컴퓨터 공학 기초 실험2 보고서

실험제목: FIFO

실험일자: 2020년 10월 19일 (월)

제출일자: 2020년 10월 26일 (월)

학 과: 컴퓨터정보공학부

담당교수: 공진흥 교수님

실습분반: 월요일 0, 1, 2

학 번: 2019202052

성 명: 김 호 성

1. 제목 및 목적

A. 제목

- FIFO

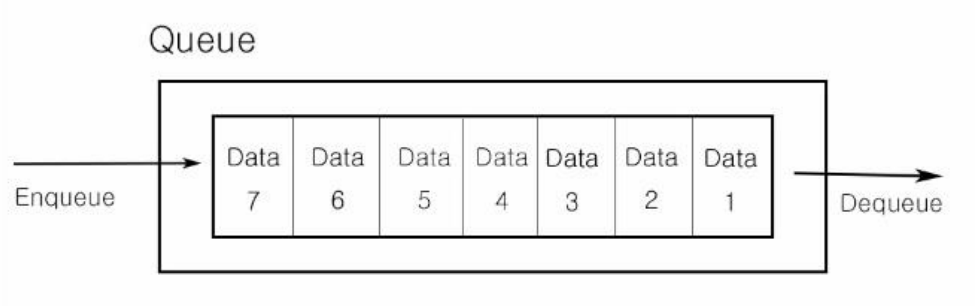
B. 목적

Queue의 원리를 알고 이대로 data를 입력 및 출력하는 프로그램을 구현한다. 이전 실습에서 배운 FSM logic과 register file을 이용하여 원하는 data를 저장 및 출력해본다.

2. 원리(배경지식)

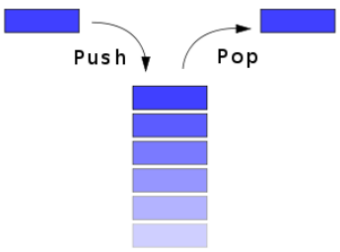
-Queue

자료구조의 한 형태로, 가장 먼저 입력된 data가 가장 먼저 나오는 First In, First Out(FIFO) 방식을 사용하는 구조다. 가게 앞에서 줄을 서서 기다리는 사람들을 연상하면 편하다. 큐의 가장 앞쪽을 head, 가장 뒤쪽을 tail이라 하며, front에서 data가 빠져나가는 것을 dequeue, rear에 data가 들어오는 것을 enqueue라고 한다.



-Stack

자료구조의 한 형태로, 가장 늦게 입력된 data가 가장 먼저 나오는 Last in, First Out(RIFO) 방식을 사용하는 구조다. 쌓아 올린 접시를 닦을 때 가장 위에 있는 접시를 먼저 설거지하는 것을 연상하면 편하다. (가장 위에 있는 접시는 일반적으로 가장 늦게 쌓인 접시이다.) data를 넣을 때는 PUSH, data를 빼낼 때는 POP이라고 하며, 해당 연산을 실행하는 위치를 기억하고 있는 것을 스택 포인터(SP, Stack Pointer) 라고 한다. 스택 포인터는 다음 값이 들어갈 위치를 가리킨다. (일반적으로 SP의 기본 값은 -1)이다

