전공: 컴퓨터공학 학년: 2학년 학번: 20161603 이름: 신민준

1. SR Latch의 결과 및 simulation 과정에 대해서 설명하시오.

SR Latch는 S-R 입력의 조합이 10일때 값을 1로 ‘set’ 하고, 조합이 01일 때는 값을 0으로 ‘reset’하며, 입력 값이 00으로 주어졌을 때는 현재 값을 그대로 유지하는 간단한 latch이다. 이를 진리표로 나타내었을 때, 다음처럼 표현이 가능하다.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Input | | | Output | |
| 순서 | R | S | Q | ~Q |
| 1 | 0 | 1 | 1 | 0 |
| 2 | 0 | 0 | 1 | 0 |
| 3 | 1 | 0 | 0 | 1 |
| 4 | 0 | 0 | 0 | 1 |
| 5 | 1 | 0 | 0 | 1 |
| 6 | 1 | 1 | x | x |

위와 같은 진리표를 Verilog code로 구현한 SR Latch 모듈은 다음과 같다.

|  |
| --- |
| `timescale 1ns / 1ps  module SRLatch(en, s, r, q, qbar);  input en, s, r;  output q, qbar;  reg q;  assign qbar = ~q;  always @(en or s or r) begin  if(en) begin  if((s==1)&&(r==1))  q <= 1'bx;  else if((s==1)&&(r==0))  q <= 1'b1;  else if((s==0)&&(r==1))  q <= 1'b0;  else  q <= q;  end  end  endmodule |

S와 R의 가능한 4가지의 모든 조합에 대해 올바른 값을 q에 전달해주는 방식으로 구현한 모습인데, 이 때 SR 조합이 11인 경우는 고려하지 않아야 하므로, don’t care 값인 x를 넘겨주도록 했다. 이렇게 구현한 모듈의 RTL schematic 회로도는 다음과 같았다.

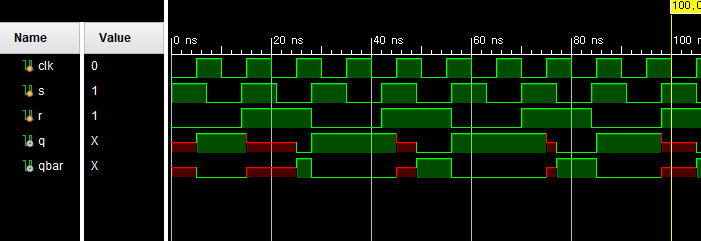
지도, 텍스트이(가) 표시된 사진

자동 생성된 설명

구현한 모듈이 의도한 대로 정확하게 구현되었는지 확인하기 위해, 다음과 같이 Simulation code를 작성해 확인하고자 했다. 이 때, 의도적으로 기준이 되는 클럭 값과 S, R값을 서로 다른 주기로 변화시켜 다양한 input에 대한 결과를 확인하고자 했다.

|  |
| --- |
| `timescale 1ns / 1ps  module srlatchsim;  reg clk, s, r;  wire q, qbar;    SRLatch srlatch(clk, s, r, q, qbar);    initial clk=1'b0;  initial s=1'b1;  initial r=1'b0;    always clk = #5 ~clk;  always s = #7 ~s;  always r = #14 ~r;    initial begin  #1600  $finish;  end  endmodule |

위 코드를 실행한 결과, 다음처럼 결과가 나왔음을 확인할 수 있었다.



위 결과에서 일 때 값이 둘 다 don’t care 값으로 변하는 것을 확인할 수 있었고, 다른 경우의 입력에 대해서도 의도한 대로 정확하게 결과값이 나옴을 볼 수 있다. 또한, clock 신호 clk의 값이 1인 동안에는 입력값에 생기는 변화가 결과값에 즉각 반영된다는 사실도 확인할 수 있기에, latch의 transparent한 성질을 볼 수 있다.

