 등이 제어 신호임
- 데이터의 흐름을 조절하고 시스템의 정해진 작업을 수행



(1) 입력장치와 저장장치



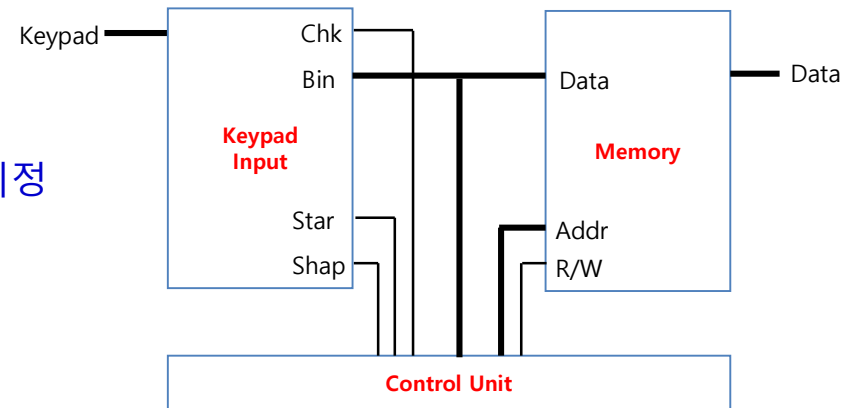
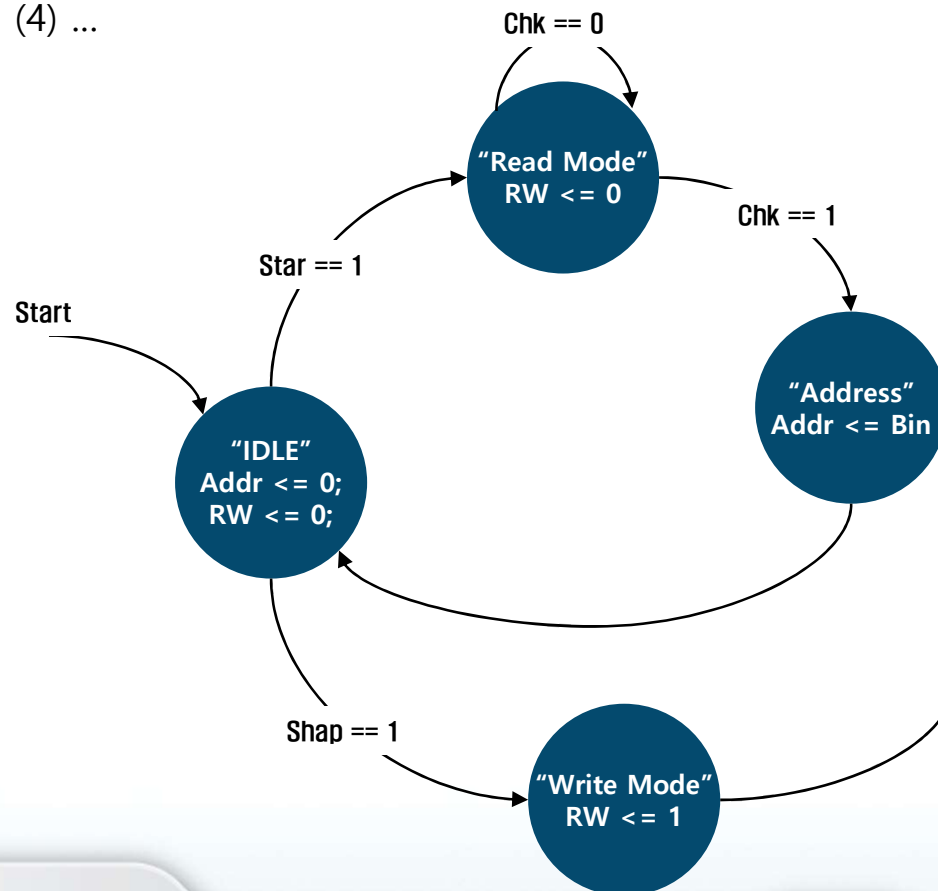
(2) 컨트롤 유닛의 제어를 받는 입력장치와 저장장치

- (1) 의 경우 메모리에 원하는 값을 저장하기 위해서는 Keypad 입력 외에 Addr 값과 R/W 값을 직접 지정해줘야 함
- (2) 의 경우 Keypad 입력만으로 메모리에 읽기, 쓰기, 주소 지정 등이 가능함

Control Unit

Ex. Keypad의 입력 조합으로 메모리를 제어하는 Control Unit의 FSM

- (1) *을 누르면 읽기 모드
- (2) #을 누르면 쓰기 모드
- (3) 읽기 모드에서 숫자를 누르면 값을 읽을 주소 지정
- (4) ...

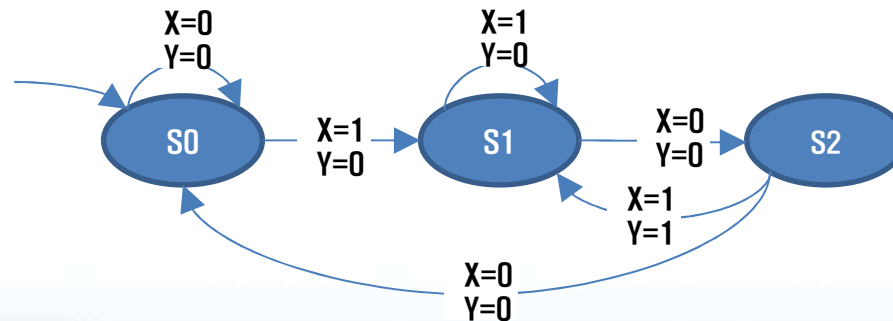
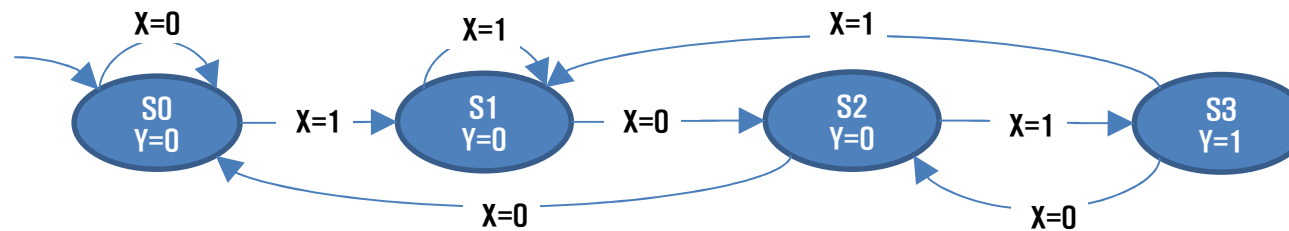


