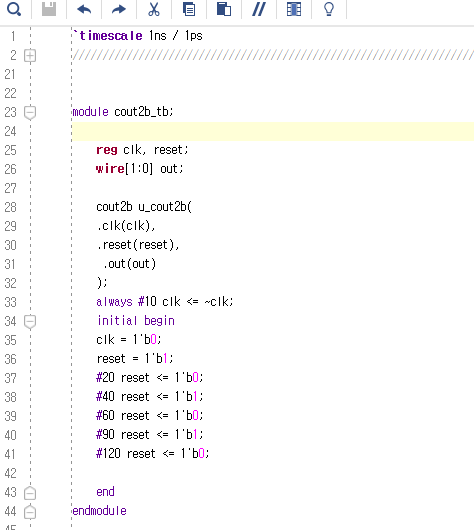
12주차 결과보고서

전공 : 아트앤테크놀로지 학년 : 3학년 학번 : 20191098 이름 : 백승주

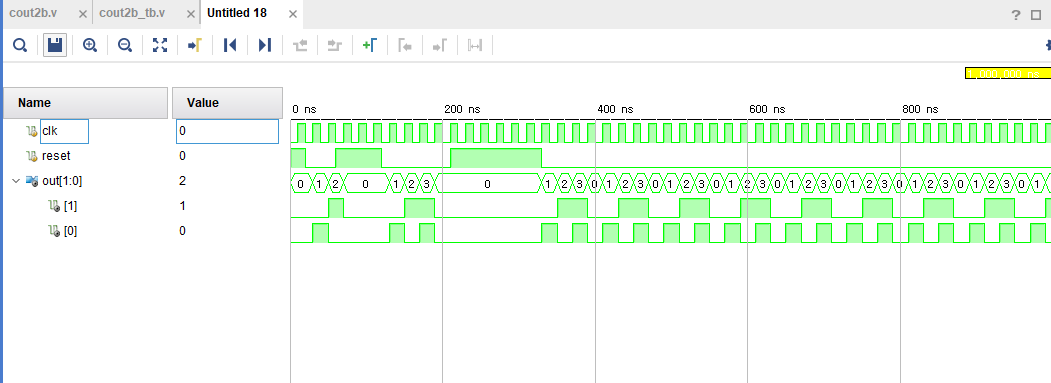
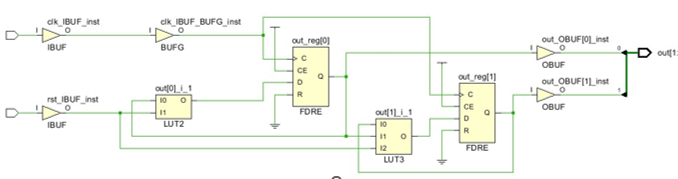
1. 2 bit 이진 카운터는 clock 값을 반영해 이진 숫자 00~11까지 현재 state를 변경해가며 숫자를 셀 수 있는 회로다. 2개의 플립 플롭을 사용해 2비트의 정보를 저장하기 때문에 00~11, 즉 0~3까지의 숫자를 세고 이 범위를 넘으면 숫자가 다시 초기화되어야 한다. 그 verilog 코드는 다음과 같다.



위 코드에서는 배열을 out 변수를 선언해 사용해 2비트의 결과값을 나타내고자 했다. 먼저 out값을 0으로 초기화시킨 후 always(posedge), 즉 클락 값이 positive edge 상태에 이를 때마다 reset 값이 0이 아니라면 out <=out +1; 식을 통해 out값을 1씩 증가시킨다. 이 때 out값은 그 크기가 2비트인 변수이기 때문에 만일 out값이 11의 값을 넘어 증가할 경우 그 값은 carry out 값으로 넘어지만 여기서 따로 carry out값은 설정해두지 않았기 때문에 11의 값을 넘을 경우 out값은 0으로 초기화되게 된다.   
 또한 만일 if(reset==1), 즉 reset값이 1의 값을 가질 경우 out값은 0으로 초기화되도록 코드를 구성했다.

