Verilog-HDL 기반
4bit binary Adder 설계

Shinwoong Kim

Hardware Description Language (HDL)

- **Manual method for designing logic circuits are feasible only when the circuit is small**
- **Practically, designers use computer-based design tools**
  - ✓ All modern design tools rely on a hardware description language(HDL) to describe the circuit
  - ✓ It is like C, C++, Java or Python (programming language)
- **The most widely used language is *Verilog* and *VHDL***
  - ✓ They have many similarities
  - ✓ Verilog-HDL is the more common in industry
    - It is an easier than VHDL to describe, learn and use

Half-Adder

```
module Half_Adder ( x, y, C, S );
```
*port list

```
input x, y;
output C, S;
```
*port declaration

*Body

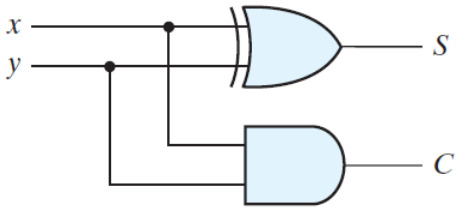
```
    //Continuous assignment type
    assign S = x^y; // XOR
    assign C = x&y; // AND
```
*Comment

```
endmodule
```

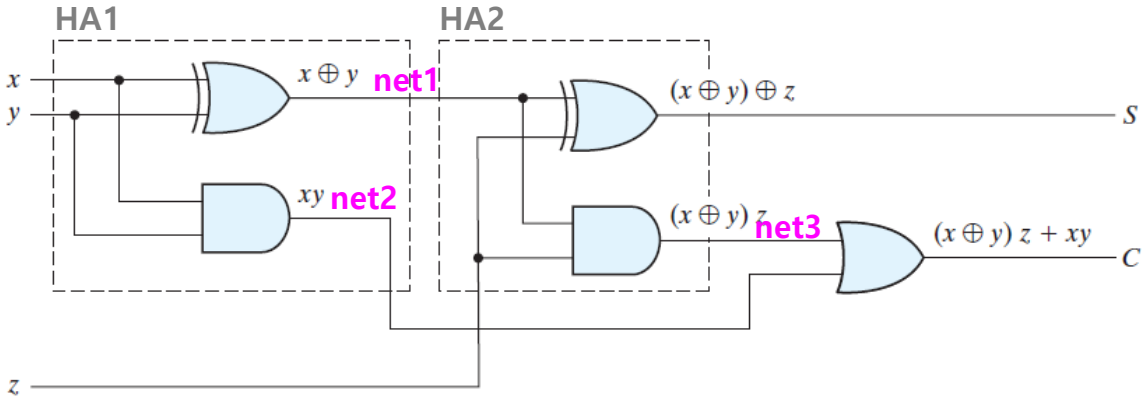
Half Adder

x	y	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

```
module Half_Adder ( x, y, C, S );

input x, y;
output C, S;
reg C, S;

    //Procedure assignment type
    always@(x or y) begin // same as "always@(*) begin"
        S = x^y;
        C = x&y;
    end

endmodule
```