Star : It's occur when pressed keypad's * button
Shap : It's occur when pressed keypad's # button
Chk : It's occur when pressed any number on keypad

FSM을 이용한 Sequential Filter

- 어떤 문자열에서 특정한 패턴이 발견될 때 지정한 동작을 수행하는 필터

Ex. 문자열에서 101 패턴을 찾아내는 Sequential Filter



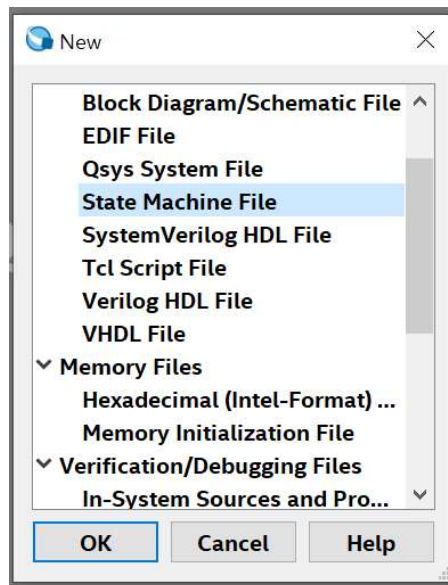
실습

Quartus를 이용한 FSM 설계

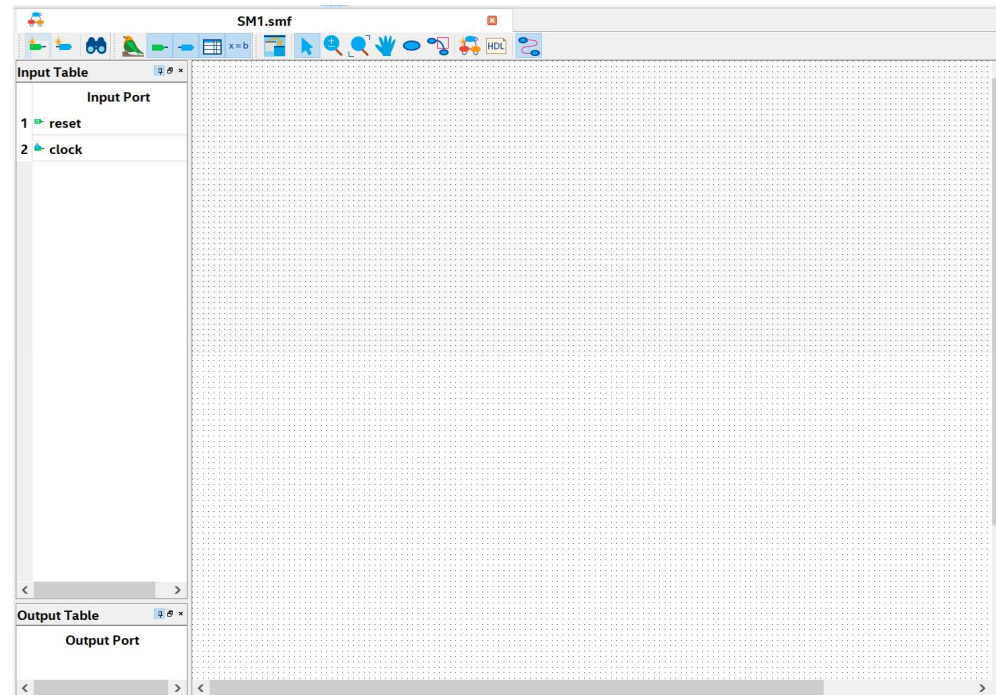
상태도 생성

1. BDF 파일 생성 대신 **State Machine File** 선택
2. SMF 파일 생성 확인

1

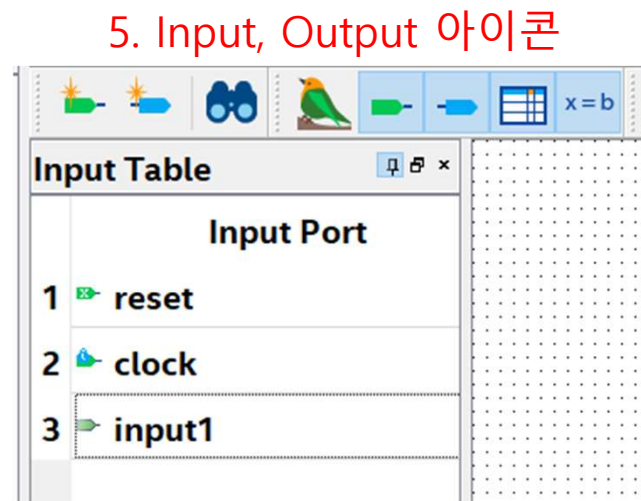
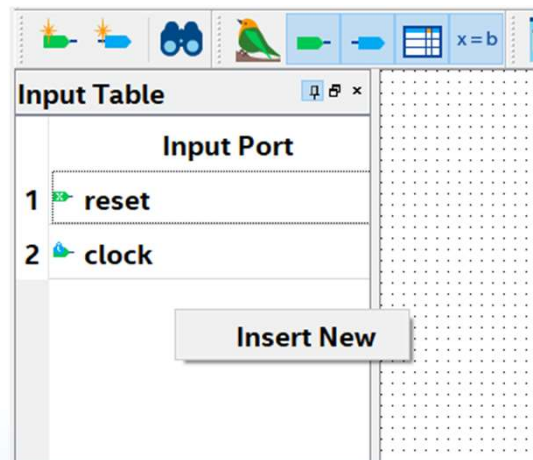


2



상태도 작성 (1/7)

