컴퓨터 공학 기초 실험 2 – Lab 6

Shifter & Counter

# Shifter

이번 실습에서는 flip-flop과 combinational logic을 이용하여 sequential logic인 shifter와 counter를 설계하여 보도록 한다.

* Description
* Shifter는 register에 저장되어 있는 정보를 단방향이나 양방향으로 이동시킬 수 있는 하드웨어이다.
* 설계할 shifter는 8-bit loadable shifter이고, 다음과 같은 control signal을 입력으로 받는다.

|  |  |
| --- | --- |
| Signal | Description |
| reset\_n | Active low에 동작하는 reset signal로 register 값을 0으로 초기화 |
| op | Shift를 시키기 위한 명령어로써 다음의 명령어를 가진다. - NOP : No operation(현재 register의 값을 그대로 출력) - Load : 입력된 data를 출력  - LSL : Logical shift left를 수행 - LSR : Logical shift right를 수행 - ASR : Arithmetic shift right를 수행  (ASL은 LSL과 동작이 같기 때문에 구현하지 않는다.) |
| shamt | Shift amount로2 bit 값을 가진다. |

Table 1 - Control signal of shifter

* Structural specification

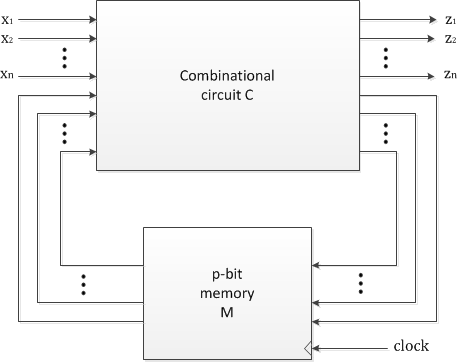


Figure 1 - Example of sequential logic

* 위의 그림은 일반적인 순차회로의 그림을 나타낸다. 위의 그림에서 확인할 수 있듯이 순차회로는 게이트뿐만 아니라 플립플롭까지 포함하고 있어 회로에 저장능력이 있는 회로를 뜻한다. 즉, 순차회로의 출력은 현재의 입력 값과 현재 플립플롭에 저장되어 있는 값에 의해 결정된다.
* 실습에서 구현할 shifter는 위의 그림과 마찬가지로 입력 값과 현재의 register의 값에 의하여 동작하는 combinational circuit과 이를 저장하고 있는 register로 구현한다.
* Design specification
* Module configuration

|  |  |  |
| --- | --- | --- |
| 구분 | 이름 | 설명 |
| Top module | shifter8 | 8-bit loadable shifter의 top module |
| Sub module | LSL8 | 8-bit logical shift left module(top module에서 instance) |
| Sub module | LSR8 | 8-bit logical shift right module(top module에서 instance) |
| Sub module | ASR8 | 8-bit logical shift right module(top module에서 instance) |
| Sub module | mx4 | 1-bit 4-to-1 multiplexer(\_LSL8, \_LSR8, \_ASR8에서 각각 instance) |

Table 2 – Module configuration of shifter

* I/O configuration

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Module 이름 | 구분 | 이름 | 비트 수 | 설명 |
| shifter8 | input | clk | 1-bit | Clock |
| reset\_n | 1-bit | Active low에 동작하는 reset 신호로 값이 인가되면 register의 값을 0으로 초기화 |
| op | 3-bit | Operation으로 NOP, LOAD, LSL, LSR, ASR 중 하나를 선택 |
| shamt | 2-bit | Shift amount로, 2-bit이기 때문에 0~3까지 shift를 수행할 수 있음 |
| d\_in | 8-bit | op에서 LOAD가 인가되었을 때, d\_in을 통하여 들어온 값을 register에 저장하기 위한 입력 |
| output | d\_out | 8-bit | Register의 값을 출력 |
| LSL8 LSR8 ASR8 | input | d\_in | 8-bit | Data in |
| shamt | 2-bit | Shift amount |
| output | d\_out | 8-bit | Data out |
| mx4 | input | d0 | 1-bit | Multiplexer의 1번째 data 입력 |
| d1 | 1-bit | Multiplexer의 2번째 data 입력 |
| d2 | 1-bit | Multiplexer의 3번째 data 입력 |
| d3 | 1-bit | Multiplexer의 4번째 data 입력 |
| s | 1-bit | Multiplexer의 선택 신호 |
| output | y | 1-bit | Multiplexer의 결과 |

