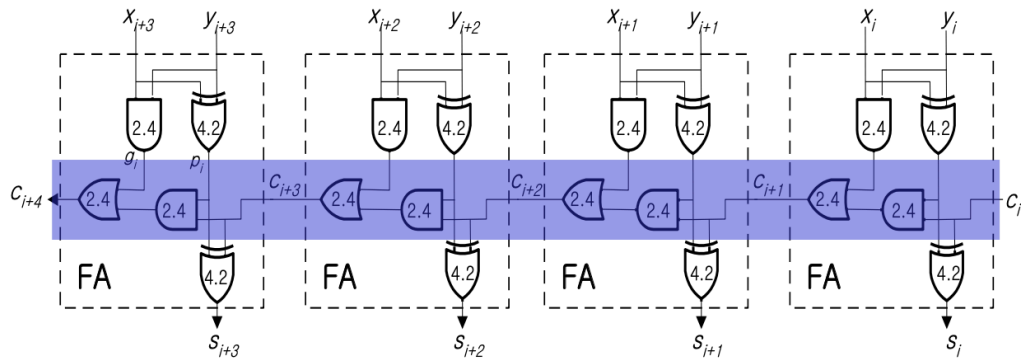


1. 4bit Binary Parallel Adder의 결과 및 Simulation 과정에 대해서 설명하시오.

(verilog source, 출력 예시, 과정 상세히 적을 것)



4bit Binary Parallel Adder는 위와 같이 1bit full adder를 일렬로 4개 연결하여 구현한다. 각 자리의 수를 더하고, (carry도 포함), 합을 반환하여 다음 전가산기에 새로운 carry를 넘겨주는 구조이다. (앞서 10주차 예비보고서에서 서술) 1bit full adder를 module로 구현한 뒤 이를 여러 번 호출하여 4bit Binary Parallel Adder를 호출하였다. 다음은 4bit Binary Parallel Adder의 Verilog 코드이다.

```

1  `timescale 1ns / 1ps
2
3  module adder1bit(A, B, Ci, S, Co);
4
5  input A, B, Ci;
6  output S, Co;
7  assign S=A^B^Ci;
8  assign Co=(A&B)|((A^B)&Ci);
9
10 endmodule
11
12 module adder4bit(A, B, Ci, S, Co);
13 input [3:0] A, B; input Ci;
14 output [3:0] S; output Co;
15
16 wire [3:0] A, B, S; wire Ci, Co;
17 wire [2:0] C;
18
19 adder1bit add1(A[0], B[0], Ci, S[0], C[0]);
20 adder1bit add2(A[1], B[1], C[0], S[1], C[1]);
21 adder1bit add3(A[2], B[2], C[1], S[2], C[2]);
22 adder1bit add4(A[3], B[3], C[2], S[3], Co);
23
24 endmodule
25
26 module adderInterface(A, B, Ci, S, Co);
27
28 input [3:0] A, B; input Ci;
29 output [3:0] S; output Co;
30 wire [3:0] A, B, S;
31 wire Ci, Co;
32 adder4bit(A, B, Ci, S, Co);
33
34 endmodule