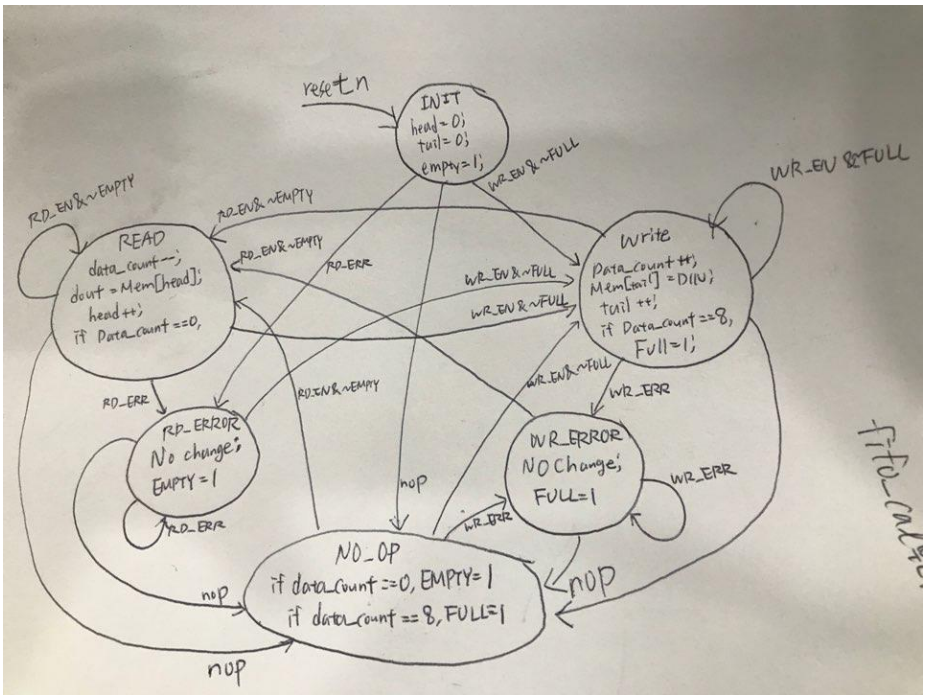


3. 설계 세부사항

먼저 크게 구분하면 FSM 회로, 레지스터 파일과 그 외 mux, 32비트 레지스터를 사용했다.

-fifo_ns

가장 먼저 입력을 받는 모듈로, fifo가 data를 읽거나 쓸 수 있는지 확인해주는 wr_en, rd_en 입력 값을 받고 각 조건에 따른 상태를 나타내는 모듈이다. 이 모듈에서의 출력 값 next_state는 해당 flip-flop에 입력되고, 여기서 출력된 값은 다음 state 지정을 위해 다시 fifo_ns로 입력 받게 된다.



각 state 별 조건은 다음과 같다.

INIT: 아무것도 없는 상태. 그러므로 head = tail = data_count = 0이다.

WRITE: data를 입력받는 상태. 그러므로 tail++ 및 data_count++가 된다.

만약 queue에 data가 가득 찼다면, full 플래그가 참이 된다.

WR_ERROR: data를 더 이상 입력 받을 수 없는 상태. 그러므로 full 플래그가 참이 되고 읽기 동작만 다음 동작이 가능하다.

READ: data를 읽는 상태. 그러므로 head++ 및 data_count—가 된다. 만약 queue에 data가 비었다면 empty 플래그가 참이 된다.

RD_ERROR: data를 더 이상 읽을 수 없는 상태. 그러므로 empty 플래그가 참이 되고 다음 동작으로는 쓰기만 가능하다.

NO_OP: queue가 비어 있거나 가득 차 있는 상태. Empty 플래그가 1이거나 full 플래그가 1이다.

-fifo_cal

Queue 안에서의 data의 이동을 계산하는 모듈이다. 이전 state에서 계산되어 flip-flop을 통해 출력된 head, tail, data_count를 입력 받아 각 상태 별로 증가 및 감소시켜준다. Write의 경우 data가 입력되므로 tail과 data_count를 증가시켜주고 read의 경우 data가 출력되므로 head를 증가시키고 data_count를 감소시킨다. 그리고 이 값들이 다시 해당 flip-flop에 들어가게 되고 위 과정을 반복한다.

-fifo_out

flip-flop에서 나온 state와 data_count 값을 받아 해당 state의 상태를 출력하는 모듈이다. Queue의 상태를 확인해주는 full, empty 플래그를 먼저 출력하고 그 다음 읽을 수 있는지의 여부, 쓸 수 있는 지의 여부를 참 또는 거짓으로 출력한다.

