**13주차 결과 보고서**

남민혁

**1.** **4-bit Shift Register의 결과 및 Simulation 과정에 대해서 설명하시오. (verliog source, 출력 예시, 과정 상세히 적을것!)**

4-bit shift register는 clock에 의하여 신호가 들어올 때 왼쪽에 들어온 입력이 다음 메모리로 밀리는 논리회로이다. 구체적인 입력과 그에 따른 출력의 결과는 다음과 같다.

**- State Table**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| # of Clock Transitions | Output | | | | |
| Clock Tick | IN | L1 | L2 | L3 | L4 |
| 1 | 1 | 1 | 0 | 0 | 0 |
| 2 | 0 | 0 | 1 | 0 | 0 |
| 3 | 0 | 0 | 0 | 1 | 0 |
| 4 | 0 | 0 | 0 | 0 | 1 |
| 5 | 1 | 1 | 0 | 0 | 0 |
| 6 | 0 | 0 | 1 | 0 | 0 |
| 7 | 0 | 0 | 0 | 1 | 0 |

시뮬레이션 코드 기준으로 clock tick은 5초 단위로 발생하고, 가장 왼쪽에 있는 메모리에 새로운 input이 20초 단위로 발생하는 것을 기준으로 작성하였다. Clock tick이 있을 때 shift하도록 하는 입력이 있으면 (시뮬레이션 코드 기준으로 3초 단위로 발생하여, 엇갈리는 경우도 있으나, 논의의 편의를 위하여 clock tick의 주기와 shift 입력의 주기가 동일하다고 가정하자) 첫번째 메모리에서 두번째 메모리로, 두번째에서 세번째로, 세번째에서 네번째로 이동한다.

**-Verilog 코딩**

Shift register구현을 위하여 module shift(shr,rst,shr\_in,clk,Q);로서 모듈을 선언하였으며, 각각의 입력에 따라 state table에서 정해진 결과를 출력하도록 코딩하였다. Shr는 shift를 하도록 하는 입력신호, rst는 모든 메모리를 0으로 초기화하는 reset 입력이다. Shr\_in은 가장 좌측에 있는 메모리에 들어오는 입력값, Clk은 Clock tick이며, 마지막으로 Q는 메모리 array이다. Clk 신호가 있을 때, Rst가 있으면 모든 Q를 0으로 초기화하고, Shr이 있을 경우 Shr\_in을 첫번째 Q에, 차례로 오른쪽으로 밀리는 방식으로 구현하였다.

각 행은 논리회로 구성을 위한 Verilog 코드, 이를 바탕으로 모든 가능한 입력으로 부터 기대되는 출력이 나타나는지를 확인하기 위한 Test bench 코드이다.

|  |  |
| --- | --- |
| Verilog | `timescale 1ns / 1ps  module shift(shr,rst,shr\_in,clk,Q);  input shr, shr\_in;  input clk, rst;  output [3:0] Q;  reg [3:0] R;  always @(posedge clk) begin  if(rst == 1) begin  R<=4'b0000;  end  else if (shr ==1) begin  R[3] <= shr\_in; R[2] <= R[3];  R[1] <= R[2]; R[0] <= R[1];  end  end  assign Q =R;  endmodule |
| Test Bench | `timescale 1ns / 1ps  module shift\_tb();  reg clk;  reg rst;  reg shr,shr\_in;  wire [3:0] out;  shift connect(shr,rst,shr\_in,clk,out);  always #5 clk = ~clk;  always #3 shr = ~shr;  always #20 shr\_in = ~shr\_in;  initial begin  shr <= 0;  shr\_in <= 1;  clk <= 0;  rst <= 0;  #20 rst <=0;  #80 rst <=1;  #50 rst <=0;  end  endmodule |

**-Simulation 출력 결과 비교**

**A picture containing text, electronics

Description automatically generated**

모든 가능한 입력 값에 대한 출력값을 확인하기 위하여 Simulation을 시행한 결과 의도한 진리표와 모든 결과가 동일함을 확인하였다.