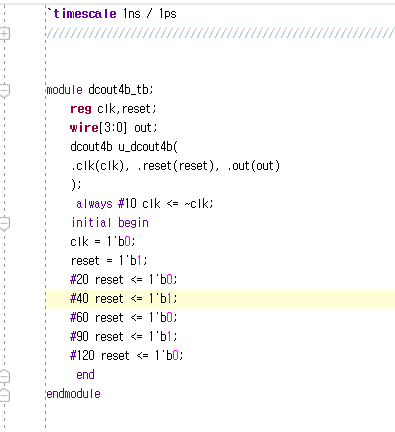
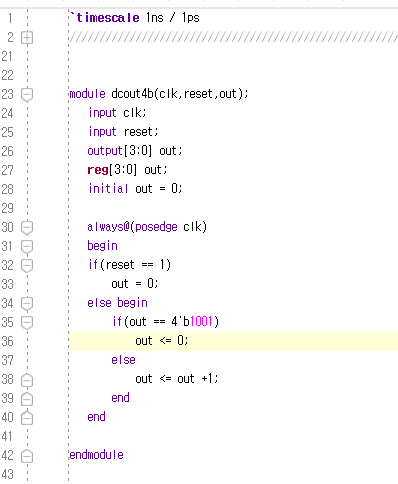


위 코드는 앞선 count2비트의 testbench 코드로 clk값은 10ns을 주기로 값이 변하게 된다. 따라서 out 값 역시 10ns마다 그 값이 증가하게 될 것이다. 또한 일정 시간동안 reset 값을 1의 값을 가지게 만들어 초기화된 후 변화를 관찰하고자 했다. 이에 따른 schematic diagram과 시뮬레이션 결과는 다음과 같다.

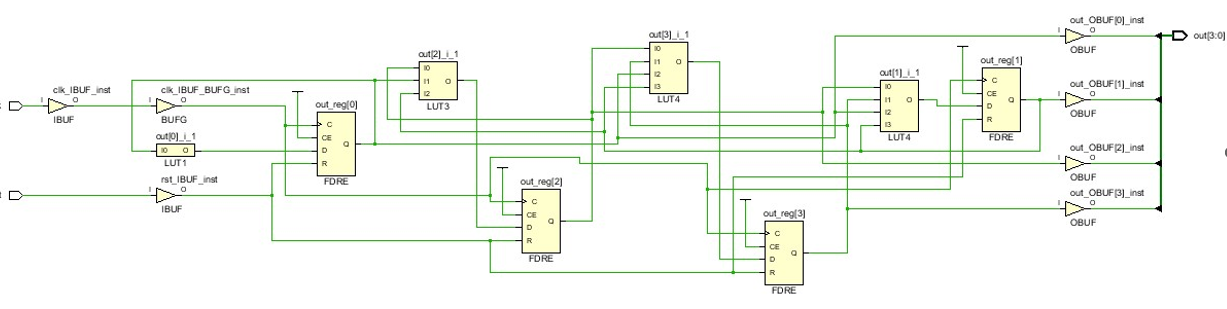


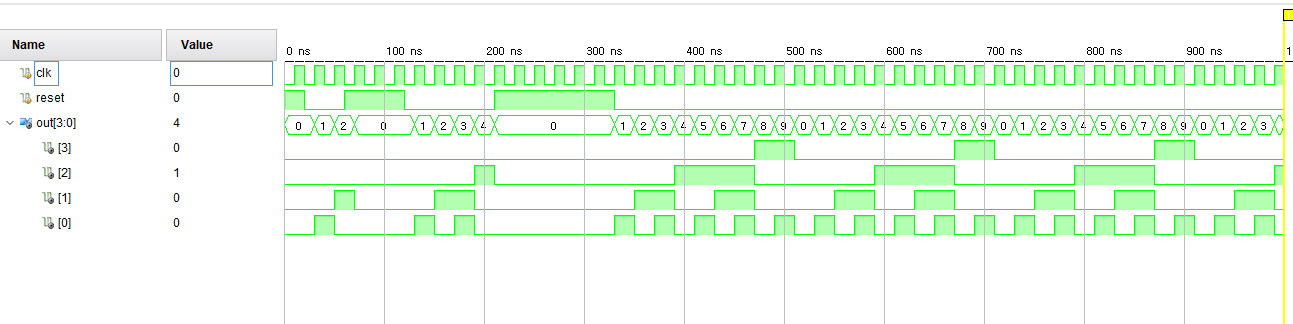
위 시뮬레이션 결과를 살펴보면 먼저 reset값이 0일 때는 그 이전 out 값과는 상관없이 항상 out 값이 0으로 초기화된 것을 관찰할 수 있다. 그리고 reset 값이 0일 때 clk 값이 positive edge 값을 가질 때마다 out 값이 변화하는 것을 관찰할 수 있는데 00, 01, 10, 11 의 값들, 다시 말해 0,1,2,3의 값들을 가지며 계속 그 값이 1씩 증가하는 것을 관찰할 수 있다. 또한 이후 ou값이 11의 값을 초과할 경우에는 그 값이 0으로 다시 초기화된 후 다시 증가를 반복하는 것을 관찰할 수 있었다.