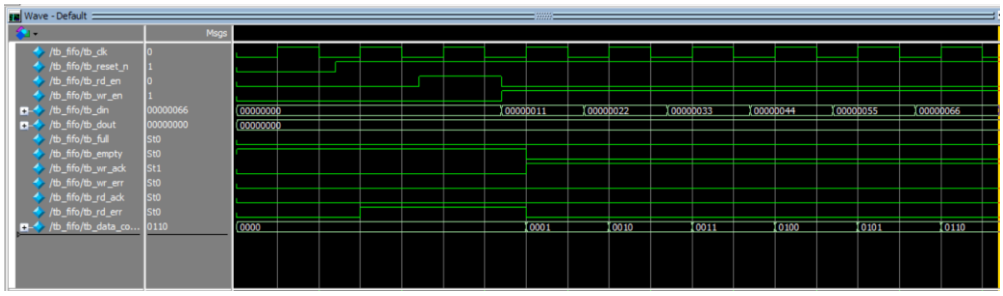
-register file

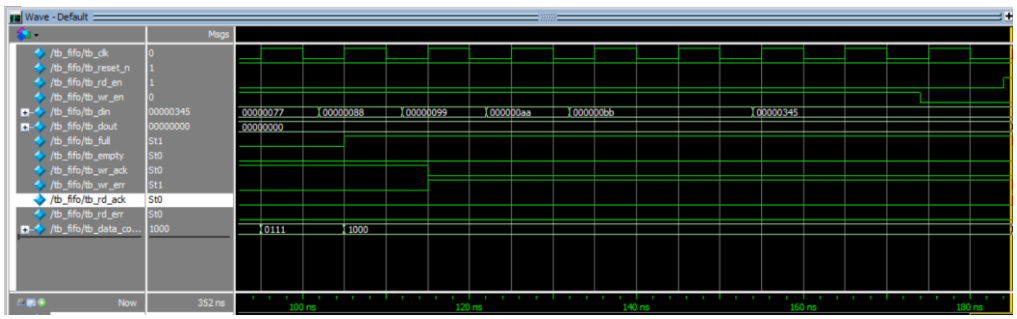
이전에 짠 레지스터 파일과 기능은 동일하다. fifo에서 입력한 메모리를 저장 및 출력하는 모듈이다. flip-flop에서 출력된 head와 tail 값, 그리고 사용자가 입력한 32비트 값을 입력 받아 write operation에 따르면 data를 입력만 하고, read operation에 따르면 그 data를 출력할 수 있게 된다. 추가로 D flip-flop 3, 4, 32bit와 mx232bit를 구현했다.

