**Digital System Design - 1분반**

**Design Snake game using Verilog**

성명: 안병웅, 최지환

학번: 21600399, 21800752



제출일: 2020/12/14

**목차**

문제 정의 및 배경설명 3

I/O 및 수행해야 하는 주요 기능 위주의 설명 3

설계를 위해서 알아야 하는 배경이 되는 이론적 설명 3

설계 7

Block Diagram (전체 Block들의 연결도) 7

Datapath (논리 회로도) 7

Controller 설계 (FSM 그림) 8

시뮬레이션 8

구현 및 실험 결과 9

DISCUSSION 9

공개 코드 사용에 대한 구체적 내역 9

기존의 공개코드를 재사용한 블록 리스트와 기능 9

스스로 전체를 설계한 모듈의 목록과 기능 설명 10

목표대비 달성도 11

성능 평가 12

향후 추가 개발해야 할 내용 및 아쉬운 점 12

부록 12

Bit 파일 다운로드 받아서 사용하는 방법 (제품 매뉴얼에 해당) – 스크린샷이나 사진 포함 12

소스코드 중 중요한 부분의 설명과 해설 13

전체 소스코드 구성 16

1. 본문
   1. 문제 정의 및 배경설명
      * + 입출력 및 수행해야 하는 주요 기능 위주의 설명

본 Design의 입출력은 6개의 입력과 16개의 출력으로 구성되어 총 22개이다. 구체적인 이름과 기능설명은 다음과 같다.

1. CLK100MHZ: FPGA에서 제공하는 100MHZ의 Clock으로 system 전체의 기본적인 clock 신호
2. BTNC: FPGA 보드의 버튼으로, reset 신호를 위하여 사용
3. SW: FPGA 보드의 스위치로, Audio의 화음 조절을 위하여 사용
4. ARP\_ON: FPGA 보드의 스위치로, Audio의 화음을 키고 끄는데 사용
5. PS2\_CLK: PS/2를 이용하여 Keyboard로부터 입력을 받기 위하여 FPGA에서 제공하는 Clock 신호
6. PS2\_DATA: PS/2를 이용하여 Keyboard로부터 받는 데이터의 형식
7. AUD\_PWM: Mono Audio out의 Pulse Wave Modulation을 위해 FPGA에서 제공하는 신호
8. AUD\_SD: Mono Audio out의 기능을 활성화하는데 사용
9. LED: FPGA 위의 LED를 끄고 키는데 사용되며, Audio의 화음을 시각적으로 보이기 위해 사용
10. VGA\_HS, VGA\_VS, Vga: VGA 모니터 출력을 위해 RGB 신호를 화면에 알맞은 주파수에 따라 나타내는데 사용
11. AN, CA, CB, CC, CD, CE, CF, CG, DP: 7-segment display를 통하여 Timer를 나타내는데 사용
    * + - 설계를 위해서 알아야 하는 배경이 되는 이론적 설명
          1. FSM Design

FSM 은 Finite State Machine의 약자이다. FSM Design을 하면 우리가 design한 system은 한번에 하나의 state만 가진다. 또한 특정한 input과 조건에 의해 어떠한 state에서 다른 state로 넘어갈 수 있다.

FSM Design은 전체 모듈을 크게 state를 나타내는 Controller와 data들의 이동과 연산을 나타내는 Datapath로 나누어 구현한다. FSM Design은 내부 구조와 구현이 쉽고, schematic울 그리기 쉽고, 직관적이며 유연성이 있고, 오류 수정이 용이하다는 장점을 가진다.

* + - * 1. PS/2

PS/2는 직렬 통신의 하나로, 마우스나 키보드 등 user interface device에 사용되는 communication protocol이다. protocol로서는 직렬, 동기식, 양방향의 특성을 가지고 있다. 또한 device가 clock신호를 생성하는 특징을 가지고 있다. FPGA 보드와의 연결은 USB를 통하여 연결된다.

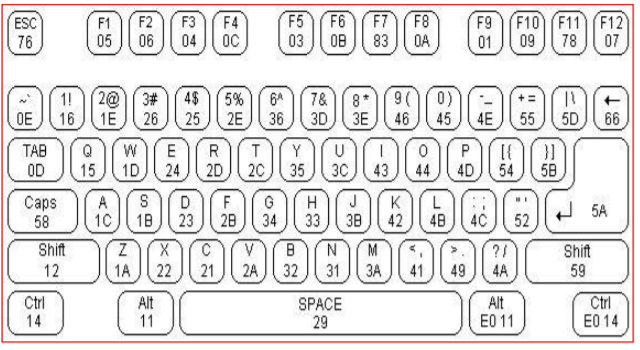
PS/2 Keyboard는 키보드로부터 PS/2 통신 프로토콜을 통해 신호를 입력 받는 모듈이다. 기본적인 동작과정은 다음과 같다.

키보드의 키를 누르면 키보드는 Host에게 키에 해당하는 값을 전송한다. 이 값을 make codes라고 부른다. Make codes는 대개 한 바이트이며, 경우에 따라 extension codes(E0)가 포함된 2 바이트의 make coded를 가지기도 한다.

키보드의 같은 키가 계속 눌러지는 경우, 같은 make codes를 반복해서 전송한다. 이를 typematic state라고 하며, 정해진 typematic delay 마다 한번씩 make codes를 전송하게 된다.

키보드의 키를 떼는 경우(Release), 해당하는 키의 break codes를 전송한다. Break codes는 0xF0 + make codes로 구성된다.

각 키보드 키에 해당하는 make codes는 다음과 같다.