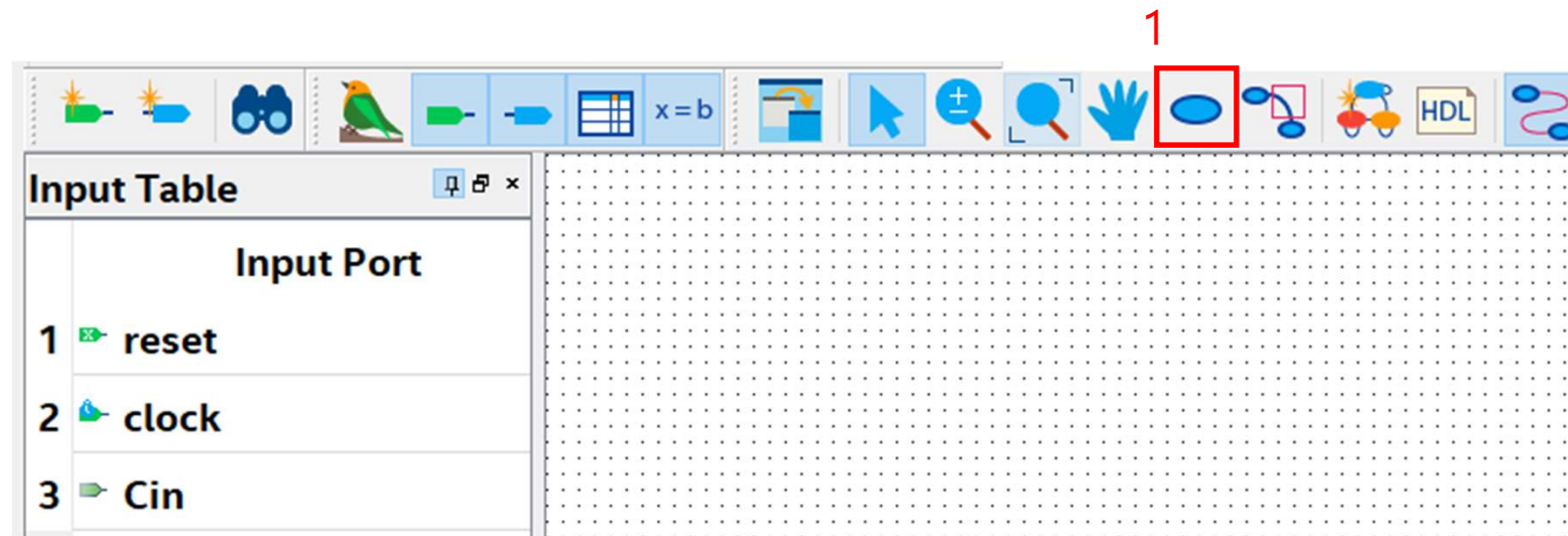
1. Input table에서 오른쪽 마우스 클릭
2. Insert New 클릭하여 새로운 Input 생성
3. 생성된 Input을 더블클릭하여 이름 변경 가능
 - 기본적으로 1비트의 input, input의 이름 뒤 [n:0] 을 추가하면 멀티플 비트
4. Output도 동일한 방식으로 생성
5. 아이콘을 클릭하면 Table에 바로 생성



4비트 인풋 Din 선언 예시

상태도 작성 (2/7)

1. 해당 아이콘을 클릭하여 state 생성 모드로 변경
2. 클릭하여 원하는 개수의 state 생성
→ 기존 BDF에서 게이트 생성과 동일



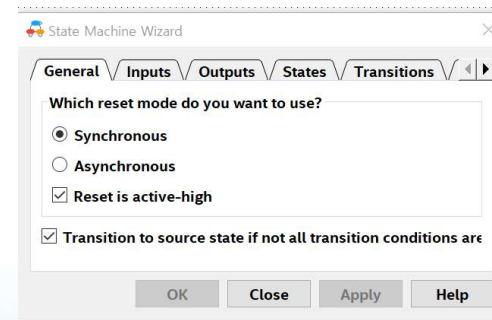
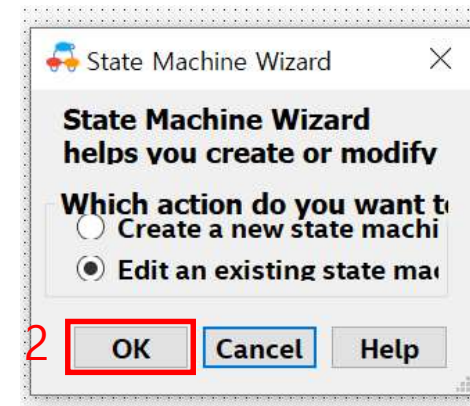
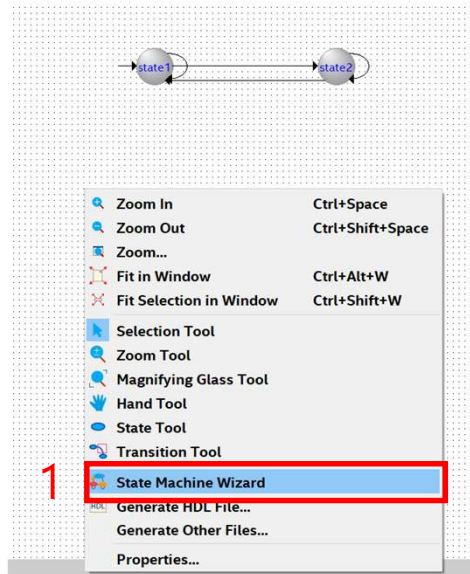
상태도 작성 (3/7)

1. 해당 아이콘을 클릭하여 state Transition 생성 모드로 변경
2. State에서 State로 드래그를 하여 Transition 추가
 - State를 단순 클릭하면 재귀 Transition 추가



상태도 작성 (4/7)