4. 설계 검증 및 실험 결과

A. 시뮬레이션 결과

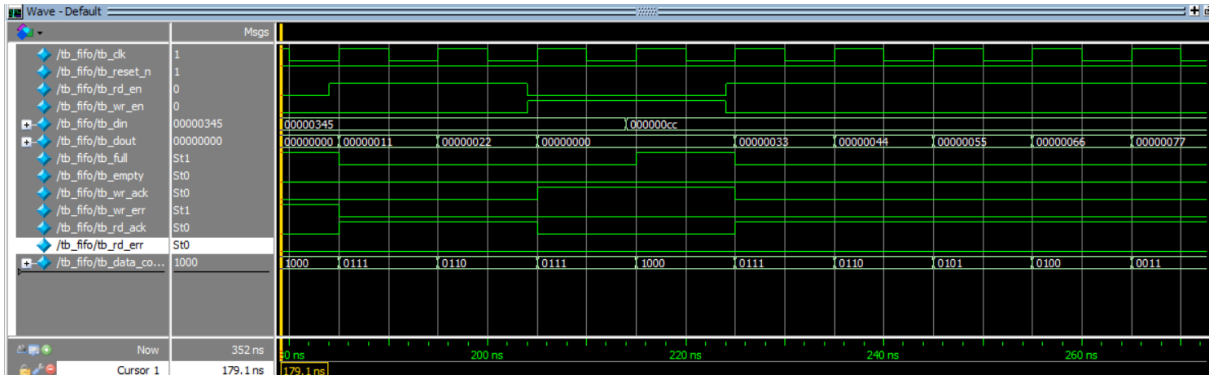
- Write test bench

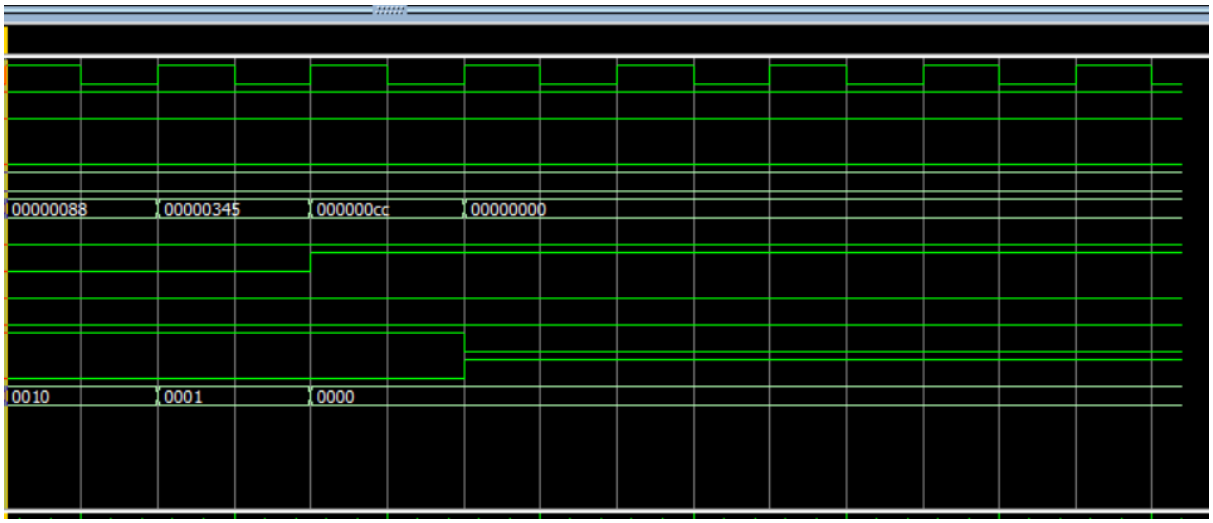




8번째 입력까지는 정상적으로 들어가지만, 그 이후는 full flag가 참이 되어, wr_err가 1이 된다.

-READ test bench(with Write)

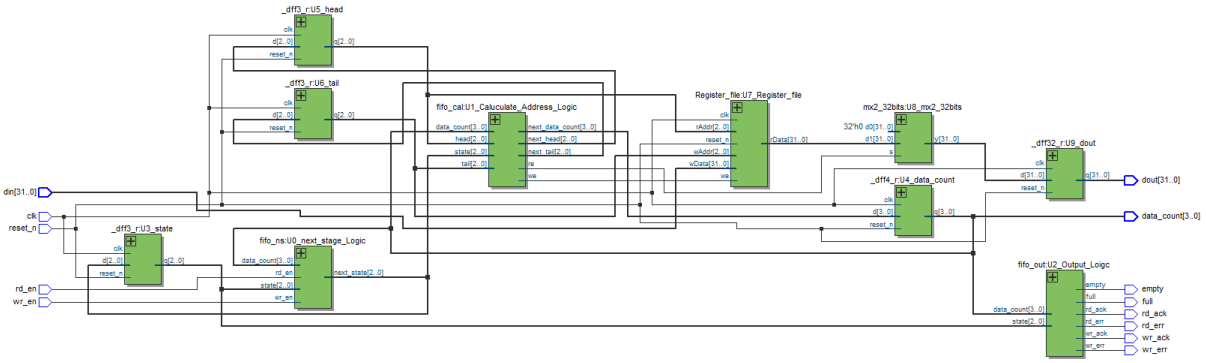




10번 출력이후는 empty flag가 참이 되어 rd_err가 1이 된다. (FSM 정상작동)

또, 출력 과정 중 총 두 번의 입력이 이루어지는데, Register의 경우 다음 값이 입력되기 전까지 이전 값이 유지되는 특징에 의해 1번 째는 32'h0000_0345 2번 째는 32'h0000_00cc 값이 입력된다. 그래서 총 출력은 10번이 이루어지게 된다. (입력도 test bench 전체를 기준으로 10번이 이루어진다.)

B. 합성(synthesis) 결과



입력 값 쪽에서 보면 먼저 clk는 fifo_ns와 fifo_out 사이 4개의 flip-flop과 레지스터 파일에 입력된다. reset_n 또한 각 flip-flop과 레지스터 파일 뒤에 붙어있는 32비트 flip-flop에 입력된다. 이외에도 각 모듈에 맞게 입력이 되고, flip-flop의 출력 값들이 향하는 곳을 보면 각각의 모듈에 맞게 다시 입력이 된다.

5. 고찰 및 결론

A. 고찰

이번 실험은 지금까지 중에 제일 어려웠다. 이전 시간에 배운 Register와 FSM을 활용해서 만드는 과제였으며, 강의 자료실의 FIFO FSM을 기준으로 화살표가 fifo_ns, state안에 적힌 하늘색 글씨가 fifo_cal, 파란색 글씨가 fifo_out이 하는 일이라는 것을 알았다.

B. 결론

INIT: 초기 값 세팅(reset)

NO_OP: data가 empty거나 full인 상태.

READ: fifo에 저장되어 있는 data를 읽음. (+ 삭제)

RD_ERROR: flag상태가 empty로 읽을 수 있는 data가 없음.

WRITE: fifo에 data를 저장.

WR_ERROR: flag상태가 full로 data를 저장할 수 없음.

6. 참고문헌

이준환/디지털논리회로2/광운대학교(컴퓨터정보공학부)/2020

공영호/디지털논리회로2/광운대학교(컴퓨터정보공학부)/2020