**-출력 예시**

비대면 수업으로 인하여 실제 FPGA로 부터 입력을 받지 못하였으나, 기존 실험때와 마찬가지로CLK 과 SHR, RST, SHR\_IN비트를 스위치로, 각 메모리의 상태를 LED에 매칭한 후 불이 들어오는 것을 1로 가정하면, SHR이 켜져있고, CLK 스위치가 한번 왔다 갔다 할 때마다 좌측에서 우측으로 계속 입력 값이 이동하는 것을 확인할 수 있을 것이다. 이는 위의 Simulation 결과를 통하여 확인할 수 있다.

**-구현된 Schematic**

Diagram, schematic

Description automatically generated

**2. 4-bit Ring counter의 결과 및 Simulation 과정에 대해서 설명하시오. (verliog source, 출력 예시, 과정 상세히 적을것!)**

4-bit Ring counter는 clock에 의하여 신호가 들어올 때 마다 1이 각 메모리에서 계속 순환하는 논리회로이다. 마지막 메모리에 도달할 경우 다시 처음으로 돌아가도록 구성된 논리회로이다. 구체적인 입력과 그에 따른 출력의 결과는 다음과 같다.

**-Truth Table**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # of Clock Transitions | Output | | | |
| Clock Tick | L1 | L2 | L3 | L4 |
| 1 | 1 | 0 | 0 | 0 |
| 2 | 0 | 1 | 0 | 0 |
| 3 | 0 | 0 | 1 | 0 |
| 4 | 0 | 0 | 0 | 1 |
| 5 | 1 | 0 | 0 | 0 |
| 6 | 0 | 1 | 0 | 0 |
| 7 | 0 | 0 | 1 | 0 |

Clock tick이 있을 때 마다 왼쪽에 저장된 1의 값이 오른쪽으로 이동하며, L4에 도달한 경우 다시 1은 L1으로 이동한다.

**-Verilog 코딩**

Ring counter구현을 위하여 module ring(clk,clr,Q);로서 모듈을 선언하였으며, 각각의 입력에 따라 state table에서 정해진 결과를 출력하도록 코딩하였다. Clk은 Clock tick이며, Clr는 첫번째 메모리에만 1을 부여하고, 나머지는 0으로 초기화하는 입력이다. 마지막으로 Q는 메모리 array이다. Clk 신호가 있을 때, Clr 신호가 있으면 첫번째 메모리에는 1을 부여하고 나머지 Q를 0으로 초기화하며, 나머지 상황에서는 지속적으로 1이 다음 메모리로 이동하도록 구현하였다.

각 행은 논리회로 구성을 위한 Verilog 코드, 이를 바탕으로 모든 가능한 입력으로 부터 기대되는 출력이 나타나는지를 확인하기 위한 Test bench 코드이다.

|  |  |
| --- | --- |
| Verilog | `timescale 1ns / 1ps  module ring(clk,clr,Q);  input clk;  input clr;  output [3:0] Q;  reg [3:0] Q;  always @(posedge clk or posedge clr)  begin  if(clr==1)  Q<=1;  else begin  Q[3] <= Q[0];  Q[2:0] <= Q[3:1];  end  end  endmodule |
| Test Bench | `timescale 1ns / 1ps  module ring\_tb();  reg clk,clr;  wire [3:0] Q;  ring connect(clk,clr,Q);  always #5 clk = ~clk;  initial begin  clk <= 0;  clr <= 0;  #20 clr <=0;  #80 clr <=1;  #50 clr <=0;  end  endmodule |

**-Simulation 출력 결과 비교**

**A picture containing graphical user interface

Description automatically generated**

모든 가능한 입력 값에 대한 출력값을 확인하기 위하여 Simulation을 시행한 결과 의도한 진리표와 모든 결과가 동일함을 확인하였다.

**-출력 예시**

비대면 수업으로 인하여 실제 FPGA로 부터 입력을 받지 못하였으나, 기존 실험때와 마찬가지로CLK과 CLR 비트를 스위치로, 각 비트를 순서대로 LED에 불이 들어오는 것으로서 1로 가정하면, CLR은 꺼지고, CLK 스위치가 한번 왔다 갔다 할 때마다 0001-> 1000 -> 0100 -> 0010 -> 0001 순서로 계속 순환하는 것을 LED로서 확인할 수 있을 것이다. 이는 위의 Simulation 결과를 통하여 확인할 수 있다.