<figure1>make codes number

* + - * 1. VGA

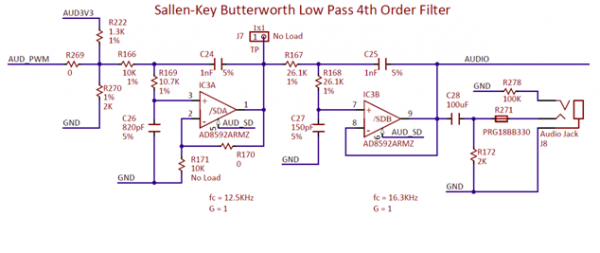
VGA는 Video Graphics Array의 약자로 아날로그 컴퓨터 display의 표준이다. 가로는 최대 800 pixel이고 세로는 최대 600줄이다. 화면의 refresh rate는 최대 70Hz이다. R, G, B 아날로그 신호를 이용하여 색을 표현하고, 각 R, G, B 아날로그 신호는 0.7V peak-to-peak으로 색을 표현한다.

VGA의 픽셀 당 표현 가능 컬러 수는 4096개이다. 빨강, 초록, 파랑 각각 4bit씩 할당해서 색을 표현 할 수 있다.

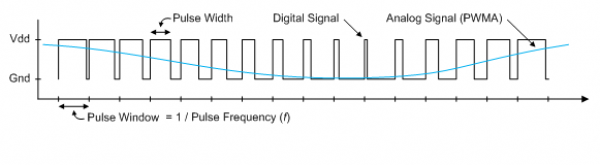
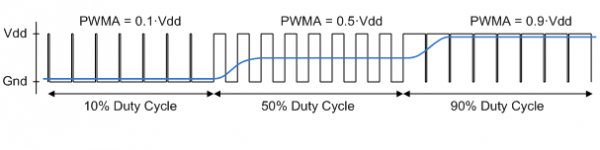
VGA는 보통 640 x 480의 non-interlace 방식을 가진다. 한 픽셀당 clock은 25.175 MHz를 가진다. 아래 그림처럼 R, G, B가 가질 수 있는 전압의 크기다 다르다. Vr = 0.117R3 + 0.059R2 + 0.029R1 + 0.015R0 의 식으로 각 색마다 걸리는 전압의 크기에 따라 Vr의 크기가 결정되고 이에따라 색깔이 결정된다.   
텍스트, 영수증이(가) 표시된 사진

자동 생성된 설명  
<figure2> VGA interface circuit

* + - * 1. PWM Audio

FPGA 보드의 J8 핀에 해당하는 오디오 잭은 오디오 출력을 제공하는 Sallen-Butterworth Low-Pass 4th Order Filter 기반으로 작동한다. Low-pass filter의 회로는 다음 그림과 같다.   
 <figure3> LPF circuit   
입력(AUD\_PWM)은 FPGA의 A11번 핀과 연결 되어있다. 디지털 입력은 일반적으로 FPGA에 의해 생성되는 PWM(Pulse Width Modulation) 또는 PDM(Pulse Density Modulation)의 open-drain signal이다. 신호는 논리 ‘0’의 경우 low로 작동되고, 논리 ‘1’의 경우 high로 작동된다.