1. 4 비트 decade counter는 십진수 형태의 수를 세는 카운터로 0~9가지의 숫자를 카운트한다. 0~9, 즉 10개의 숫자를 세야 하기 때문에 최소 4개의 플립 플롭이 필요하게 된다. 그 Verilog 코드는 다음과 같다.



앞선 2bit counter와 비슷하게 동작하기 때문에 비슷한 형태의 코드가 구성되었다. output[3:0] out; 을 통해 4비트 크기의 정보를 저장할 수 있는 out 배열을 선언했다. 이후 reset == 1 일 때는 out값이 0으로 초기화되도록 했고 out값이 카운터의 범위인 9를 넘겨 10이 될 경우, 즉 1001의 값이 될 경우 초기화 되도록 했다. 또한 2비트 이진 카운터와 마찬가지로 클락 값이 positive edge 일 때마다 그 값을 1씩 증가시키도록 했다. 그에 따른 schematic diagram과 시뮬레이션 결과는 다음과 같다.

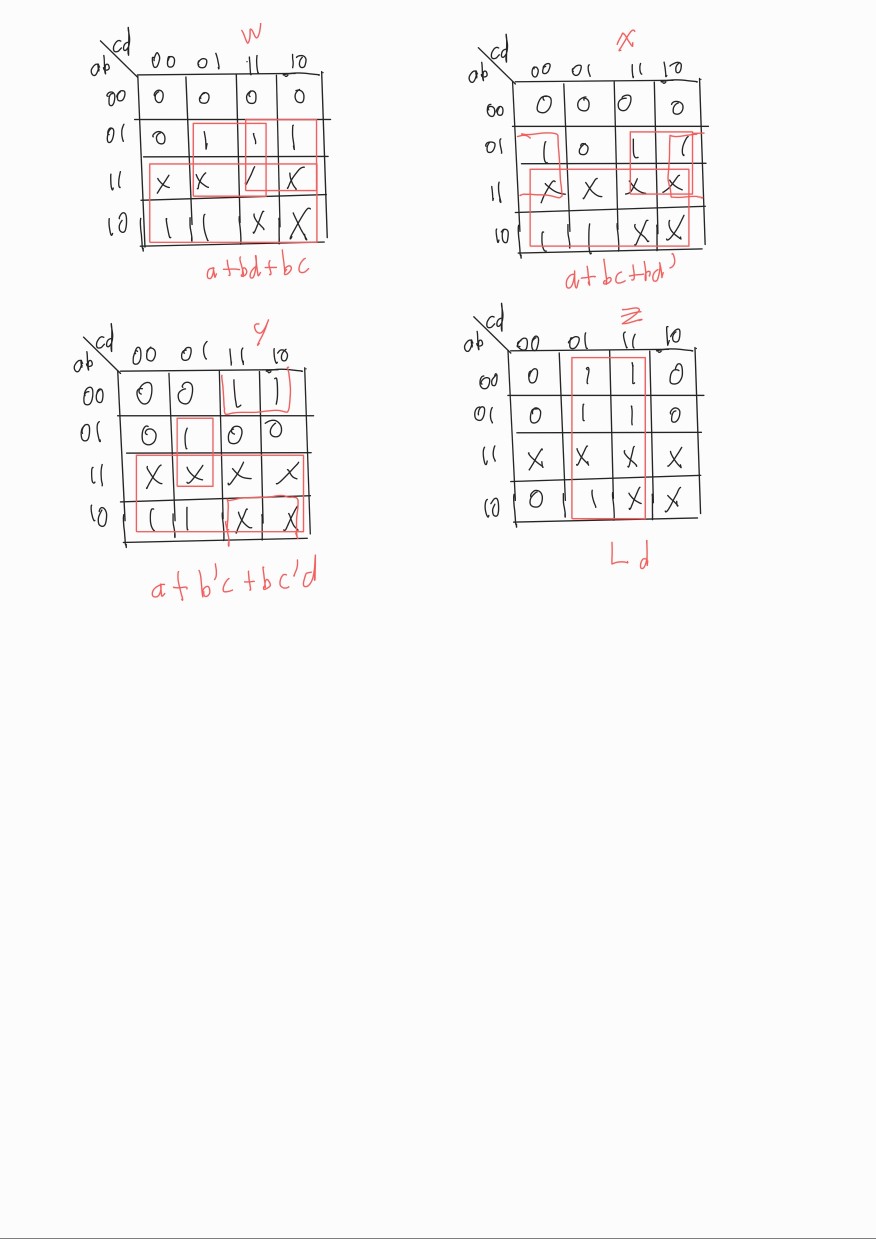




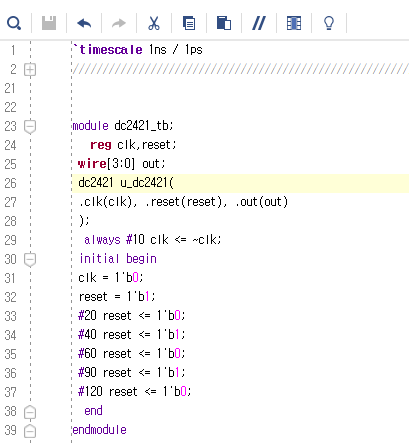
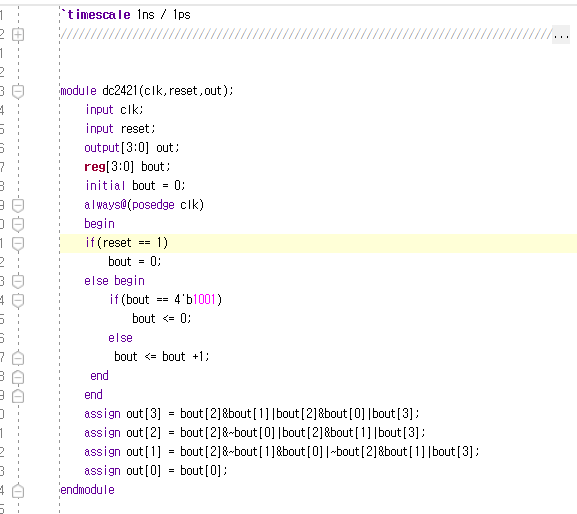
위 시뮬레이션 결과를 살펴보면 먼저 reset 값이 0인 구간에서는 4개의 out 값 배열 모두 0의 값을 가지며 0을 나타내고 있는 것을 확인할 수 있다. 이후 clk 값이 positive edge 일 때마다 out 배열의 out 값이 1씩 증가해 그 값들이 0001,0010, 0011, 0100, 0101,0110, 0111, 1000, 1001 까지 계속 증가하다가 이 1001 값 이후에는 다시 0000의 값을 가지고 반복해서 1씩 증가하는 모습을 확인할 수 있었다.

1. 4비트 2421 decade counter는 앞선 4 비트 decade counter 처럼 0~9까지의 숫자들을 카운트한다는 점에서 그 형태가 유사하지만 앞선 decade counter는 숫자를 세며 그 결과를 8421 방식에 따라 출력했던 것과 달리 이번 counter는 그 결과들을 2421 형태에 따라 나타낸다. 즉, 앞선 4 비트 decade counter의 결과값들을 2421 decade counter 에 따라 변환해야 한다. 이를 위해서 앞선 6주차 실험에서 다뤘던 8421-2421 code converter의 내용을 참고할 수 있다.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Decimal | 8(a) | 4(b) | 2(c) | 1(d) | 2(w) | 4(x) | 2(y) | 1(z) |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| 2 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| 3 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| 4 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| 5 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 |
| 6 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 0 |
| 7 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 |
| 8 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 |
| 9 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |

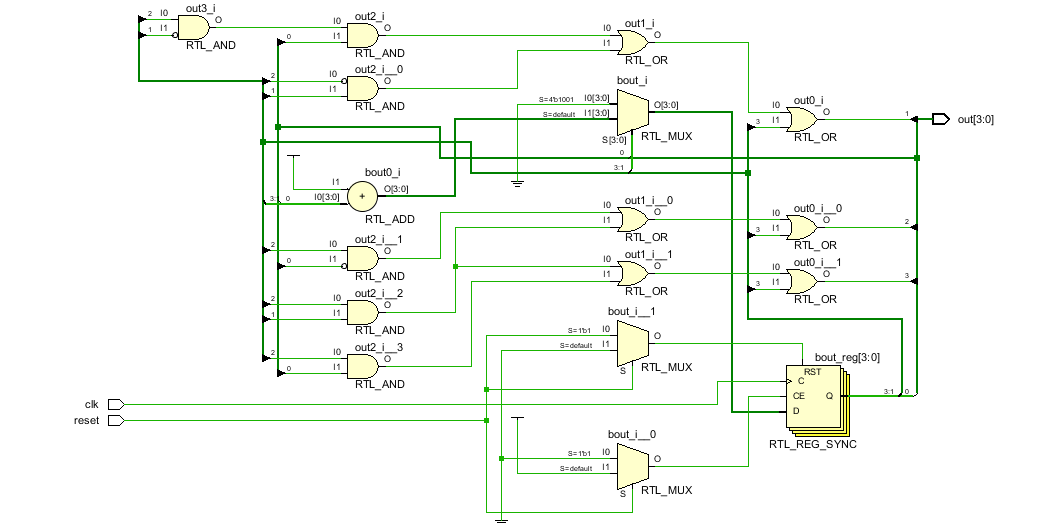


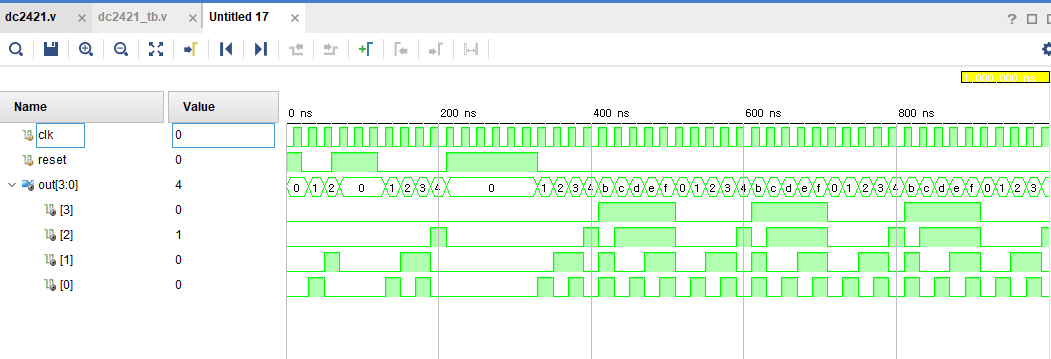
위 내용들을 토대로 Verilog 코드를 세우면 다음과 같다.



먼저 앞선 4비트 decade counter와 비슷하게 코드를 짜 변수 bout에 카운트 값에 앞선4비트 decade counter의 결과값들을 설정했다. 그리고 이러한 4 비트 decade counter의 값들을 2421 형식으로 변환하기 위해 6주차 실습에서 8421-2421 converter를 구성하기 위해 만든 카르노 맵과 식을 이용했다. Bout 배열 4개의 값들을