1. SR Flip-Flop의 결과 및 simulation 과정에 대해서 설명하시오.

SR Flip-Flop의 진리표는 아래와 같이, SR Latch와 동일하다.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Input | | | Output | |
| 순서 | R | S | Q | ~Q |
| 1 | 0 | 1 | 1 | 0 |
| 2 | 0 | 0 | 1 | 0 |
| 3 | 1 | 0 | 0 | 1 |
| 4 | 0 | 0 | 0 | 1 |
| 5 | 1 | 0 | 0 | 1 |
| 6 | 1 | 1 | x | x |

그러나, Flip-flop은 latch와는 다르게 leading/trailing edge에서의 순간적인 input 값을 읽어와 읽은 값만을 반영한다. 따라서, 이 차이를 Verilog code에서 구현하기 위해, 키워드 posedge / negedge를 사용해 다음과 같은 모듈을 작성했다.

|  |
| --- |
| `timescale 1ns / 1ps  module SRFlipFlop(  clk, s, r, q, qbar  );  input clk, s, r;  output q, qbar;  reg q;  assign qbar = ~q;  always @(posedge clk) begin  if((s==1)&&(r==1))  q <= 1'bx;  else if((s==1)&&(r==0))  q <= 1'b1;  else if((s==0)&&(r==1))  q <= 1'b0;  else  q <= q;  end  endmodule |

모듈의 verilog code를 확인해보면, always문 안의 조건을 제외하면 코드에 차이가 없음을 확인할 수 있다. 로 조건을 정함으로써 latch에서 사용한 조건들이 leading edge에서만 실행되도록 구현했다. 위 소스를 회로도로 나타내면 다음과 같은 다이어그램으로 나타내어진다.

텍스트, 지도이(가) 표시된 사진

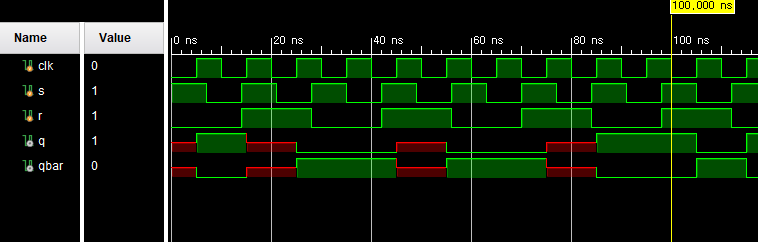
자동 생성된 설명

SR latch의 다이어그램과는 사용된 메모리 소자가 clock을 받지 않는다는 점에서 차이가 있다.

위 구현한 모듈을 검사하기 위해 작성한 simulation code는 SR latch에서 사용한 코드와 동일하게 작성했다.

|  |
| --- |
| `timescale 1ns / 1ps  module srffsim;  reg clk, s, r;  wire q, qbar;    SRFlipFlop srflipflop(clk, s, r, q, qbar);    initial clk=1'b0;  initial s=1'b1;  initial r=1'b0;    always clk = #5 ~clk;  always s = #7 ~s;  always r = #14 ~r;    initial begin  #1600  $finish;  end  endmodule |

서로 다른 주기로 clk, s, r 값을 변화시켜 변화값을 확인한 결과는 다음과 같았다.



위 결과로부터, Q 값의 갱신이 반드시 leading edge에서만 일어난다는 점과 그 결과가 SR Latch의 결과와 일치함을 확인할 수 있다.

1. D Latch의 결과 및 simulation 과정에 대해서 설명하시오.

D Latch는 입력으로 하나의 bit을 받으며, 1일 때는 값을 1로 set, 0일 때는 값을 0으로 reset하는 latch이다. 이 때, 하나의 enable signal을 추가하면, 이 D latch의 진리표는 다음처럼 나타낼 수 있다.