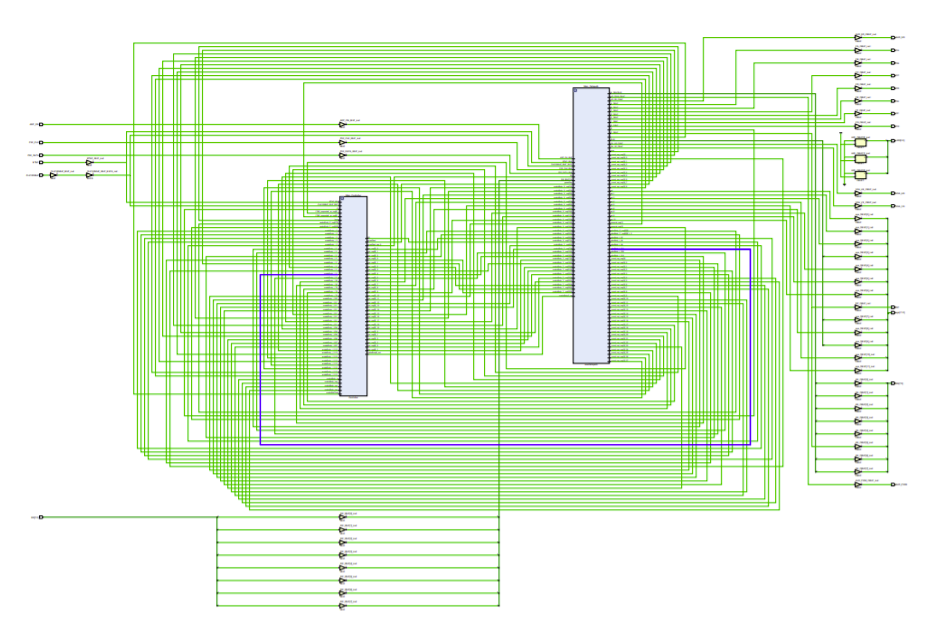
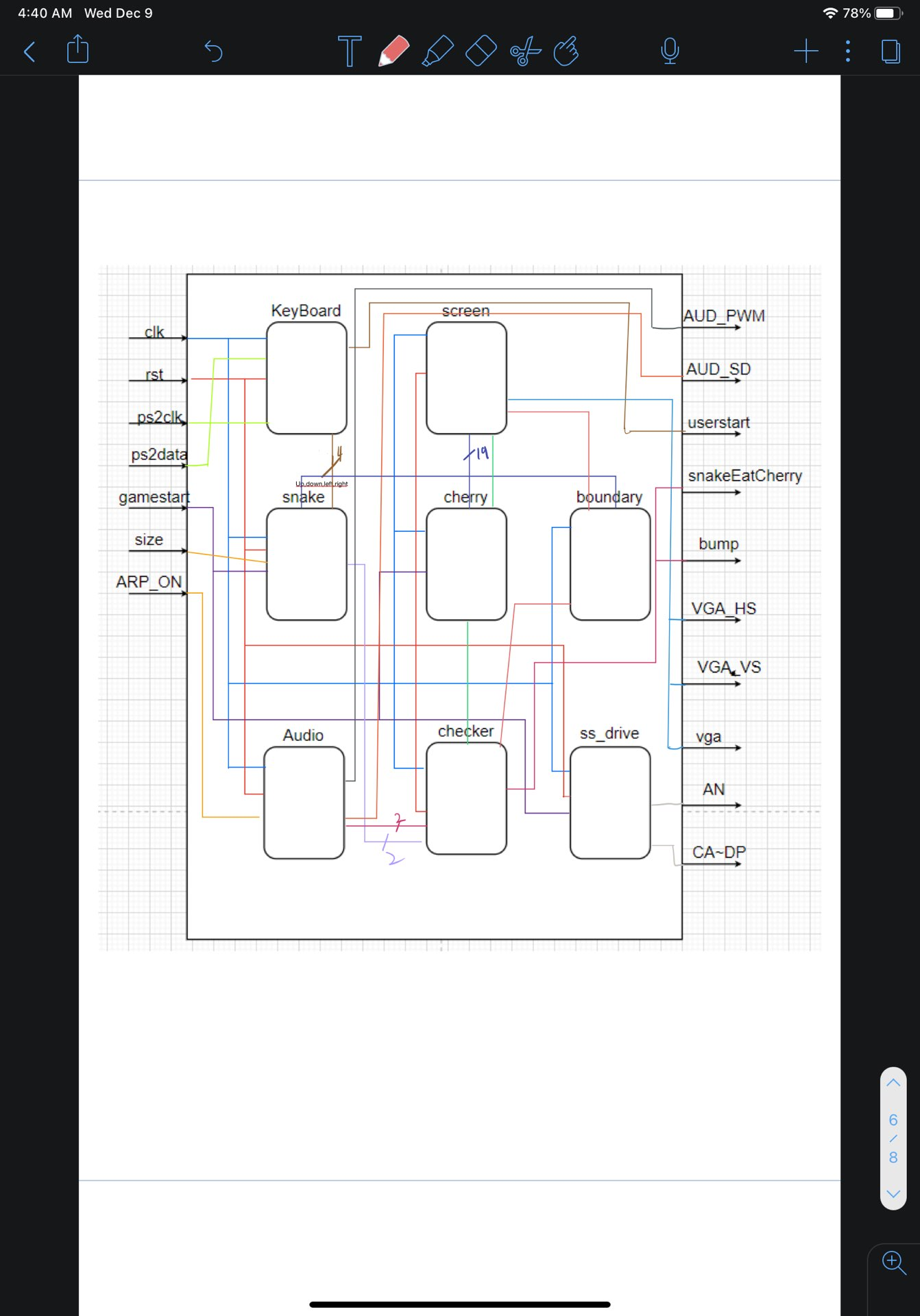
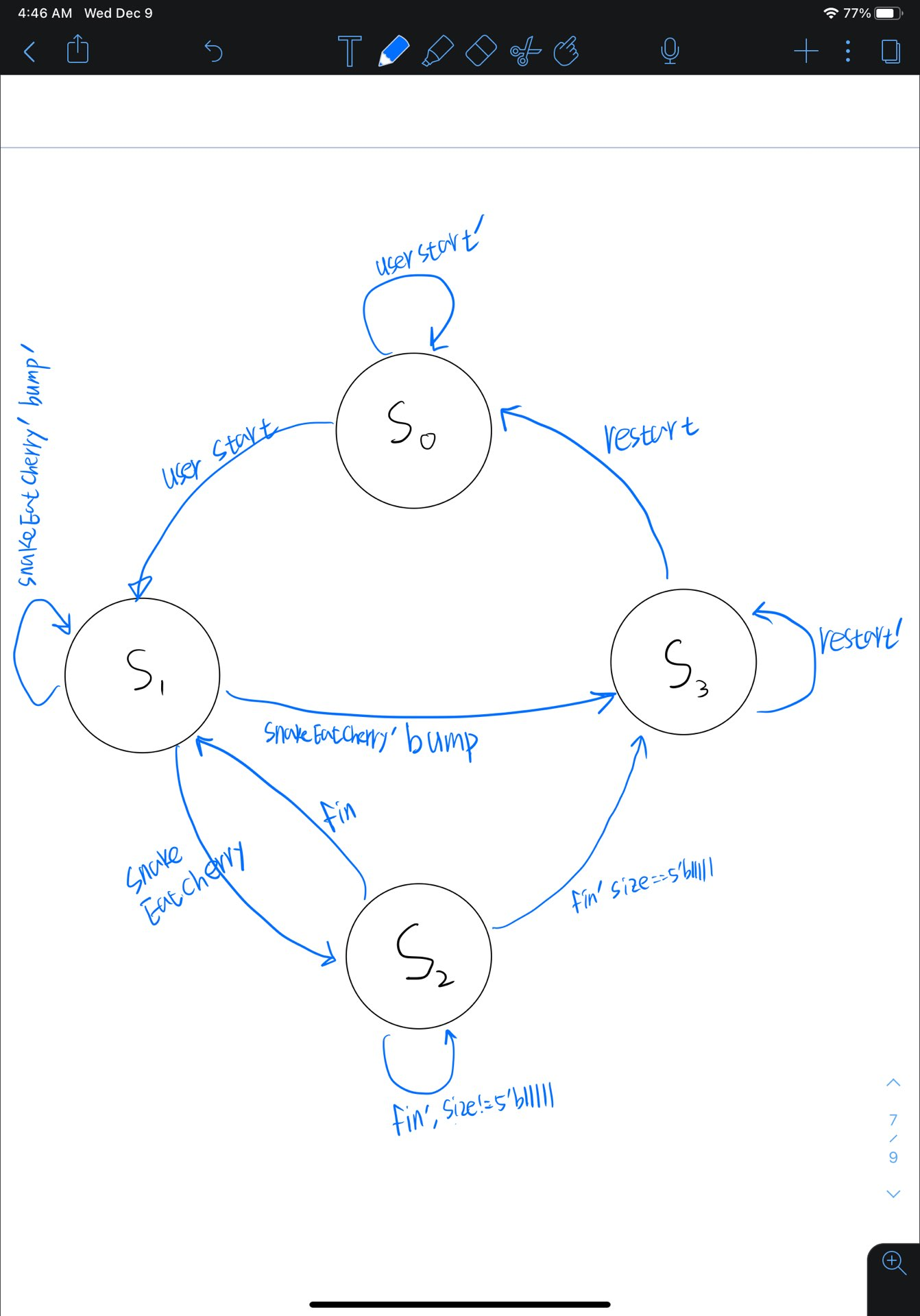