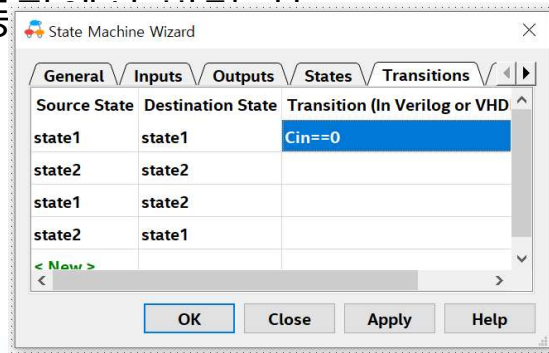
1. 프로젝트창에서 오른쪽 마우스 클릭
2. State Machine Wizard 버튼 클릭 후 Edit an existing state machine 체크 후 OK 버튼 클릭
3. State Machine Wizard 창 확인



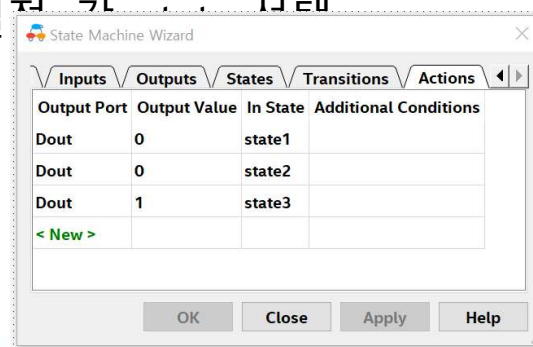
상태도 작성 (5/7)

1. Transition 탭에서 생성한 Transition에 대한 전이 조건 기술
2. Action 탭에서 동작 기술
 - Moore machine은 전이 조건만, Mealy machine은 전이 조건과 동작 둘 다 기술
3. 전이 조건 기술방법
 - C 언어 if 문의 괄호() 안에 들어가는 내용처럼 조건을 기술. Ex) `Cin == 0`
4. 동작 기술 방법

- 동작 기술 방법

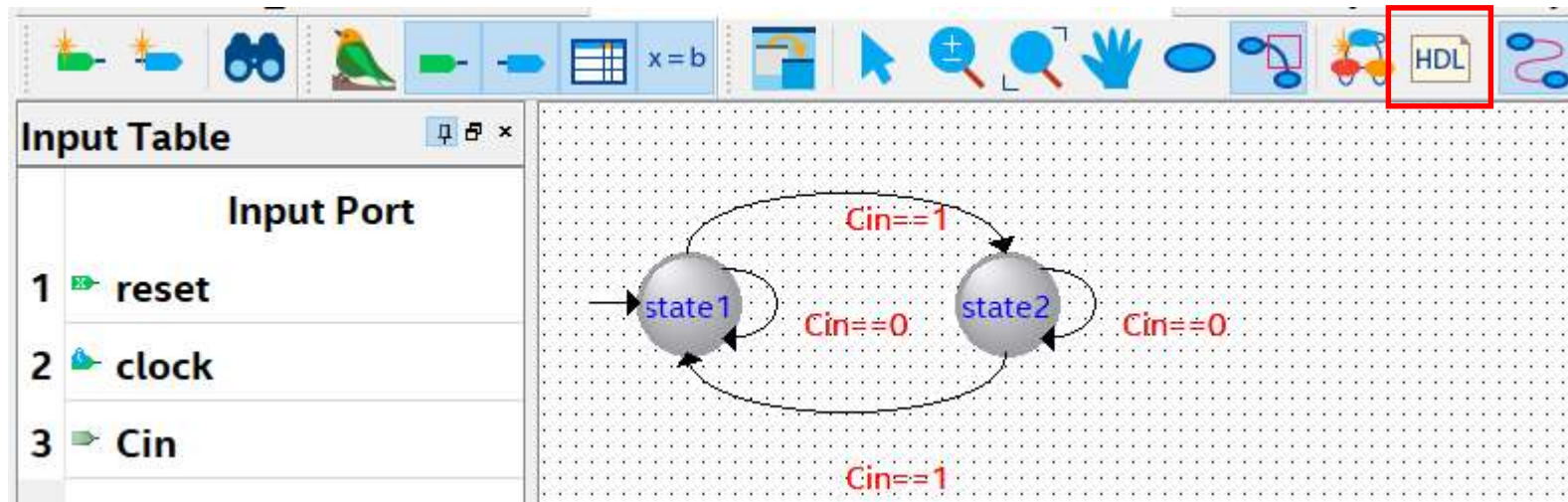


- 동작 기술 방법



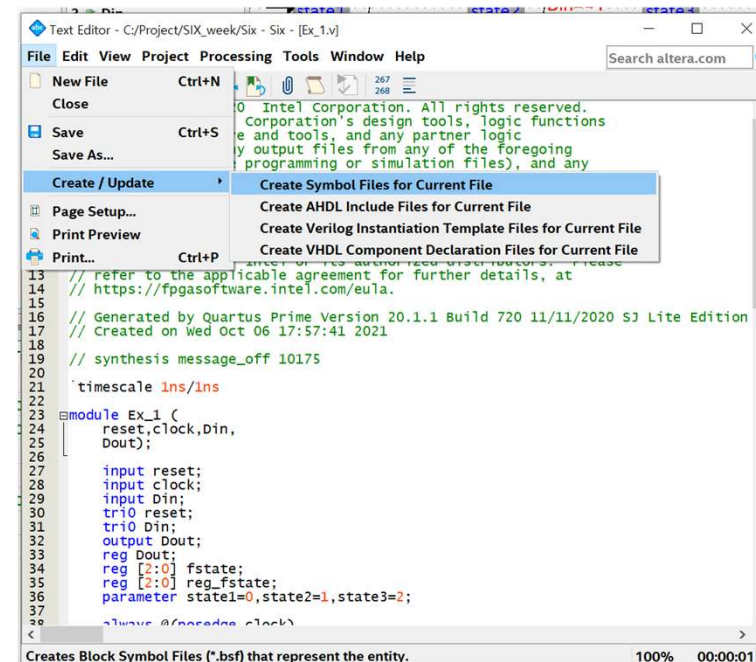
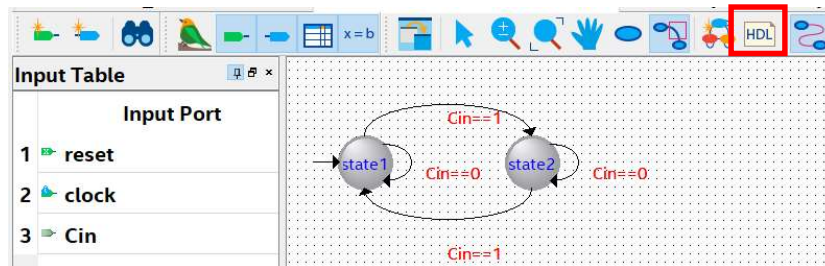
상태도 작성 (6/7)