Full-Adder

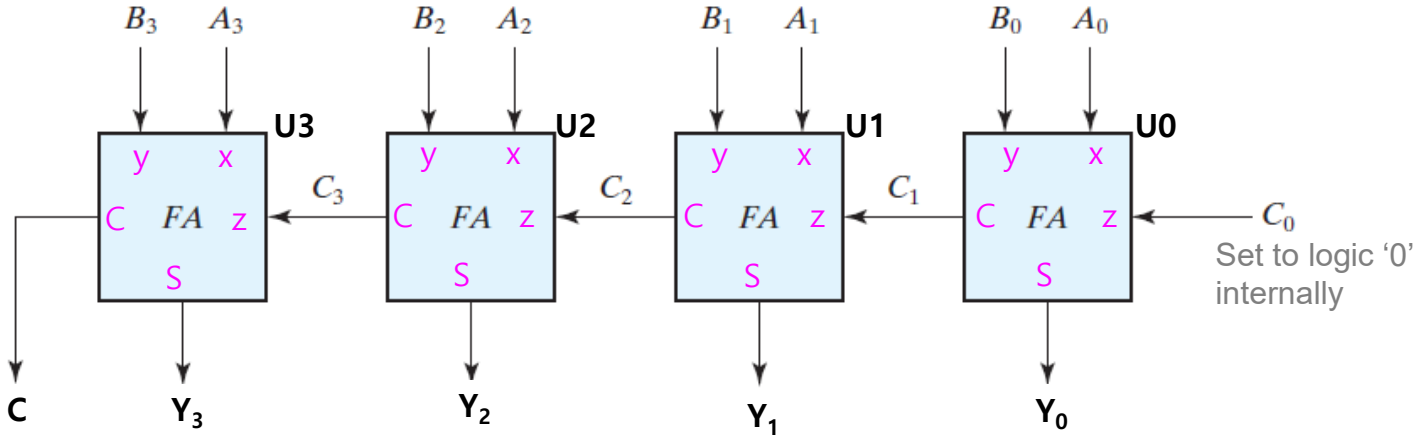


Full Adder

x	y	z	C	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

```
module Full_Adder ( x, y, z, C, S );

input x, y, z;
output C, S;

wire net1, net2, net3;

    //Continuous assignment type
    //for HA1
    assign net1 = x^y;  // XOR
    assign net2 = x&y;  // AND

    //for HA2
    assign S = net1^z;      // XOR
    assign net3 = net1&z;   // AND

    //for C
    assign C = net3 | net2; // OR

endmodule
```

Binary 4-bit Adder



*Module Instantiation (using *Full_Adder* module)

```
module Binary_4bit_Adder ( A, B, C, Y );

input [3:0] A, B;
output C;
output [3:0] Y;

wire C1, C2, C3;

Full_Adder U0( .x(A[0]), .y(B[0]), .z(1'b0), .C(C1), .S(Y[0]) );
Full_Adder U1( .x(A[1]), .y(B[1]), .z(C1),   .C(C2), .S(Y[1]) );
Full_Adder U2( .x(A[2]), .y(B[2]), .z(C2),   .C(C3), .S(Y[2]) );
Full_Adder U3( .x(A[3]), .y(B[3]), .z(C3),   .C(C), .S(Y[3]) );

endmodule
```

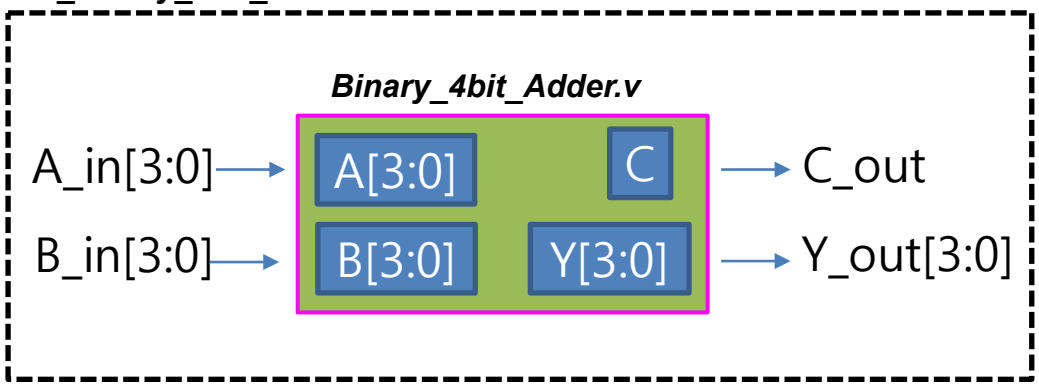
Simulation: Binary 4-bit Adder

- **Usually make 'test-bench' file first**
  *Environment to verify the design source
  - ✓ Including an input condition

```
module tb_Binary_4bit_Adder();

reg [3:0] A_in, B_in;
wire C_out;
wire [3:0] Y_out;

Binary_4bit_Adder DUT(
    .A(A_in),
    .B(B_in),
    .C(C_out),
    .Y(Y_out)
);

initial begin
    A_in = 4'b0011;
    B_in = 4'b0110;
end

endmodule
```