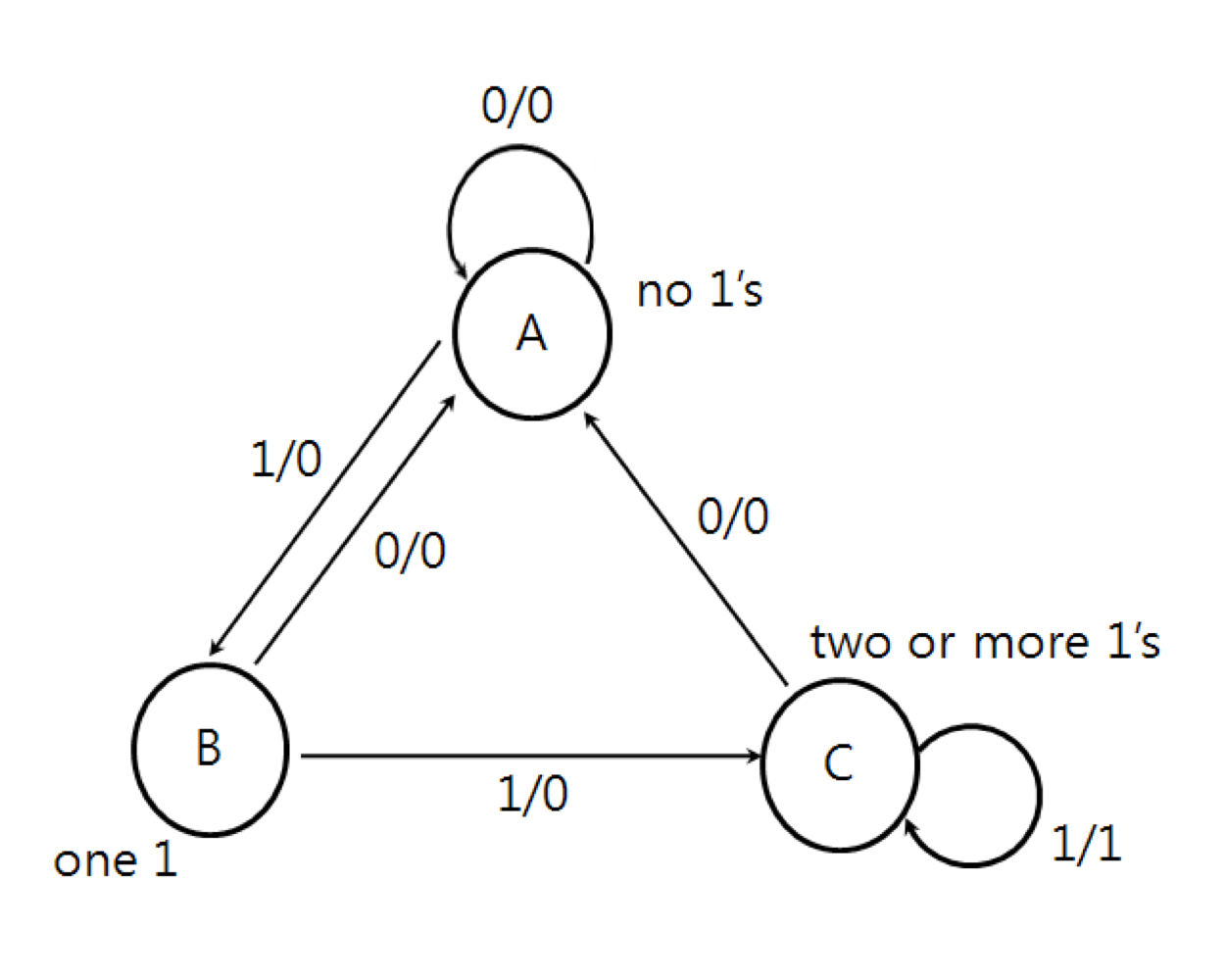
위 Verilog 코드에 따른 schematic과 시뮬레이션 결과는 다음과 같다.