```

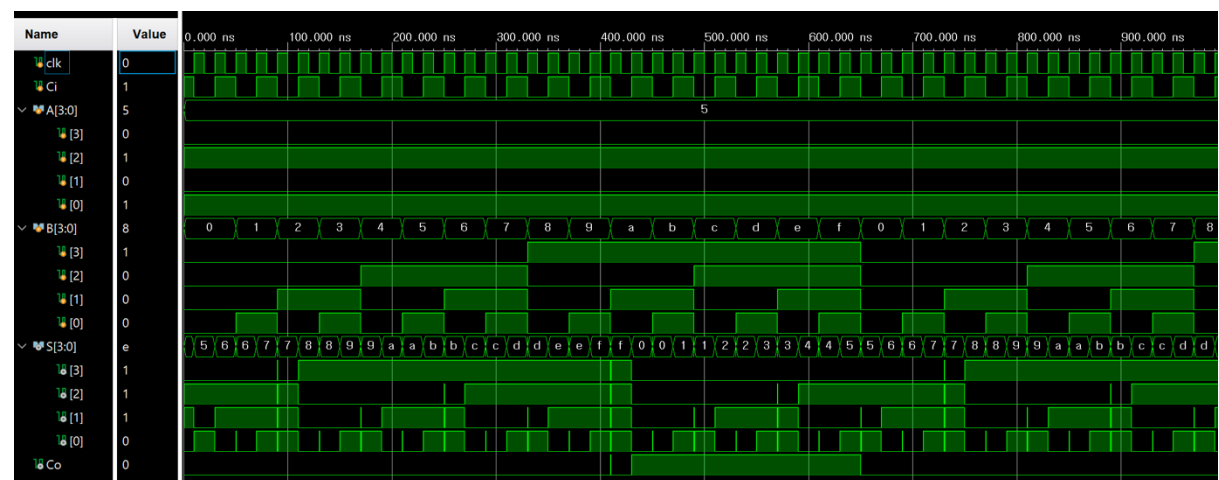
위와 같은 구조로 작성한 Verilog 코드를 다음과 같은 테스트벤치 코드로 시뮬레이션을 돌려 결과를 확인하였다.

```

1  `timescale 1ns / 1ps
2
3  module adder4bit_tb();
4
5  reg clk, Ci; reg [3:0] A, B;
6  wire [3:0] S; wire Co;
7
8  adder4bit connect(A, B, Ci, S, Co);
9
10 initial begin
11     clk=0; A[3]=0; A[2]=1; A[1]=0; A[0]=1;
12     B[3]=0; B[2]=0; B[1]=0; B[0]=0; Ci=1;
13     end
14
15 always clk = #10 ~clk;
16 always @(posedge clk) begin
17     Ci <= ~Ci; B[0] <= #39.999 ~B[0];
18     B[1] <= #79.999 ~B[1]; B[2] <= #159.999 ~B[2];
19     B[3] <= #319.999 ~B[3];
20     end
21
22 endmodule

```

시뮬레이션 결과는 다음과 같다.



일례로 Ci=1, A=0101, B=0000일 때 결과 S값이 0110이 나와야 하는데, 위 시뮬레이션 결과에서 동일하게 나옴을 확인할 수 있다.

FPGA를 다음과 같은 과정으로 작성한 코드를 실행해 보았다.

Design Source, Simulation Source, Constraints file을 모두 작성한 후 Run Synthesis, Run Implementation을 진행한다.

▼ SYNTHESIS

▶ Run Synthesis

▼ IMPLEMENTATION

▶ Run Implementation

Open Implemented Design -> Constraints Wizard -> Define Target -> check으로 target을 설정한다.


> Open Implemented Design

Generate Bitstream -> Open Hardware Manager -> Open Target -> Auto Connect -> Program Device
로 FPGA에 업로드를 진행한다.

 Generate Bitstream

▼ Open Hardware Manager

Open Target

 Auto Connect

Program Device

위와 같은 FPGA 실행 과정은 4bit Binary Parallel Subtractor, BCD Adder 실습에서 동일하므로 반복
서술하지 않는다.

2. 4bit Binary Parallel Subtractor의 결과 및 Simulation 과정에 대해서 설명하시오.

(verilog source, 출력 예시, 과정 상세히 적을 것)

4bit Binary Parallel Subtractor는 기본적으로 Binary Parallel Adder와 구조가 비슷하다. 1bit full
subtractor에는 B(Borrow)와 D값을 입-출력으로 가지고, 4개를 일렬로 연결하여 구현한다. d와
bout(bo)를 Verilog 코드로 나타내면 다음과 같다.

```
assign d = A^B^bi;
```

```
assign bo = ((~A)&bi)|((~A)&B)|(B&bi);
```

➔ 최종 4bit Binary Parallel Subtractor의 Verilog 코드는 아래와 같다.