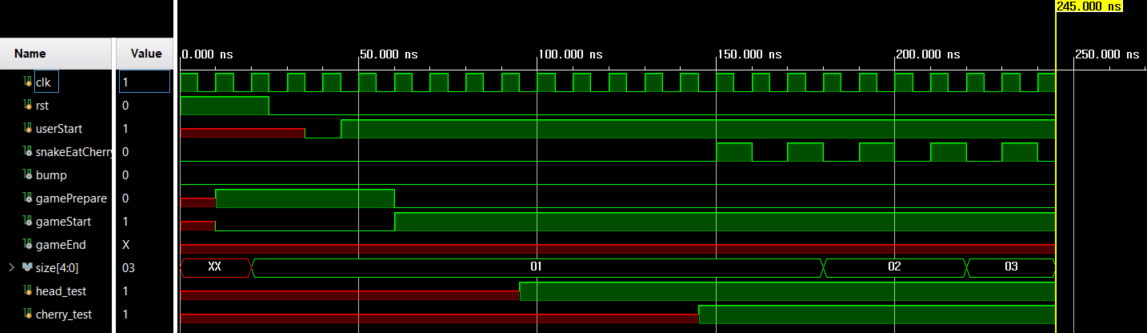
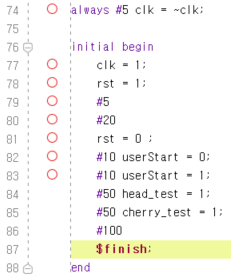
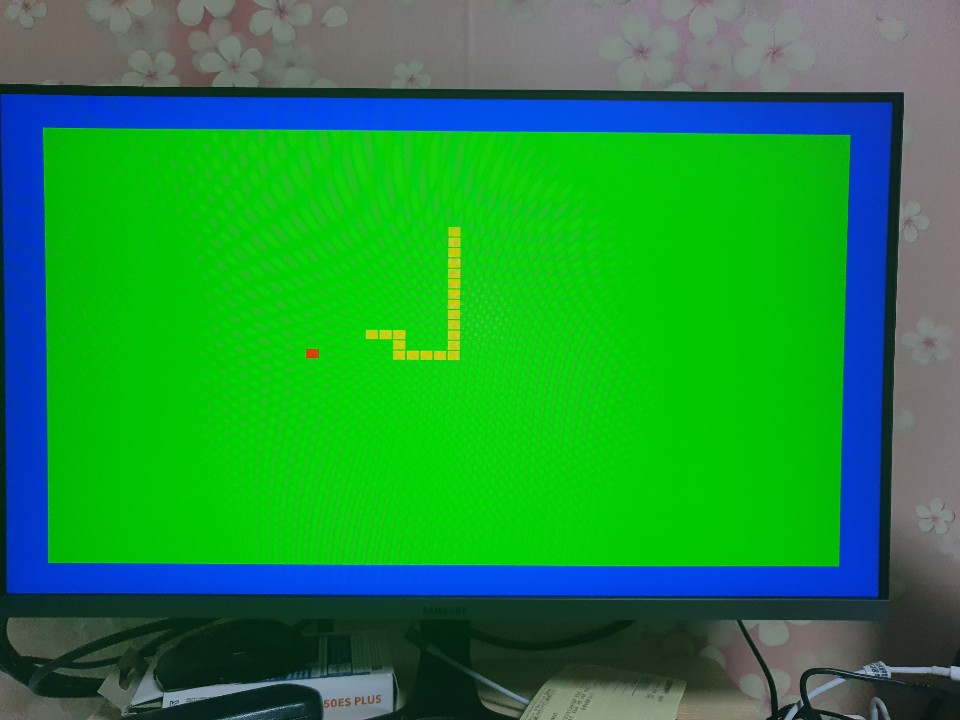
Pulse Width Modulation