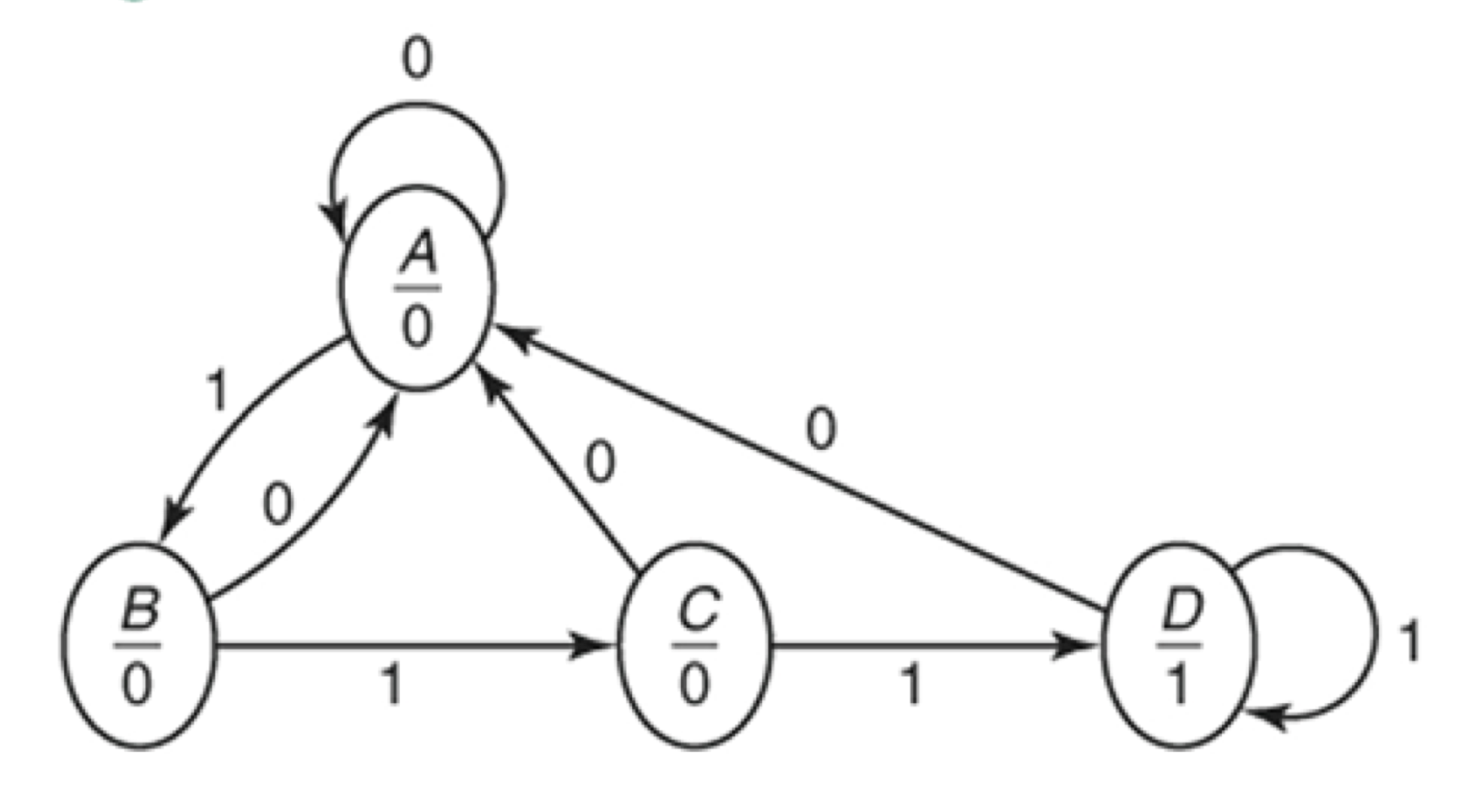


위 시뮬레이션 결과를 살펴보면 앞선 counter들과 마찬가지로 reset 값이 0일 때는 그 결과값들이 0으로 초기화되는 것을 확인할 수 있다. 그리고 clk값이 positive edge일 때마다 out 값들은 0000,0001,0010,0100 의 값들을 가지며 앞선 decade counter와 똑같이 1씩 증가하다가 그 다음 값부터는 0101이 아닌 1011의 값을 가지고 그 이후 1100, 1101, 1110, 1111의 값들을 가지게 된다. 이 값들은 b,c,d,e,f 로 나타났지만 차례대로 5,6,7,8,9 를 의미한다. 이러한 값들의 증가가 0~9까지 positive edge 일 때마다 반복되는 것을 확인할 수 있었다.

1. 이번 회로 구성은 이전과 달리 if else 구문을 통해 회로를 작성했다. 기존까지는 대부분 진리표나 카르노 맵을 통해 논리식을 세워 그 회로를 구성하는 코드를 작성했지만 이번에는 if else문을 통해 카운터의 회로를 바로 구성할 수 있었다. if문을 통해 카운터의 범위를 벗어나거나 reset이 1이 되는 경우 그 상태를 0으로 초기화했으며 reset이 1이 아니고 카운터의 범위를 벗어나지 않았다면 positive edge일 때마다 카운트 값을 1씩 증가시킬 수 있었다.
2. 순차 논리 회로에는 mealy machine과 moore machine이 있다.



먼저 Mealy machine은 출력이 현재 상태와 입력에 의존하는 FSM이다. 따라서, Mealy Machine의 출력은 즉시 또는 동기적으로 변경될 수 있다. 이러한 Mealy machine은 일반적으로 더 적은 수의 상태로 시스템을 표현할 수 있으므로, 설계가 더 효율적일 수 있지만 시스템의 출력이 입력에 직접적으로 연결되어 있기 때문에, 입력의 변화에 따라 출력에 직접적인 영향을 미칠 수 있다.



Moore Machine은 출력이 현재 상태에만 의존하는 FSM으로 Mealy Machine과는 반대로, 입력의 직접적인 변화가 출력에 영향을 미치지 않는다. 따라서 Moore Machine은 mealy machine과 달리 입력에 대해 더 안정적일 수 있다. 하지만 이 러한 Moore machine은모델은 일반적으로 더 많은 상태를 필요로 하고출력이 바뀌는데 시간 지연이 생길 수 있게 된다.