1. 해당 아이콘 클릭하여 Verilog 파일 생성 (Verilog hdl 선택)
 - Verilog파일과 동일한 이름의 SMF파일 생성됨
2. BDF 파일과 동일한 방식으로 컴파일 후 웨이브폼 설정 및 확인



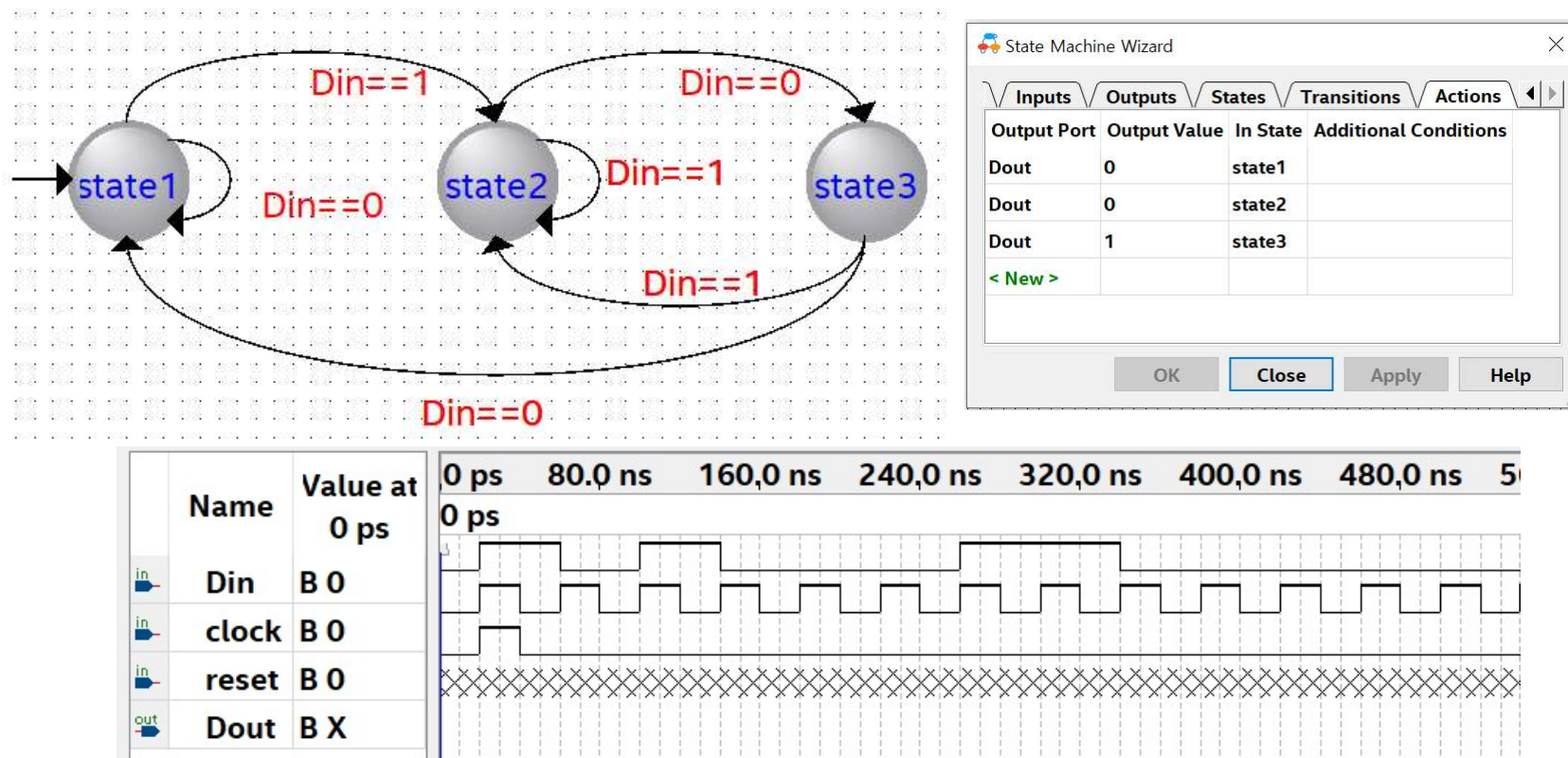
상태도 작성 (7/7)

1. 컴파일 후 해당 아이콘으로 최종 Verilog 파일 오픈 (Verilog hdl 선택)
2. Text Editor 창에서 File – Create symbol 선택
3. BDF로 만든 심볼과 동일하게 사용 가능



실습 1

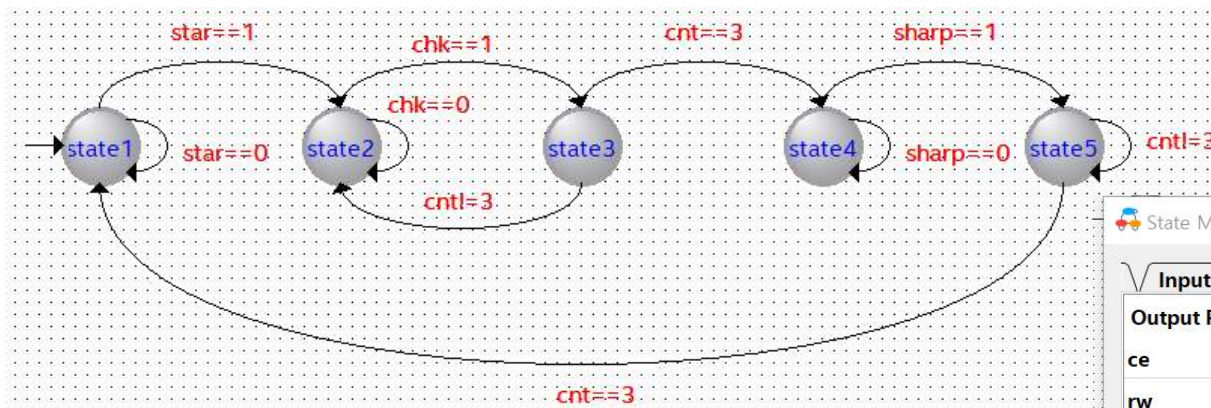
아래 FSM에 대한 입력이 다음과 같다
출력값 Dout을 나타내시오



실습 2 (1/4)