(다음쪽에)

```

1  `timescale 1ns / 1ps
2
3  module sub1bit(A, B, bi, d, bo);
4
5  input A, B, bi;
6  output d, bo;
7  assign d = A^B^bi;
8  assign bo = ((~A)&bi)|((~A)&B)|(B&bi);
9
10 endmodule
11
12 module sub4bit(A, B, bi, d, bo);
13
14 input [3:0] A, B; input bi;
15 output [3:0] d; output bo;
16
17 wire [3:0] A, B, d; wire bi, bo;
18 wire [2:0] bout;
19
20 sub1bit sub1(A[0], B[0], bi, d[0], bout[0]);
21 sub1bit sub2(A[1], B[1], bout[0], d[1], bout[1]);
22 sub1bit sub3(A[2], B[2], bout[1], d[2], bout[2]);
23 sub1bit sub4(A[3], B[3], bout[2], d[3], bo);
24
25 endmodule
26
27 module sub4bitInterface(A, B, bi, d, bo);
28
29 input [3:0] A, B; input bi;
30 output [3:0] d; output bo;
31 wire [3:0] A, B, d;
32 wire bi, bo;
33
34 sub4bit(A, B, bi, d, bo);
35
36 endmodule

```

위와 같은 구조로 작성한 Verilog 코드를 다음과 같은 테스트벤치 코드로 시뮬레이션을 돌려 결과를 확인하였다.

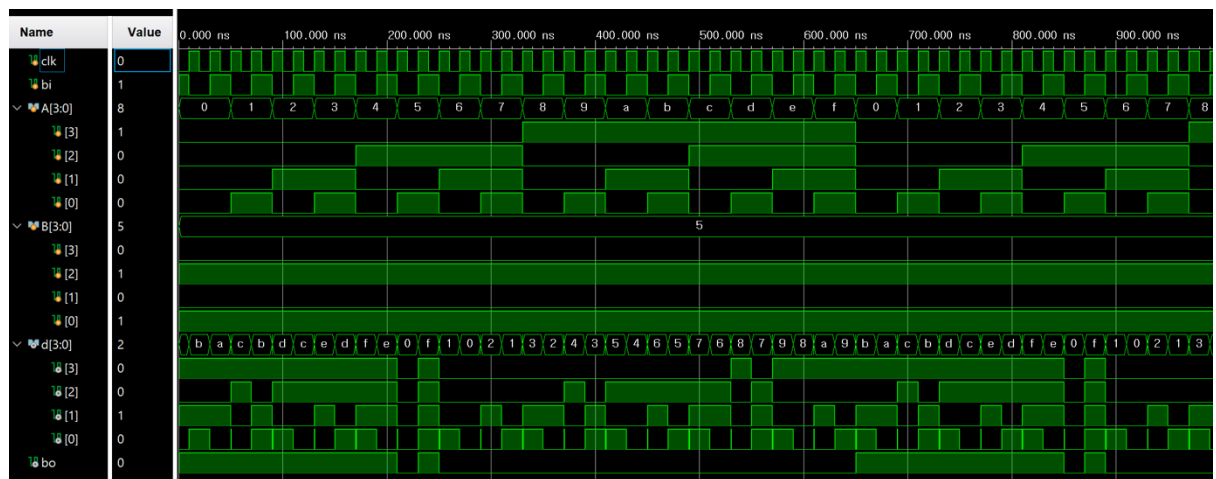
(다음쪽에)

```

1 `timescale 1ns / 1ps
2
3 module sub4bit_tb();
4
5 reg clk, bi; reg [3:0] A, B;
6 wire [3:0] d; wire bo;
7
8 sub4bit connect(A, B, bi, d, bo);
9
10 initial begin
11 clk=0; B[0]=1; B[1]=0; B[2]=1; B[3]=0;
12 bi=1; A[3]=0; A[2]=0; A[1]=0; A[0]=0;
13 end
14
15 always clk = #10 ~clk;
16 always @(posedge clk) begin
17 bi <= ~bi;
18 A[0] <= #39.999 ~A[0];
19 A[1] <= #79.999 ~A[1];
20 A[2] <= #159.999 ~A[2];
21 A[3] <= #319.999 ~A[3];
22
23 end
24 endmodule