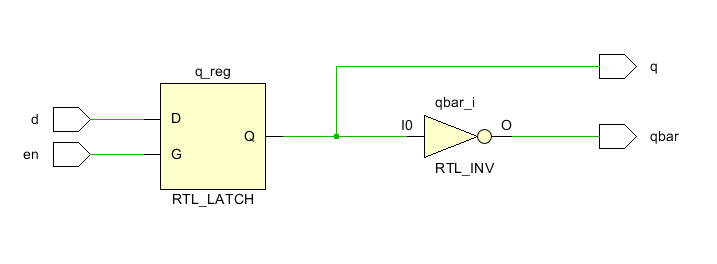
|  |  |  |  |
| --- | --- | --- | --- |
| Input | | Output | |
| D | en | Q | ~Q |
| 0 | 0 | N/A | N/A |
| 0 | 1 | 0 | 1 |
| 1 | 0 | N/A | N/A |
| 1 | 1 | 1 | 0 |

이 입력으로 들어오면, 이 latch는 이전의 값을 그대로 유지한다.

그러므로, 위와 같은 특성을 Verilog code로 구현해 작성한 모듈은 다음과 같다.

|  |
| --- |
| `timescale 1ns / 1ps  module DLatch(  en, d, q, qbar  );  input en, d;  output q, qbar;  reg q;  assign qbar = ~q;  always @(en or d) begin  if(en)  q <= d;  end  endmodule |

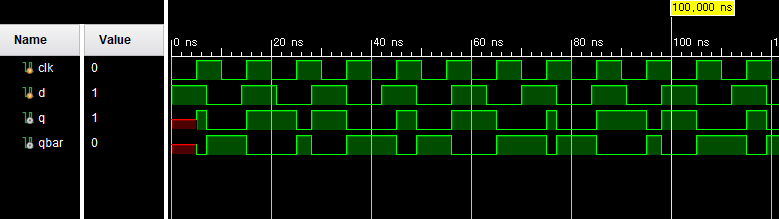
이라면, 값이 입력받은 값으로 바뀌는 것을 확인할 수 있다. 아래의 회로도로 위 모듈을 나타낼 수 있었는데, 회로도에서 D Latch를 사용했기에 별 다른 combinational circuit의 처리 없이 결과값을 바로 나타낼 수 있음을 확인할 수 있다.



구현한 모듈이 의도한 대로 정확하게 작동하는지 확인하기 위해, simulation code를 작성했다.

|  |
| --- |
| `timescale 1ns / 1ps  dlatchsim;  reg clk, d;  wire q, qbar;    DLatch dlatch(clk, d, q, qbar);    initial clk=1'b0;  initial d=1'b1;    always clk = #5 ~clk;  always d = #7 ~d;    initial begin  #1600  $finish;  end  endmodule |

이 simulation 또한, clock 값의 타이밍과 d 입력에 따른 출력 값의 변화를 알아보기 위해 입력값들이 서로 엇갈린 시간 간격으로 변화하도록 지연 시간을 설정했다. 위의 simulation의 결과는 다음과 같이 나타났다.



이 때, latch의 특성인 transparency가 나타나는데, 첫 클럭이 high 값을 가지는 부분에서 클럭값이 high인 동안에 생기는 d 값의 모든 변화가 q값에 영향을 끼침을 확인할 수 있다.

1. D Flip-Flop의 결과 및 simulation 과정에 대해서 설명하시오.

D Flip-Flop은 다음에서 볼 수 있듯이 D latch와 진리표 상에서 차이나지 않는다.

|  |  |  |  |
| --- | --- | --- | --- |
| Input | | Output | |
| D | en | Q | ~Q |
| 0 | 0 | N/A | N/A |
| 0 | 1 | 0 | 1 |
| 1 | 0 | N/A | N/A |
| 1 | 1 | 1 | 0 |

다만, D latch와 D Flip-Flop의 차이는 입력 값을 받는 타이밍에서 오는데, D Latch는 enable 신호가 high인 동안의 모든 입력 값의 변동이 출력 값으로 전달되는데 반해, D Flip-Flop은 clock 값이 leading, 또는 trailing edge에 있을 때의 입력 값 만을 출력 값에 반영한다. 따라서, D Flip-Flop을 Verilog code로 구현하기 위해선 D Latch 모듈의 코드에서 실행 체크 조건문 부분을 clock의 leading 또는 trailing edge로 변경해주면 된다. 이를 코드로 구현한 것이 다음 모듈이다.