PWM은 일부 고정 주파수에서 연쇄적인 pulse로 인식되기도 하며, 해당 pulse는 잠재적으로 다른 width를 가지게 된다. 이 디지털 신호는 디지털 파형을 통합하여 일정 간격 동안 평균 펄스 폭에 비례하는 아날로그 전압을 생성하는 simple low-pass filter를 통과할 수 있다. (low-pass filter와 pulse frequency는 3dB cut-off frequency에 의해 간격이 결정된다.) 예를 들어 사용 가능한 pulse의 주기의 평균 10% 동안 pulse가 높으면 integrator가 Vdd 전압의 10%인 아날로그 값을 생성한다. 아래 그림은 PWM 신호로 표시된 파형을 보여준다.  
  
 <figure4> simple waveform represented as PWM  
PWM 신호는 아날로그 전압을 정의하기 위해 통합되어야 한다. Low-pass filter 3dB 주파수는 PWM 주파수보다 작은 크기 순서로, PWM 주파수의 신호 에너지가 신호에서 필터링 되도록 해야 한다. 예를 들어, 오디오 신호에 최대 5KHz의 주파수 정보가 포함되어야 하는 경우 PWM 주파수는 최소 50KHz(그리고 더 높은 주파수)여야 한다. 일반적으로 아날로그 신호 충실도 측면에서 PWM 주파수가 높을수록 좋다. 아래 그림은 pulse train을 통합하여 출력 전압을 생성하는 PWM Integrator의 모습을 보여준다. Vdd에 대한 정상 상태 필터 출력 신호 진폭 비는 pulse width duty와 동일하다는 점에 주목해야한다. (duty cycle은 pulse-high time을 pulse-window time으로 나눈 값으로 정의됨).  
  
<figure5> Representation of a PWM integrator Producing an Output Voltage by Integrating the Pulse Train

* + - * 1. Block Memory

FPGA에는 메모리로 사용할 수 있는 공간이 따로 존재한다. Logic 영역을 이용해 data를 저장하는 것은 매우 비효율적이기 때문에 NEXYS4 DDR 보드의 XC7A100T 칩은 총 607.5KB의 Block RAM을 사용할 수 있게 해준다. 18kb의 block 270개와 36kb의 block 135개를 사용할 수 있기 때문이다.

COE FILE: COE파일은 확장자(.coe)가 붙는 파일로, Block RAM를 초기화하는데 사용한다. 이미지 파일을 COE. 파일로 변환하거나 다양한 형태의 DATA를 COE파일로 변환할 수 있다.

* 1. 설계
     1. Block Diagram (전체 블록들의 연결도)  
          
         <figure6> Block Diagram of Snakegame
     2. Datapath (논리 회로도)  
           
         <figure7> Logic block circuit of snakegame
     3. Controller 설계 (FSM 그림)  
          
        <figure8> FSM controller Diagram of snakegame
  2. 시뮬레이션
     1. 본 simulation은 snake game에서 snake가 cherry의 위치와 같아질 때(cherry를 먹을 때) 몸의 size가 증가하는 것을 확인하기 위한 simulation이다. 본래 snake를 control하기 위해서는 keyboard로부터 입력을 받아야 하지만, random\_number\_generator module로 인하여 cherry의 위치는 random한 위치로 변하기 때문에 simulation 환경에서 해당 위치를 직접 control을 통하여 알아내기는 한계가 존재한다. 그런 이유로 본 simulation에서는 snake가 cherry의 위치에 도달하였을 상황을 재현하기 위하여 head\_test(임의로 지정한 snake의 head 위치)와 cherry\_test(임의로 지정한 cherry의 위치)를 동일하게 하여 해당 조건이 만족할 경우, size가 증가하는 것을 확인할 수 있다.  
          
        <figure9> simulation plot of snakegame  
          
        <figure10> simulation code
  3. 구현 및 실험 결과
     1. 본 프로젝트를 위하여 위의 기능들을 모두 수행할 수 있도록 설계하였고, 필요한 장비들을 연결하여 실행한 결과 아래 사진과 같이 snake(노란색)이 cherry(빨간색)를 먹으면서 size up하는 것을 확인할 수 있었다.  
          
         <figure11> game screen of snakegame  
        스스로의 body에 충돌하거나 외벽(파란색)에 충돌하면 뱀의 움직임은 멈추고, 타이머가 종료되면서 게임은 종료된다.
     2. 게임이 시작되고, cherry를 먹게 되면, 특정 음성이 mono-audio를 통하여 출력되는데, 16번째 SW를 활성화하게 되면, 해당 음성이 좀 더 다양한 음성으로 변환되어 출력됨을 확인할 수 있었다.
  4. DISCUSSION
     + - 공개 코드 사용에 대한 구체적 내역
         1. 기존의 공개코드를 재사용한 블록 리스트와 기능

VGA

기존 25Hz의 Clock으로 분주하고, 조건들을 주어서 원하는 픽셀에 원하는 색깔이 표현 될 수 있도록 기존 사용법에 근거하여 동일하게 수행하도록 설계하였다.

PS/2

기존의 사용법과 동일하게 Keyboard 입력을 받아서 해당 키가 작동함에 따라 특정 기능을 수행하도록 설계하였다.

ss\_drive

기존의 사용법과 동일하게 clock을 조절하여 1초단위의 timer를 구성하여 game의 총 실행시간을 출력하도록 설계하였다.

* + - * 1. 스스로 전체를 설계한 모듈의 목록과 기능 설명

Top: 모든 module에 대한 입출력을 총괄하는 top-module의 기능 수행

newDatapath

FSM design에서 Datapath역할을 하는 모듈이다. 모든 Submodule(Snake, Cherry, Boundary etc..)들을 manage하고. Submodule의 데이터들을 서로 연결시켜주는 역할을 한다.

Controller와 연결되어 서로의 데이터를 주고받는다. Datapath의 data값으로 FSM의 state가 변화하고(게임준비, 게임시작, Snake Size증가, 게임종료 등), 또한 Controller로부터 받은 데이터들을 Submodule의 전달한다.