키패드로 SRAM에 값을 저장하고 읽는 Control Unit을 만들려고 한다
부분 모듈의 동작과 전체 모듈의 동작을 읽고 이해한 후, 그와 같은 동작을
하도록 Control Unit의 빈칸에 알맞은 값을 넣으시오

- 1) 처음엔 키패드의 아무 숫자를 눌러도 반응하지 않는다
- 2) *을 한 번 누르고 난 후,
- 3) 숫자를 네 번 누르면 차례대로 SRAM의 0, 1, 2, 3번지에 저장된다
- 4) 이후 #을 누르면 SRAM에서 0, 1, 2, 3번지의 값이 반복해서 읽어진다



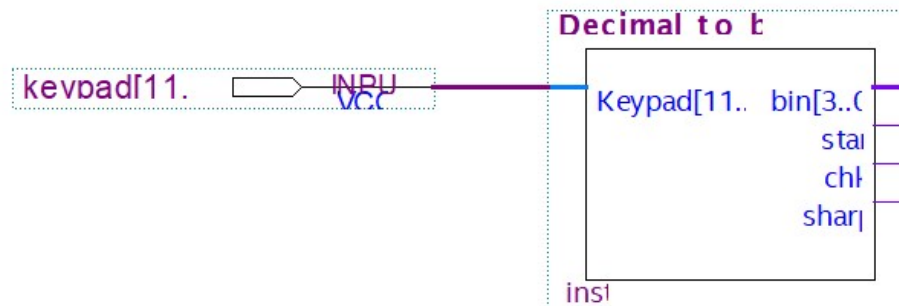
State 3~ 5 까지의 Action을 유추하여 설계

| State Machine Wizard | | | |
|----------------------|--------------|----------|-----------------------|
| Inputs | Outputs | States | Transitions |
| Output Port | Output Value | In State | Additional Conditions |
| ce | 0 | state1 | |
| rw | 0 | state1 | |
| ce | 0 | state2 | |
| rw | 0 | state2 | |

실습 2 (2/4)

부분 모듈 BCD는 일반적인 키패드의 0~9, *, #을 입력으로 가진다
(제공 파일을 심볼화해서 사용) – Decimal_to_binary.bdf

- 1) 숫자를 누르면 해당하는 숫자의 2진수값이 Bin으로 나온다
- 2) 아무 숫자나 누르면 chk에 1이 나온다
- 3) *을 누르면 star에 1이 나온다
- 4) #을 누르면 sharp에 1이 나온다
- 5) *, #을 누를 땐 chk에는 아무 값도 안 나온다
- 6) 입력은 12비트(최상단 11번 비트는 *, 최상단 12번 비트는 #)

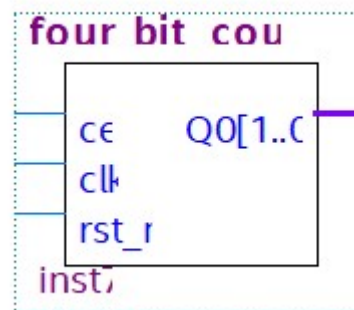


실습 2 (3/4)

부분 모듈 Counter4_ce는 4진 카운터의 일종이다

(제공 파일을 심볼화해서 사용] – four_bit_couter.bdf

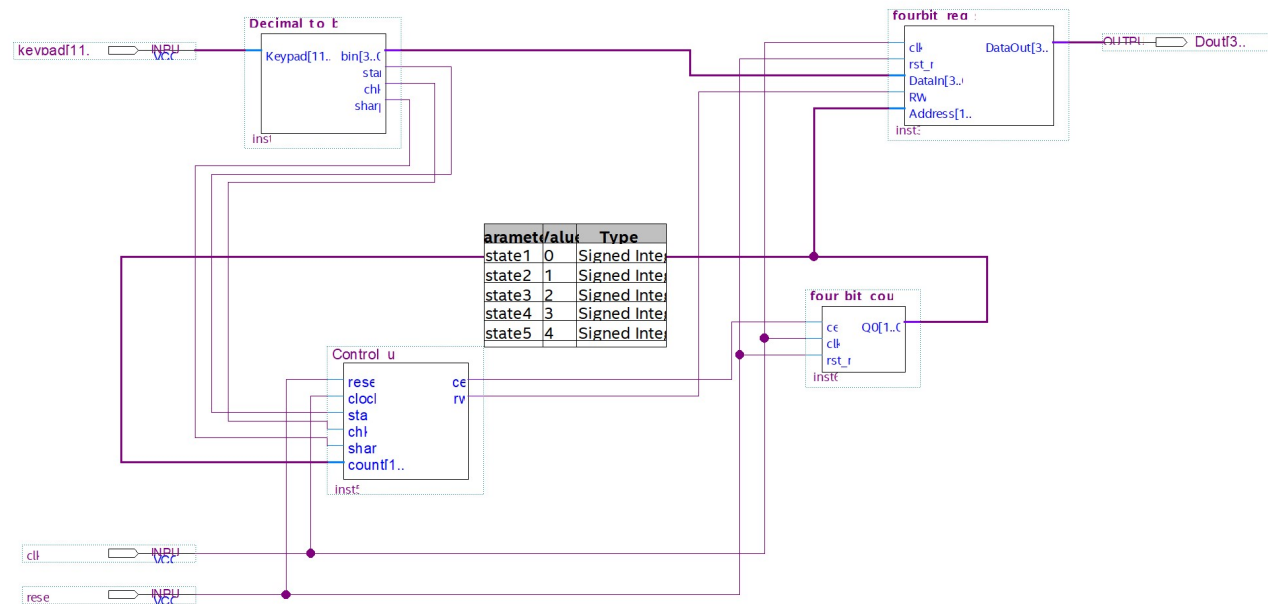
- 1) 처음 출력값은 0 이다
- 2) Ce에 1이 1 클럭 들어가면 출력값이 1 증가한다
- 3) Ce에 1이 들어갈 때 마다 출력값이 증가하여 0, 1, 2, 3, 0, 1... 을 반복한다
- 4) Ce에 0이 들어가면 출력값은 유지된다
- 5) 출력은 2비트(00~ 11) -> SRAM의 주소로 사용