|  |
| --- |
| `timescale 1ns / 1ps  module DFlipFlop(  clk, d, q, qbar  );  input clk, d;  output q, qbar;  reg q;  assign qbar = ~q;  always @(posedge clk) begin  q <= d;  end  endmodule |

위 모듈에서, always의 조건문 부분에 posedge 키워드를 사용해, clk 값이 leading edge가 될 때 출력인 q에 d값을 전달해준다는 것을 확인할 수 있다. 위 모듈의 회로 다이어그램을 다음과 같이 나타났다.

텍스트, 지도이(가) 표시된 사진

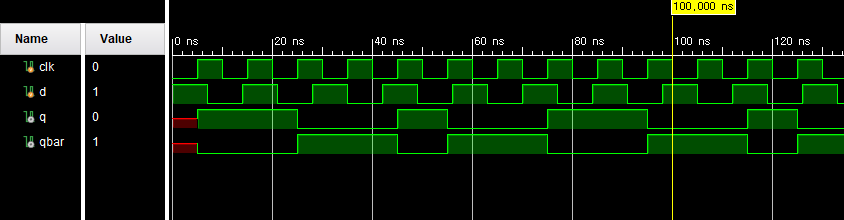
자동 생성된 설명

D Latch를 구현한 모듈의 회로 다이어그램에서 Latch 부분만 D Flip-Flop으로 바뀐 것을 확인할 수 있다.

구현한 모듈이 정확하게 작동하는지 확인하기 위해, clock 신호와 d 입력 신호의 값이 서로 다른 타이밍에 변하도록 설정하는 다음과 같은 simulation code를 작성해 시뮬레이션을 돌렸다.

|  |
| --- |
| `timescale 1ns / 1ps  module dffsim;  reg clk, d;  wire q, qbar;    DFlipFlop dff(clk, d, q, qbar);    initial clk=1'b0;  initial d=1'b1;    always clk = #5 ~clk;  always d = #7 ~d;    initial begin  #1600  $finish;  end  endmodule |

위 코드를 실행한 결과는 아래와 같이 나타났는데, d의 값이 clk 신호의 leading edge에서만 반영이 되어 q로 전달되는 것을 확인할 수 있어, Flip-flop의 opaque한 특성이 드러남을 볼 수 있다.



1. JK Flip-Flop의 결과 및 simulation 과정에 대해서 설명하시오.

JK Flip-Flop은 SR Flip-Flop과 다른 모든 면에서 같지만, J와 K 입력값이 각각 로 들어올 경우, 기존의 SR Flip-Flop에서는 정의되지 않았으나, JK Flip-Flop은 이를 invert 신호로 받아들이고 현재의 값을 invert한 값을 저장한다는 차이가 있다. 따라서, 이 성질을 적용해 표를 그리면 다음과 같다.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Input | | | Output | |
| 순서 | J | K | Q | ~Q |
| 1 | 1 | 0 | 1 | 0 |
| 2 | 0 | 0 | 1 | 0 |
| 3 | 0 | 1 | 0 | 1 |
| 4 | 0 | 0 | 0 | 1 |
| 5 | 0 | 1 | 0 | 1 |
| 6 | 1 | 1 | 1 | 0 |

마지막 6번째 작업에서 입력으로 11이 들어왔을 때 값이 invert된 값으로 바뀌는 부분이 확실히 드러난다.