Controller

FSM design 의 각 state를 나타내는 Module이다. S0는 게임준비, S1은 게임플레이, S2는 snake size증가, S3는 게임 종료의 state이다.

초기상태는 S0로 Datapath로부터 게임시작 신호를 받으면 S1으로 state가 이동한다. 이후 S1에서 cherry를 먹은 신호를 받으면 S2로 이동하고, S1에서 bump신호를 받으면 S3로 이동하여 게임을 종료한다.

Snake

Snake의 모양 및 이동을 나타내는 모듈이다. Snake가 키보드로부터 direction에 대한 신호가 들어오면 이에 따라 방향을 바꾼다. 적당한 속도로 이동할 수 있도록 Snake의 속도를 지정한다. 또한 현재 진행하는 방향과 반대방향으로는 움직일 수 없도록 설정하였다.

Contoller로부터 오는 Size신호를 통해 Snake의 size를 증가시켜준다.

Snake의 Head와 Body의 데이터를 따로 저장하고, Snake의 이동에 따라 이전 그림이 VGA display 되지 않도록 한다.

Cherry

Clock 분주: 너무 빠르게 새로운 위치로 변경하게 되면, 기존의 위치와 가까운 위치에서 재 생성될 수 있기 때문에 cherry의 재생성 주기를 느리게 하기 위하여 clock을 분주한다.

Boundary안에서만 생성: cherry가 boundary 밖에 생성될 시, 정상적인 game이 진행될 수 없기 때문에 적절한 위치로 제한한다.

새로운 cherry 생성: snake가 cherry를 먹었다는 신호가 오면, 기존의 위치와 다른 새로운 위치로 cherry가 생성되도록 한다.

Ran\_num\_generator: LFSR를 사용하여, random한 값을 생성한다. 이는, bit-wise 연산을 하여 한 bit씩 shift하며 random한 값을 생성한다.

Boundary: 상하좌우 벽에 대한 위치정보를 checker에게 전달하는 기능을 수행한다.

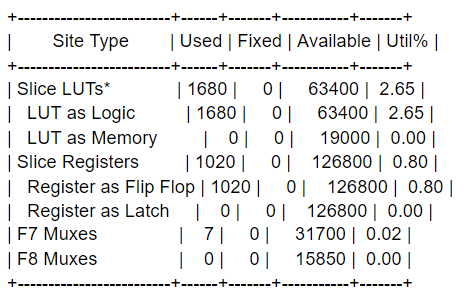
Checker: Snake, Cherry, Boundary로부터 신호를 받아 현재 module들이 어떠한 상태인지를 파악하여, Controller의 상태변화를 위한 신호를 determine하여 Controller에게 전해주는 기능을 수행한다.

pwm\_top: full-sine pulse로 되어진 COE로부터 특정 주파수에 해당되는 음을 포착하여 신호를 주는 기능을 수행한다.

Pwm\_module: pwm\_top에서 포착되는 신호를 기반으로 FPGA에 소리 신호를 전달하는 기능을 수행한다.

* + - * 목표대비 달성도:

|  |  |  |
| --- | --- | --- |
| 목표치 | 구현현황 | 달성율 |
| 키보드를 이용한 Snake 조작 | 키보드를 사용하여 자유조작 가능 | 100% |
| Random하게 생성되는 Cherry | random하게 생성 가능 | 100% |
| Cherry 먹을 시, Snake size up | Cherry 먹을 시, size 1 증가 | 100% |
| Cherry 먹을 시, Audio sound | Cherry를 먹거나 game over 조건일 시, Mono Audio Active | 100% |
| 특정 조건 시, game over | 자신의 몸에 부딪히거나, Boundary에 충돌 시 game over | 100% |
| 게임 전체 시간 출력 | 7-segment display를 통하여 시간 출력 가능 | 100% |
| VGA 화면 출력 | VGA를 통해 게임 화면 출력 가능 | 100% |

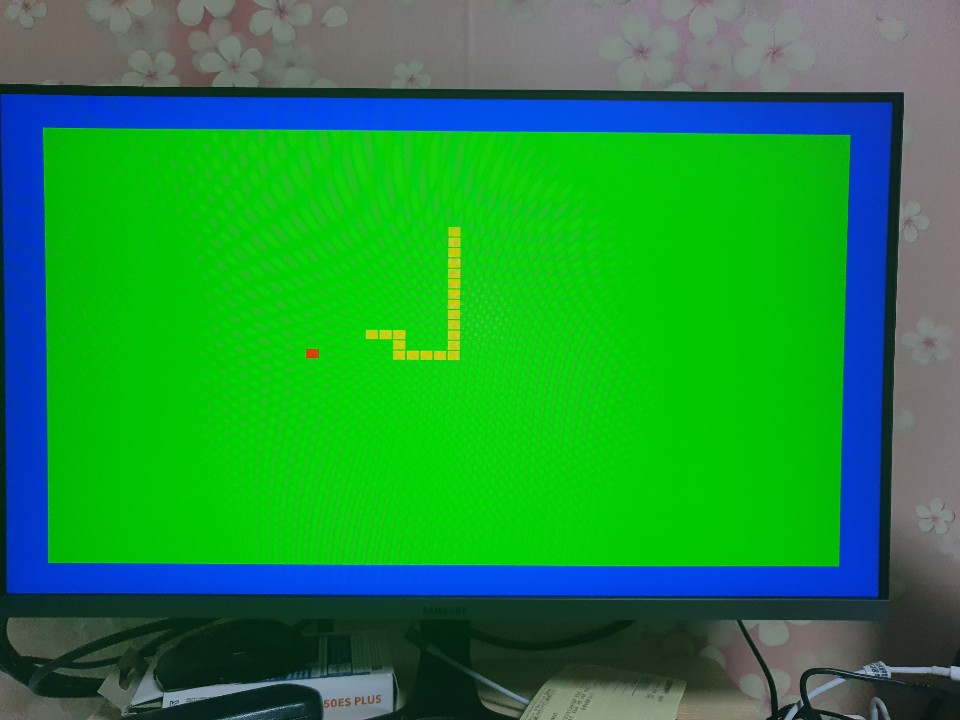
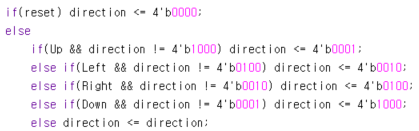
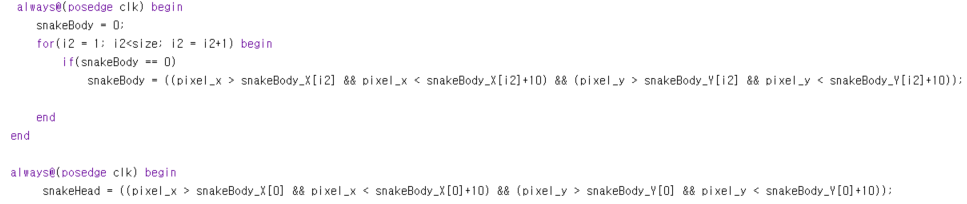
* + - * 성능 평가:
        1. Slice Logic  
           