실습 2 (4/4)

전체 모듈은 아래와 같은 데이터패스와 동작을 가진다

- 1) 처음엔 키패드의 아무 숫자를 눌러도 반응하지 않는다
- 2) *을 한 번 누르고 난 후,
- 3) 숫자를 네 번 누르면 차례대로 SRAM의 0, 1, 2, 3번지에 저장된다
- 4) 이후 #을 누르면 SRAM에서 0, 1, 2, 3번지의 값이 반복해서 읽어진다

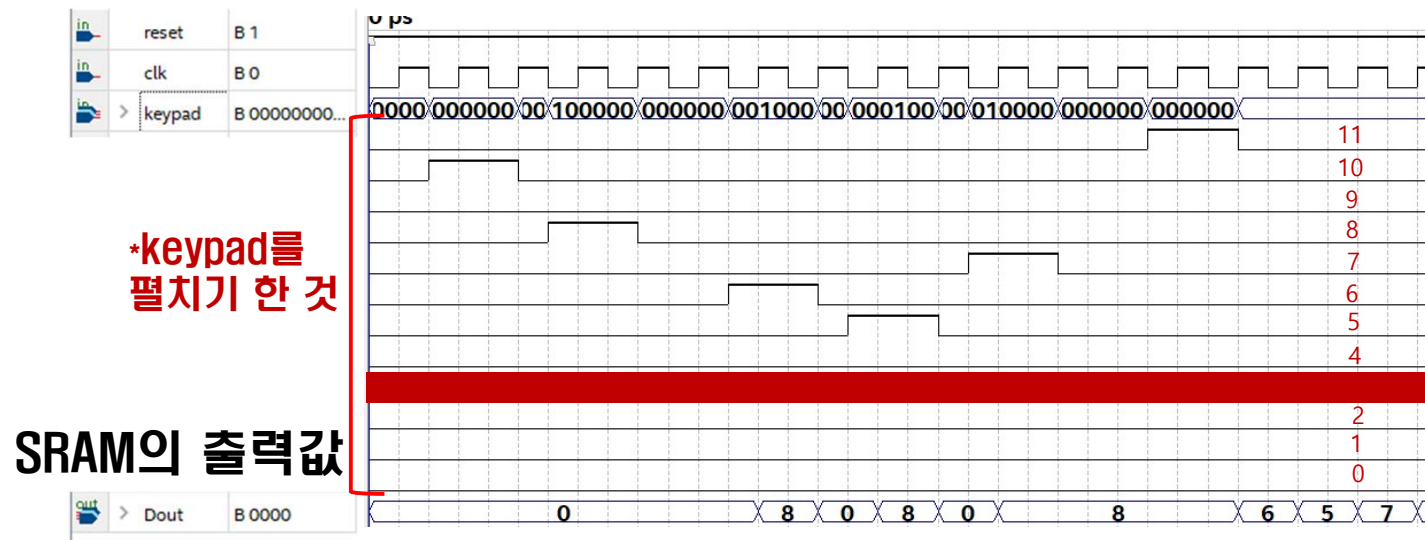


전체 모듈의 데이터패스

실습 2 (시뮬레이션 입력 및 출력)

앞서 전체 모듈의 동작이 가능하도록 Control Unit을 완성하시오

- 1) 처음엔 키패드의 아무 숫자를 눌러도 반응하지 않는다
- 2) *을 한 번 누르고 난 후,
- 3) 숫자를 네 번 누르면 차례대로 SRAM의 0, 1, 2, 3번지에 저장된다
- 4) 이후 #을 누르면 SRAM에서 0, 1, 2, 3번지의 값이 반복해서 읽어진다



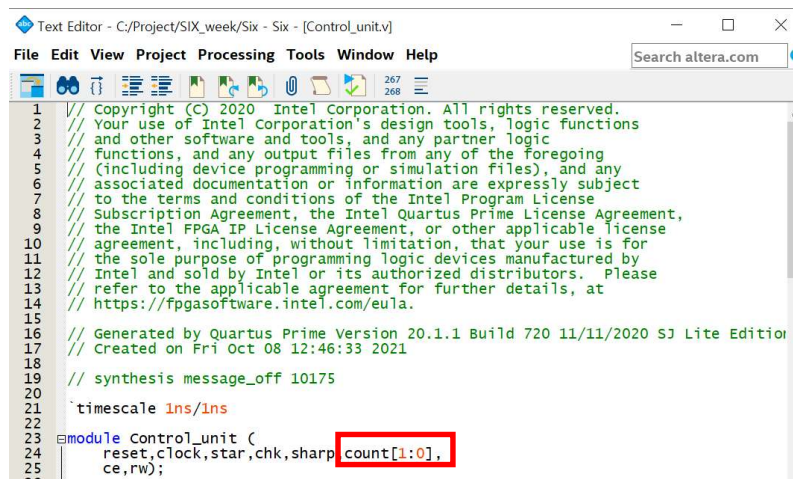
*SRAM은 평소에 00번지의 값을 계속 읽기때문에 의도한 대로 나오지 않음.