따라서, 기본적으로 SR Flip-Flop의 구현을 따라가되, 입력이 11일 때에는 현재의 값을 complement해주는 부분을 추가해주면 된다. 이 방식으로 구현한 모듈은 다음과 같다.

|  |
| --- |
| `timescale 1ns / 1ps  module JKFlipFlop(  clk, j, k, q, qbar  );  input clk, j, k;  output q, qbar;  reg q;  assign qbar = ~q;  always @(posedge clk) begin  if((j==1)&&(k==1))  q <= ~q;  else if((j==1)&&(k==0))  q <= 1'b1;  else if((j==0)&&(k==1))  q <= 1'b0;  else  q <= q;  end  endmodule |

부분의 expression에서 q를 반전시켜 구현한 것을 볼 수 있다.

이 코드를 회로 다이어그램으로 나타내면 다음과 같은 그림을 확인할 수 있다.

텍스트, 지도이(가) 표시된 사진

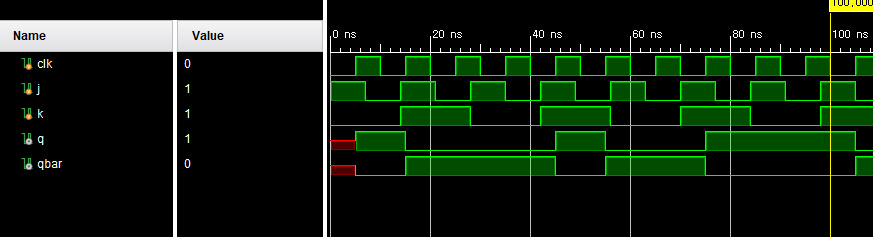
자동 생성된 설명

Q 값이 invert된 상태로 Flip-Flop의 입력에 들어온다는 것을 확인할 수 있다.

구현한 모듈이 정상적으로 작동하는지 확인하기 위해 다음처럼 simulation code를 작성했다. 이 부분은 SR Flip-Flop을 확인할 때 사용한 simulation과 같은 사고과정을 통해 각 입력의 타이밍을 설정했다.

|  |
| --- |
| `timescale 1ns / 1ps  module jkffsim;  reg clk, j, k;  wire q, qbar;    JKFlipFlop jkflipflop(clk, j, k, q, qbar);    initial clk=1'b0;  initial j=1'b1;  initial k=1'b0;    always clk = #5 ~clk;  always j = #7 ~j;  always k = #14 ~k;    initial begin  #1600  $finish;  end  endmodule |

위의 코드를 실행해 각 입력에 대한 결과를 확인해 본 결과는 다음 그래프에서 보이는 것과 같다.



일반적인 SR Flip-Flop과 다르지 않는 결과를 보이나, 입력이 J=1, K=1이 되는 경우에는 don’t care 값을 출력하지 않고, 이 전의 데이터를 반전한 데이터를 저장하게 되는 것을 확인할 수 있다.

1. 결과 검토 및 논의 사항.

* 각 실험에서 latch와 flip-flop의 결과가 같고, 또한 latch에서 볼 수 있는 transparency 특성을 flip-flop에서는 볼 수 없게 된 것으로 보아, 모든 실험이 정확하게 이루어졌고, 구현한 모듈 전부가 올바르게 구현되었다고 볼 수 있다.

1. 추가 이론 조사 및 작성.

* 이번 실험에서는 register 변수를 사용해 이를 구현했으나, Latch와 Flip-Flop은 Verilog상에서도 일반적인 NOR/NAND를 사용한 회로로 표현할 수 있다. Flip flop을 구현할 때에도 구현해 놓은 Latch를 사용해 모듈을 만드는 것이 가능하기에, 이와 같은 다른 방법으로 이 실험을 동일하게 진행할 수 있을 것이다.
* 이 실험에서는 각 Flip-Flop들을 서로 독립적으로 구현했는데, 한 Flip-Flop을 구현하기 위해 다른 종류의 Flip-Flop을 사용할 수도 있다. 예를 들어, 하나의 D Flip-Flop을 구현할 때 SR Flip-Flop의 두 입력을 하나는 inverter를 거치게 해 하나의 입력 D로 받게 하는 방법이 있다. 이는 아래 그림에서 볼 수 있다.

시계이(가) 표시된 사진

자동 생성된 설명