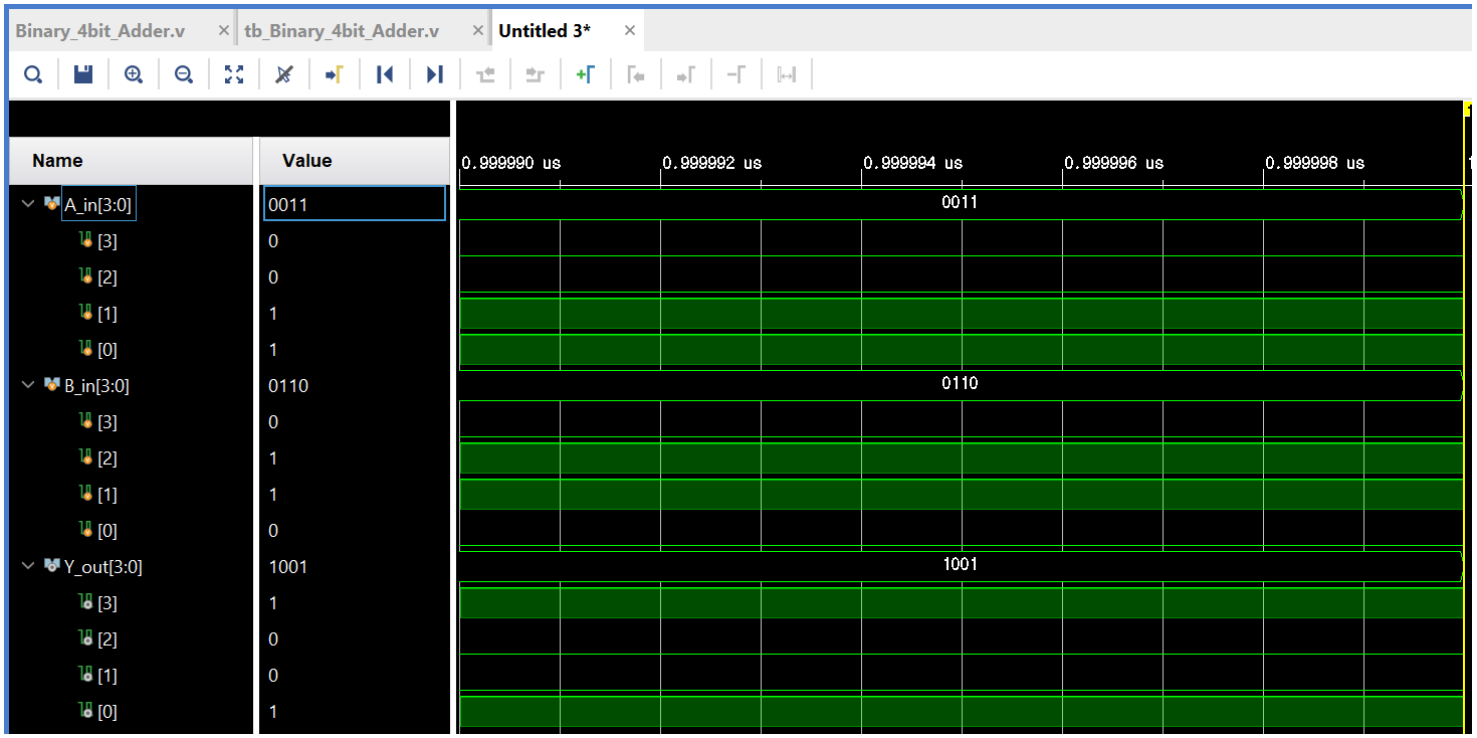
```
module Binary_4bit_Adder ( A, B, C, Y );

input [3:0] A, B;
output C;
output [3:0] Y;

wire C1, C2, C3;

Full_Adder U0( .x(A[0]), .y(B[0]), .z(1'b0), .C(C1), .S(Y[0]) );
Full_Adder U1( .x(A[1]), .y(B[1]), .z(C1),   .C(C2), .S(Y[1]) );
Full_Adder U2( .x(A[2]), .y(B[2]), .z(C2),   .C(C3), .S(Y[2]) );
Full_Adder U3( .x(A[3]), .y(B[3]), .z(C3),   .C(C), .S(Y[3]) );

endmodule
```

tb_Binary_4bit_Adder.v



Simulation: Binary 4-bit Adder

- **Simulation is also performed based on the computer-based tools**
  - ✓ 1) Source code design
  - ✓ 2) Simulate it to verify the logic function



실험

- **Xilinx Vivado 프로그램에서 4bit binary adder 설계**
  - ✓ Full_Adder.v 작성
  - ✓ Binary_4bit_Adder.v 작성
- **Xilinx Vivado 프로그램에서 Simulation 수행**
  - ✓ 테스트 벤치 파일: tb_Binary_4bit_Adder.v 작성
    - 입력 A[3:0]: 4'b0011;
    - 입력 B[3:0]: 4'b0110;
  - ✓ 시뮬레이션 수행 → 조교 검사 후 퇴실 가능
- **[참고]**
  - ✓ 다음 주 실험은 금일 설계한 4bit adder + 7 segment display decoder를 포함하여 실제 FPGA 보드에 실행해보는 작업 수행 예정
    - 따라서, 설계 data를 잘 간직하기 바람