```

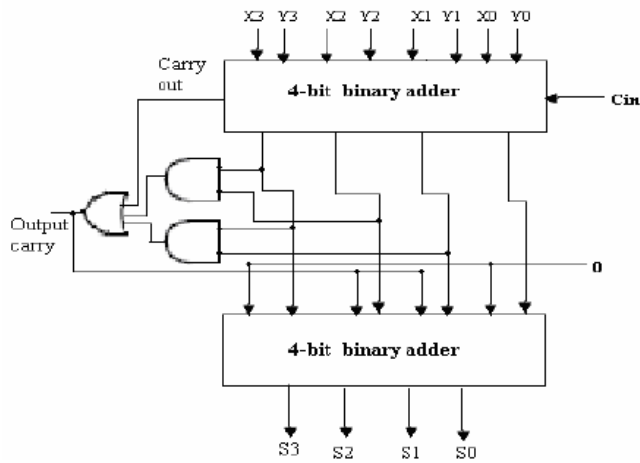
시뮬레이션 결과는 다음과 같다.



일례로 bi=0, A=1111, B=0101일 때 결과 d값이 1010이 나와야 하는데, 위 시뮬레이션 결과에서 동일하게 나옴을 확인할 수 있다.

3. BCD Adder의 결과 및 Simulation 과정에 대해서 설명하시오.

(verilog source, 출력 예시, 과정 상세히 적을 것)



BCD Adder의 개괄적인 회로도 는 위와 같다.

BCD 형태의 수로 덧셈을 하기 위해서는, 다음과 같은 점을 지켜야 한다.

- 한 자리 수를 더하면 최대 18을 넘을 수 없다.
- 0 ~ 9는 그대로 나타낸다.
- 10 이상의 결과는 둘로 쪼개어 나타낸다.
- 10 이상일 때는 1을 앞 자리로 빼서 나타내고 원래 결과에서 10 뺀 값을 그 자리에 나타낸다.

다음은 BCD Adder 의 Verilog 코드이다.

4bit Binary Parallel Adder, 4bit Binary Parallel Subtractor 와 같이 1bit adder module 을 구현한 뒤 연결하였다.

adder1bit module 은 4bit Binary Parallel Adder 에서 썼던 것과 동일하다.

BCD module 에서

adder4bit add1(A, B, 0, A2, Cout);

add1 module 을 호출하고,

assign Cout2 = (Cout)|(A2[3]&A2[2])|(A2[3]&A2[1]);

Cout 변수에 좌측과 같은 계산값을 넣어준다.

always @(Cout2)

if(Cout2 == 1'b1) begin

if 문을 통해 10 이상 유무를 체크하여

B2 = 4'b0110;

그에 맞는 연산을 진행한다.

end

else begin

B2 = 4'b0000;

end

adder4bit add2(A2, B2, 0, S, Cout3);

if-else 문을 통해 B2 값을 정리하였다면

adder4bit module 을 호출한다.