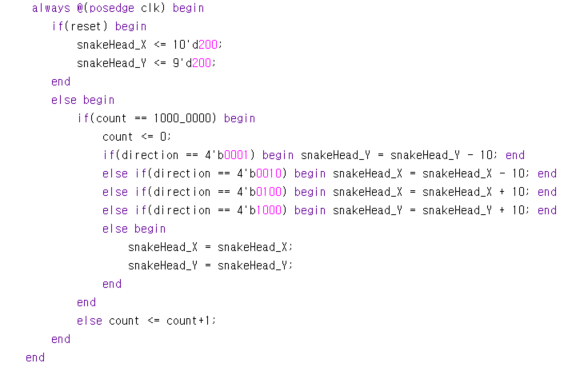
위의 사진에서 볼 수 있듯이 letch가 없는 것을 확인할 수 있다. 이는 안정적인 시스템을 구현하였다는 것을 알려준다.

* + - * 향후 추가 개발해야 할 내용 및 아쉬운 점
        1. FSM Design: snake game이 단순한 게임이기 때문에, state의 수가 많지 않았다. 좀 더 복잡한 사례에 대한 state 구성을 하는 경험을 하는게 도움이 많이 될 것 같다.
        2. PS/2: 기존의 내용과 크게 다르게 활용하지 못한 점이 아쉽다. 두가지의 키를 동시에 누를 때 간헐적으로 발생되는 오류를 잡아내야 한다는 점과, 같은 상황에서 원하는 신호가 누락되는 현상을 해결해야한다.
        3. VGA: Block memory를 활용하려 했으나 수많은 letch의 발생으로 포기한 점이 아쉽고, VGA의 소스코드의 핵심 부분들이 기존 코드와 거의 동일하기 때문에 직접 구현해보지 못한 점이 아쉽다.
        4. PWM Audio: open source로 제공되는 source code를 활용하면서, 다소 복잡하고 어려운 내용에 대한 구체적이고 정확한 이해를 하지 못하였다. 이러한 점들에 대한 이해가 더해진다면 좀 더 다채로운 음색과 기능을 수행할 수 있도록 설계할 수 있을 것 같다.
        5. Block Memory: 이번 project에서는 소리를 만들기 위한 full-sine data가 담긴 COE파일만을 사용했다면, 좀 더 다양한 type의 data를 COE로 활용하여 Block Memory를 좀 더 적극적으로 활용할 수 있을 것 같다.