**-구현된 Schematic**

Diagram, schematic

Description automatically generated

**3. 4-bit Up/Down counter 의 결과 및 Simulation 과정에 대해서 설명하시오. (verliog source, 출력 예시, 과정 상세히 적을것!)**

4-bit Up/Down counter는 clock에 의하여 신호가 들어올 때 마다 up 비트의 여부에 따라 증가하거나, 감소하는 방향으로 진행방향이 결정되는 Counter이다. 구체적인 입력과 그에 따른 출력의 결과는 다음과 같다.

**-Truth Table**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| # of Clock Transitions | Up Counter Output | | | | |
| Clock Tick | L1 | L2 | L3 | L4 | Display |
| 1 | 0 | 0 | 0 | 1 | U |
| 2 | 0 | 0 | 1 | 0 | U |
| 3 | 0 | 0 | 1 | 1 | U |
| 4 | 0 | 1 | 0 | 0 | U |
| 5 | 0 | 1 | 0 | 1 | U |
| 6 | 0 | 1 | 1 | 0 | U |
| 7 | 0 | 1 | 1 | 1 | U |

|  |  |
| --- | --- |
| # of Clock Transitions | Down Counter Output |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Clock Tick | L1 | L2 | L3 | L4 | Display |
| 1 | 0 | 0 | 0 | 0 | D |
| 2 | 1 | 1 | 1 | 1 | D |
| 3 | 1 | 1 | 1 | 0 | D |
| 4 | 1 | 1 | 0 | 1 | D |
| 5 | 1 | 1 | 0 | 0 | D |
| 6 | 1 | 0 | 1 | 1 | D |
| 7 | 1 | 0 | 1 | 0 | D |

Up Counter의 경우 (Up bit가 1인 경우)에는 Clock tick이 있을 때 마다 이진수가 점차 증가하는 반면, Down Counter의 경우 (Up bit가 0인 경우) 에는 Clock tick이 있을 때 마다 이진수가 점차 감소한다.

**-Verilog 코딩**

Ring counter구현을 위하여 module updown(clk,up,clr,Q,x,seg);로서 모듈을 선언하였으며, 각각의 입력에 따라 state table에서 정해진 결과를 출력하도록 코딩하였다. Clk은 Clock tick이며, Up은 up counter인지 여부를 결정한다. Clr는 0인 경우 모든 비트를 0으로 초기화하고, 반대의 경우 counter가 진행하도록 하는 입력이다. 마지막으로 Q, x, seg는 연산을 위해 필요한 register이며, truth table에서 L로 표현된 출력은 Q 메모리 array에 저장된다. Clk 신호가 있고, Clr 신호가 없을 때, Up의 신호가 있으면, 0부터 F까지 증가하고, 반대로 Up 의 신호가 없는 경우 현재 상태에서 감소하도록 구현하였다.

각 행은 논리회로 구성을 위한 Verilog 코드, 이를 바탕으로 모든 가능한 입력으로 부터 기대되는 출력이 나타나는지를 확인하기 위한 Test bench 코드이다.

|  |  |
| --- | --- |
| Verilog | `timescale 1ns / 1ps  module updown(clk,up,clr,Q,x,seg);  input clk,up;  input clr;  output [3:0] Q;  output x;  output [6:0] seg;  reg [3:0] Q;  reg [6:0] seg;  always @(posedge clk or posedge clr) begin  if(~clr) begin  Q <=0;  seg = 7'b1111111;  end  else if(up) begin  seg = 7'b0111110;  Q=Q+1;  end  else begin  seg = 7'b0111101;  Q=Q-1;  end  end  assign x = 1;  endmodule |
| Test Bench | `timescale 1ns / 1ps  module updown\_tb();  reg clk,clr,up;  wire [3:0] Q;  wire x;  wire [6:0] seg;  updown connect(clk,up,clr,Q,x,seg);  always #5 clk = ~clk;  always #20 up = ~up;  initial begin  up<=1;  clk <= 0;  clr <= 0;  #20 clr <=0;  #80 clr <=1;  #40 clr <=0;  #20 clr <=1;  #30 clr <=0;  end  endmodule |

**-Simulation 출력 결과 비교**

**A picture containing text, electronics

Description automatically generated**

모든 가능한 입력 값에 대한 출력값을 확인하기 위하여 Simulation을 시행한 결과 의도한 진리표와 모든 결과가 동일함을 확인하였다.