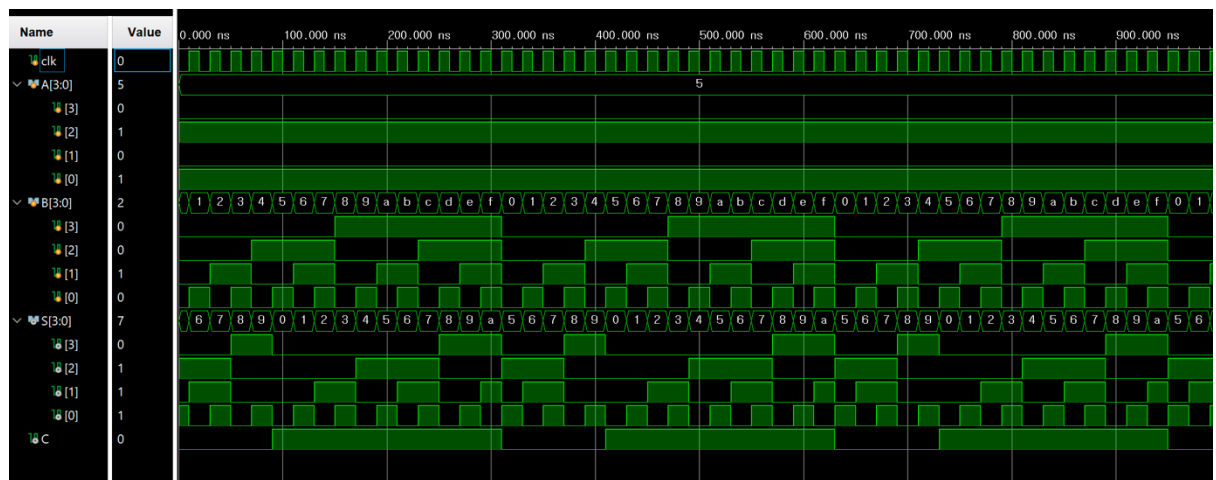
다음은 BCD Adder 의 Verilog 코드이다.

```
1  `timescale 1ns / 1ps
2
3  module adder1bit(A, B, Ci, S, Co);
4
5  input A, B, Ci;
6  output S, Co;
7  assign S=A^B^Ci;
8  assign Co=(A&B)|((A^B)&Ci);
9
10 endmodule
11
12 module adder4bit(A, B, Ci, S, Co);
13 input [3:0] A, B; input Ci;
14 output [3:0] S; output Co;
15
16 wire [3:0] A, B, S; wire Ci, Co;
17 wire [2:0] C;
18
19 adder1bit add1(A[0], B[0], Ci, S[0], C[0]);
20 adder1bit add2(A[1], B[1], C[0], S[1], C[1]);
21 adder1bit add3(A[2], B[2], C[1], S[2], C[2]);
22 adder1bit add4(A[3], B[3], C[2], S[3], Co);
23
24 endmodule
25
26 module BCD(A, B, S, C);
27
28 input [3:0] A, B;
29 output [3:0] S; output C;
30
31 wire [3:0] A2; wire Cout;
32 reg [3:0] B2; wire Cout2, Cout3;
33
34 adder4bit add1(A, B, 0, A2, Cout);
35
36 assign Cout2 = (Cout)|(A2[3]&A2[2])|(A2[3]&A2[1]);
37
38 always @(Cout2)
39 if(Cout2 == 1'b1) begin
40     B2 = 4'b0110;
41 end
42 else begin
43     B2 = 4'b0000;
44 end
45
46 adder4bit add2(A2, B2, 0, S, Cout3);
47 assign C = Cout2|Cout3;
48
49 endmodule
50
51 module BCDinterface(A, B, C, S);
52 input [3:0] A, B; output C;
53 output [3:0] S;
54 BCD bcdcode(A, B, S, C);
55 endmodule
```

위와 같은 구조로 작성한 Verilog 코드를 다음과 같은 테스트벤치 코드로 시뮬레이션을 돌려 결과를 확인하였다.

```
1 `timescale 1ns / 1ps
2
3 module BCD_tb();
4 reg clk; reg [3:0] A, B;
5 wire [3:0] S; wire C;
6 BCD connect(A, B, S, C);
7
8 initial begin
9 clk = 0; A=4'd5;
10 B=4'd0;
11 end;
12
13 always clk = #10 ~clk;
14 always @(posedge clk) begin
15 B <= B+4'd1;
16 end
17 endmodule
```

시뮬레이션 결과는 다음과 같다.



일례로 A=0101, B=1101일 때 결과 S값 1000, C=1이 나와야 하는데,

$(0101_{(2)} + 1101_{(2)} = 10010_{(2)} = 18 = 10 + 8)$ 위 시뮬레이션 결과에서 동일하게 나옴을 확인할 수 있다.