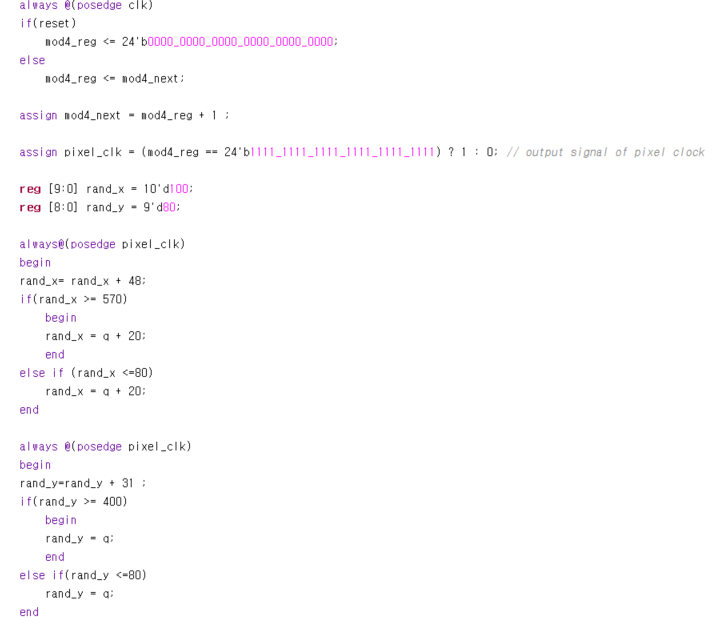
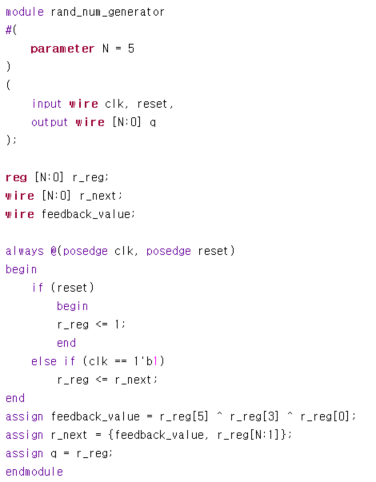
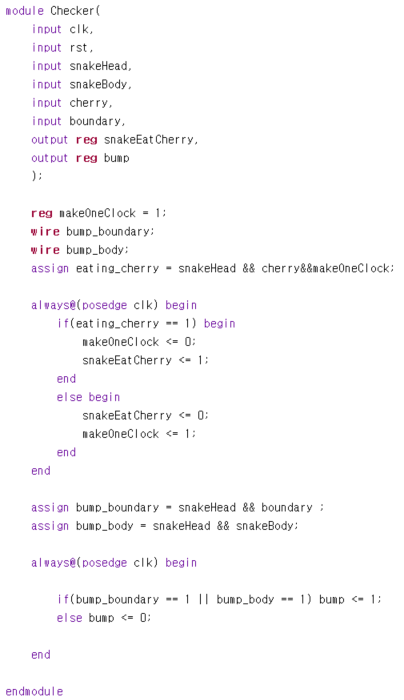
1. 부록
   1. Bit 파일 다운로드 받아서 사용하는 방법 (제품 매뉴얼에 해당) – 스크린샷이나 사진 포함
      1. FPGA의 Mono audio port에 이어폰을 연결하고, USB포트에 Keyboard를 연결한다.
      2. VGA포트를 통하여 모니터를 연결한다.
      3. FPGA에 해당 Bit 파일을 download한다.
      4. 게임 화면이 출력되면, ‘q’ 버튼을 눌러서 게임을 실행한다. (7-segment의 timer가 작동됨을 확인 가능)

  
<figure12> FPGA board displaying timer of snakegame

* + 1. ‘w’, ‘a’, ‘s’, ‘d’ 키는 각 상, 하, 좌, 우 조작으로 사용된다. (방향전환 시, 1번만 누르면 해당 방향으로 계속 움직이므로, 지속적으로 해당 키를 누를 필요가 없다.)
    2. 화면에 표시되는 cherry(붉은 블록)으로 snake(노란 블록)을 유도하여 길이를 늘려 최대길이 32개가 될 때까지 버틴다.  
         
        <figure13> game screen of snakegame
    3. Boundary(파란 블록)에 부딪히거나, snake 자신의 body에 충돌하는 경우, game over 판정으로 종료된다.
    4. 새롭게 game을 시작하기 위하여, FPGA위의 BTNC 버튼을 누른다. (이후 다시 1부터 반복)
  1. 소스코드 중 중요한 부분의 설명과 해설
     1. Snake  
           
        Keyboard의 input에 따라 snake의 방향을 바꿔주는 부분이다. snake게임에서 snake는 이동방향의 반대방향으로 이동할 수 없기 때문에 조건을 주어 반대방향으로 이동하지 못하도록 만들었다.  
        
     2. Snake의 Head와 Body의 위치를 구하는 부분이다. 여기서 나오는 snakeHead와 snakeBody를 Screen에 보내어 Snake의 전체 모양을 출력한다. 또한 snakeHead는 snake의 몸의 부분 중 제일 먼저 이동하는 부분이므로 snakeHead를 이용하여 bump 와 snakeEatCherry determine하기 위한 조건으로 사용된다.



Snake의 속도를 조정해주는 코드이다. 저기서 이전에 미리 integer type으로 선언된 count가 1000\_0000 이 되면 snakeHead를 이동시킨다. snakeHead는 또한 snakeBody의 첫번째 부분이므로, 결국 snakeHead가 움직이면 snake전체가 움직이게 된다.

* + 1. Cherry + random  
         
          
       random 위치로 생성되는 cherry의 위치가 boundary의 영역에 들어가거나 적절하지 못한 경우에 대하여 조건문을 통하여 적절한 범위내에서 random한 위치가 지정되도록 설정하는 코드이다.  
         
       해당 rand\_num\_generator module은 LFSR이론을 구현한 것이다. LFSR은 input의 5, 3, 0번째 위치에 해당하는 값을 택로 연결되어 output으로 보내는 형식으로 구성된다. 최초 LFSR값은 1로 초기화되어 설정된 된다, 그 뒤, feedback\_value를 5, 3, 0번째 위치에 해당하는 값들로 구성하고, 기존의 LFSR값의 마지막 5비트를 1비트만큼 오른쪽으로 이동시키고, 빈 자리를 feedback\_value로 구성한다. 이런 과정을 통하여 random한 값이 생성된다.
    2. Checker  
       

Snake가 어떠한 상태인지를 determine하는 모듈이다. Snake로부터 snakehead의 위치에 대한 정보를 가지고 있는 snakehead를 받고, cherry로부터 cherry의 위치에 대한 정보를 가지고 있는 cherry를 받고, Boundary로부터 boundary의 위치에 대한 정보를 가지고 있는 boundary를 받는다. 그후 이 3가지 신호들을 통해 snake가 cherry를 먹었는지, 벽에 부딪혔는지의 여부를 결정한다. 또한 makeOneClock register는 snake의 상태의 여부를 나타내는 신호를 딱 한 clock만 생성 할 수 있도록 만들어주는 register이다. 이를 통해 Controller에게 단 snake의 상태에 대한 단 한 clock의 신호를 보낸다.

* 1. 전체 소스코드 구성  
     