Table 3 - I/O configuration of shifter

# Counter

* Description
* Counter는 펄스신호에 따라 어떤 정해진 순서대로 상태의 변이가 진행되는 레지스터를 counter라 한다. Counter는 어떤 사건이 발생할 때마다 펄스신호를 만들어 그 사건의 발생횟수를 세는 등에 사용된다.
* 설계할 counter는 8-bit loadable up/down counter이고, 다음과 같은 control signal을 입력으로 받는다.

|  |  |
| --- | --- |
| Signal | Description |
| reset\_n | Active low에 동작하는 reset signal로 register 값을 0으로 초기화 |
| load | 입력된 data를 register 값으로 load |
| inc | Counter의 증가, 감소를 제어하는 신호로, 1일 경우에는 가산, 0일 경우에는 감산을 수행 |

Table 2 - Control signal of counter

* Control signal 간의 우선 순위는 reset\_n, load, inc이다.
* Structural specification

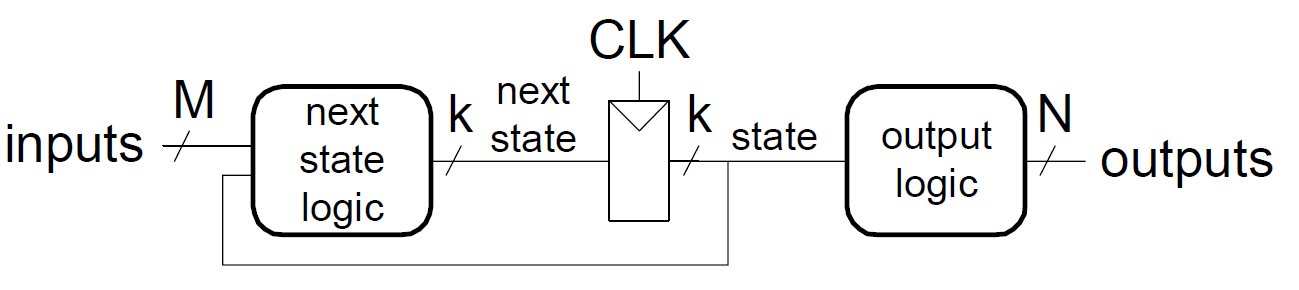


Figure 2 – Moore FSM

* 위의 그림은 Moore FSM 모델을 나타낸다. FSM 모델은 시스템의 동작을 상태(state)와 상태간의 천이(transition)로 표현한다. FSM은 동작방식에 따라 Moore machine과 Mealy machine으로 구분된다. Moore machine은 출력이 단지 현재 상태에 의해서 결정되는 회로이며, Mealy machine은 현재 상태와 입력에 의해 출력이 결정되는 회로이다.
* 해당 실습에서 구현할 counter는 위의 그림처럼 Moore FSM에 기반한 counter를 설계하여 보도록 한다.
* Design specification
* Module configuration

|  |  |  |
| --- | --- | --- |
| 구분 | 이름 | 설명 |
| Top module | cntr8 | 8-bit loadable up/down counter의 top module |
| Sub module | cla8 | 8-bit carry look-ahead adder – top module에서 두 개를 instance하여 inc가 0일 떄와 1일 때의 결과 값을 계산(top module에서 instance) |

Table 5 – Module configuration of counter

* + 1. I/O configuration

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Module 이름 | 구분 | 이름 | 비트 수 | 설명 |
| cntr8 | input | clk | 1-bit | Clock |
| reset\_n | 1-bit | Active low에 동작하는 reset 신호로 값이 인가되면 count 값을 0으로 초기화 |
| load | 1-bit | 입력으로 들어와 있는 data를 count 값에 할당 |
| inc | 1-bit | Count 값을 1일 경우엔 증가시키고, 0일 경우엔 감소시키는 control signal |
| d\_in | 8-bit | Load가 인가되었을 때, 해당 값을 할당하기 위한 data 입력 |
| output | d\_out | 8-bit | Count된 결과 값을 출력 |
| o\_state | 3-bit | 현재 state의 값을 출력(검증용) |
| cla8 | input | a | 8-bit | CLA의 입력 A |
| b | 8-bit | CLA의 입력 B |
| ci | 8-bit | CLA의 입력 carry in |
| output | s | 8-bit | CLA의 출력 S |
| co | 1-bit | CLA의 출력 carry out(연결하지 않는다) |

Table 6 – I/O configuration of counter

# 실습

* Shifter
* 1-bit 4-to-1 multiplexer  
  1-bit 2-to-1 multiplexer를 사용하여 1-bit 4-to-1 multiplexer를 구현한다.
* 8-bit logical shift left  
  Register를 shift amount만큼 왼쪽으로 shift 시킨 후, 빈 공간을 0으로 채운다.  
  다음은 8-bit logical shift left의 symbol과 1-bit 4-to-1 multiplexer를 이용하여 implementation한 그림이다. Module의 이름은 LSL8로 한다.