참고사항

실습 2의 State machine 같은 경우 멀티비트를 입력으로 받음

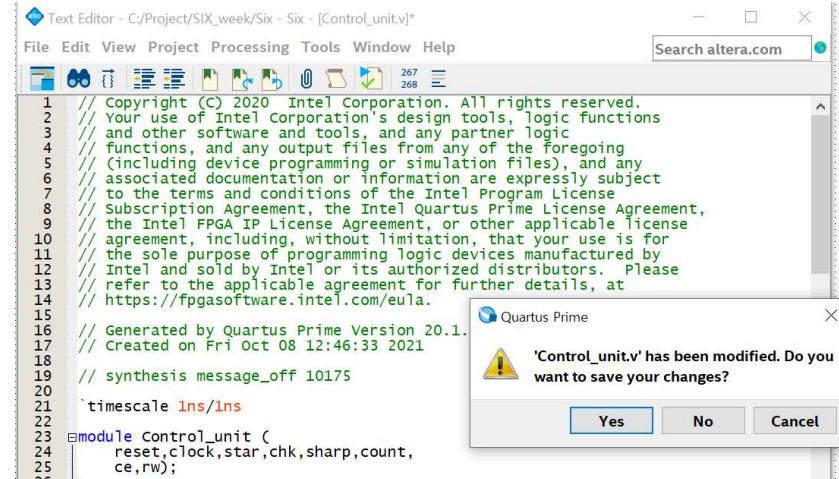
Verilog 파일에서 심볼을 만들 경우 에러가 발생

→ 해결법 : Verilog 파일의 입력 부분의 멀티비트 부분 삭제 후 심볼 생성



```
1 // Copyright (C) 2020 Intel Corporation. All rights reserved.
2 // Your use of Intel Corporation's design tools, logic functions
3 // and other software and tools, and any partner logic
4 // functions, and any output files from any of the foregoing
5 // (including device programming or simulation files), and any
6 // associated documentation or information are expressly subject
7 // to the terms and conditions of the Intel Program License
8 // Subscription Agreement, the Intel Quartus Prime License Agreement,
9 // the Intel FPGA IP License Agreement, or other applicable license
10 // agreement, including, without limitation, that your use is for
11 // the sole purpose of programming logic devices manufactured by
12 // Intel and sold by Intel or its authorized distributors. Please
13 // refer to the applicable agreement for further details, at
14 // https://fpgasoftware.intel.com/eula.
15
16 // Generated by Quartus Prime Version 20.1.1 Build 720 11/11/2020 SJ Lite Edition
17 // Created on Fri Oct 08 12:46:33 2021
18
19 // synthesis message_off 10175
20
21 `timescale 1ns/1ns
22
23 module Control_unit (
24     reset, clock, star, chk, sharp, count[1:0],
25     ce, rw);
26
```

실습 2의 State machine의 멀티비트 입력



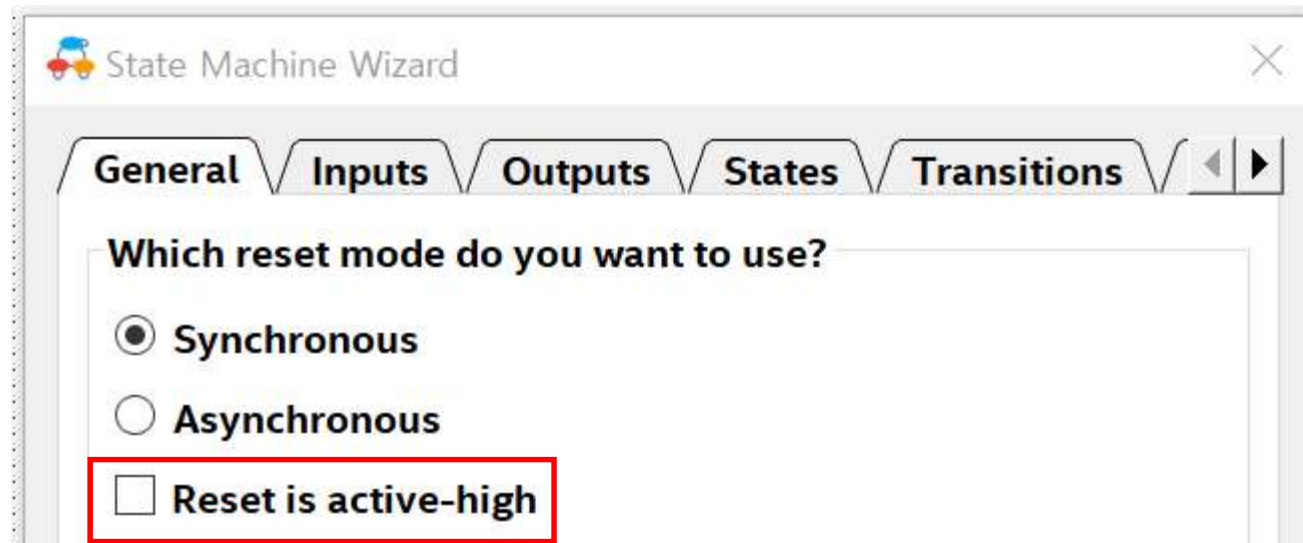
```
1 // Copyright (C) 2020 Intel Corporation. All rights reserved.
2 // Your use of Intel Corporation's design tools, logic functions
3 // and other software and tools, and any partner logic
4 // functions, and any output files from any of the foregoing
5 // (including device programming or simulation files), and any
6 // associated documentation or information are expressly subject
7 // to the terms and conditions of the Intel Program License
8 // Subscription Agreement, the Intel Quartus Prime License Agreement,
9 // the Intel FPGA IP License Agreement, or other applicable license
10 // agreement, including, without limitation, that your use is for
11 // the sole purpose of programming logic devices manufactured by
12 // Intel and sold by Intel or its authorized distributors. Please
13 // refer to the applicable agreement for further details, at
14 // https://fpgasoftware.intel.com/eula.
15
16 // Generated by Quartus Prime Version 20.1.
17 // Created on Fri Oct 08 12:46:33 2021
18
19 // synthesis message_off 10175
20
21 `timescale 1ns/1ns
22
23 module Control_unit (
24     reset, clock, star, chk, sharp, count,
25     ce, rw);
26
```

Quartus Prime
'Control_unit.v' has been modified. Do you want to save your changes?
Yes No Cancel

멀티비트 명시하는 부분 삭제 후 심볼 생성 진행

참고사항

실습 2에서 사용되는 4bit SRAM, Counter 등은 Negative clock을 사용함
Control unit의 클럭도 네거티브로 설정



General 탭에서 해당 항목 체크 해제