**-출력 예시**

비대면 수업으로 인하여 실제 FPGA로 부터 입력을 받지 못하였으나, 기존 실험 때와 마찬가지로CLK, CLR, UP 비트를 스위치로, 각 비트를 순서대로 LED에 불이 들어오는 것으로서 1로 가정하면, UP비트가 켜져있고, CLK 스위치가 한번 왔다 갔다 할 때마다 (0으로 초기화 된것을 가정하면) 0000부터 1111까지 증가하는 과정을 거치며, 반대로 UP 비트가 꺼질 경우 반대로 진행되는 것을 LED로서 확인할 수 있을 것이다. 이는 위의 Simulation 결과를 통하여 확인할 수 있다.

**-구현된 Schematic**

Diagram, timeline

Description automatically generated

**4. 결과 검토 및 논의 사항.**

이번주 실험에서는 Shift register, Ring counter, Up/Down를 구현하였다. 이를 통하여 12주차에서 구현한 여러 카운터의 개념을 확장할 수 있었다. State Table을 작성한 후 조건문을 사용하여 모든 가능한 상황에 대한 회로의 행동을 제어하는 방식으로 Verilog 코드를 작성하였다. 이후 모든 가능한 입력에 대해 simulate 하여 기대되는 결과가 나타나는지 확인하였다. 그 결과 모든 디자인이 의도한 State Table와 다르지 않고 동일하게 결과가 출력됨을 확인하였다.

**5. 추가 이론 조사 및 작성.**

이번 주에 구현한 시프트 레지스터를 활용하여 다음과 같은 응용을 할 수 있다.

-재순환 자리 이동 레지스터 (Recirculating Shift Register)

자리 이동이 되어서 지워지는 데이터를 보존하기 위하여 레지스터의 재순환 기능을 넣어 출력을 입력으로 다시 채워넣은 방식의 레지스터이다. 도식으로 나타내면 다음과 같다.

Diagram

Description automatically generated

- 비밀번호 확인

시프트 레지스터를 활용하여 순차적으로 비밀번호의 입력이 맞는지 확인하는 논리회로를 구현할 수 있다. 예를 들어 3190이 비밀번호일 경우 각각 3190 순서에 맞추어 각 Flipflop의 클록신호로서 연결하고 나머지 입력은 NOR게이트를 통해서 입력이 되도록 한다. 3190이라는 입력을 받을 경우 Shift register에 의하여 순차적으로 값이 입력되어 비밀번호를 확인할 수 있다. 아래는 이를 도식으로 나타낸 모습이다.

Diagram

Description automatically generated

**6. 참고 문헌**

강석태, “Verilog HDL Summary”, <http://vlsi.hongik.ac.kr/lecture/%EC%8B%A4%ED%97%98/Verilog_Summary.pdf>.

박재동, “[Verilog 예제] CLA (Carry Look Ahead) 모듈”, <http://egloos.zum.com/rabe/v/1285020>.

한국기술교육대학교 전기전자통신공학부, “디지털시스템 설계 및 실습” Verilog HDL 문법 강의자료, <https://cms3.koreatech.ac.kr/sites/yjjang/down/dsys11/M01_VerilogHDL01.pdf>.

장영조, “디지털공학 및 실습 7장 강의자료: 카운터와 레지스터”, <https://cms3.koreatech.ac.kr/sites/yjjang/down/digi2000/chap7.pdf>.

장영조, “디지털공학 및 실습 11장 강의자료: 레지스터”, <https://cms3.koreatech.ac.kr/sites/yjjang/down/dig13/ch11_reg.pdf>.