|  |  |
| --- | --- |
| Symbol | Implementation |
|  |  |

Figure 3 – Symbol & implementation of 8-bit LSL

* 8-bit logical shift right  
  Register를 shift amount만큼 오른쪽으로 shift 시킨 후, 빈 공간을 0으로 채운다. 다음은 8-bit logical shift right의 symbol과 1-bit 4-to-1 multiplexer를 이용하여 implementation한 그림이다. Module의 이름은 LSR8이다.

|  |  |
| --- | --- |
| Symbol | Implementation |
|  |  |

Figure 4 – Symbol & implementation of 8-bit LSR

* 8-bit arithmetic shift right  
  Register를 shift amount만큼 오른쪽으로 shift시킨 후, 빈 공간을 이전의 MSB로 채운다. 다음은 8-bit arithmetic shift right의 symbol과 1-bit 4-to-1 multiplexer를 이용하여 implementation한 그림이다. Module의 이름은 ASR8로 한다.

|  |  |
| --- | --- |
| Symbol | Implementation |
|  |  |

Figure 5 – Symbol & implementation of 8-bit ASR

* Top module  
  Top module(cntr8)의 설계는 sequential logic 설계 단계에 맞추어서 구현하도록 한다. 설계 단계는 다음과 같다.
* Drawing the finite state diagram
* Define states
* Define inputs
* Define outputs
* Draw the diagram
* Encoding states
* Coding the module header
* Coding state registers(flip-flops) – sequential circuits
* Coding combinational circuits
* Counter
* 8-bit CLA  
  이전 실습에서 구현하였던 4-bit CLA를 instance하여 8-bit CLA를 구현한다. 해당 module은 top module에서 두 개를 instance하여 각각 값을 증가시키거나 감소하는 데 사용된다.
* Design  
  Sequential logic 설계 단계에 맞추어서 구현하도록 한다. 구현하는 counter는 Moore FSM 방식으로 동작하기 때문에 combinational circuit을 구현할 때, next state logic 부분과 output logic 부분으로 나누어서 구현하여야 함에 주의한다.

# Report & Submission

레포트는 공지사항에 올린 보고서 양식에 맞추어 작성하고, 다음의 사항에 대하여서도 추가적으로 작성한다.

* 제출할 프로젝트: cnt5, shifter8, cntr8
* 원리(배경지식)에 Moore FSM과 Mealy FSM의 장단점에 대하여 조사하고 설명한다.
* 원리(배경지식)에 ring counter에 대하여 조사하고 설명한다.
* 고찰 및 결론에 loadable counter와 ring counter의 장단점 및 응용분야가 무엇일지 고민하여 작성하여 본다.
* 고찰 및 결론에 barrel shifter에 대하여 조사하고, n-bit의 길이를 가지는 register를 n-bit만큼 shift시키고자 할 때 필요한 multiplexer의 bandwidth와 수에 대하여 논의하시오.
* 제출기한 : 1주 (KLAS 과제란 참고) / delay 2 days (20%감점)
* 실습 미수강 인원은 화요일반에 기준하여 디지털 논리 과목 담당 조교에게 보낼 것
* Source code 압축 시 db, incremental\_db, simulation 폴더는 삭제(미수행시 감점)
* 실습자료와 다르게 instance를 요구한 부분에 behavioral하게 구현한 경우 (gate를 instance하지 않을 시) 감점
* 채점기준

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 세부사항 | | 점수 | 최상 | 상 | 중 | 하 | 최하 |
| 소스코드 | Source code가 잘 작성 되었는가?  (Structural design으로 작성되었는가?) | 10 | 10 | 8 | 5 | 3 | 0 |
| 주석을 적절히 달았는가?  (반드시 영어로 주석 작성) | 20 | 20 | 15 | 10 | 5 | 0 |
| 설계검증  (보고서) | 보고서를 성실히 작성하였는가?  (보고서 형식에 맞추어 작성) | 30 | 30 | 20 | 10 | 5 | 0 |
| 합성결과를 설명하였는가? | 10 | 10 | 8 | 5 | 3 | 0 |
| 검증을 제대로 수행하였는가?  (모든 입력 조합, waveform 설명) | 30 | 30 | 20 | 10 | 5 | 0 |
| 총점 | | 100 |  | | | | |