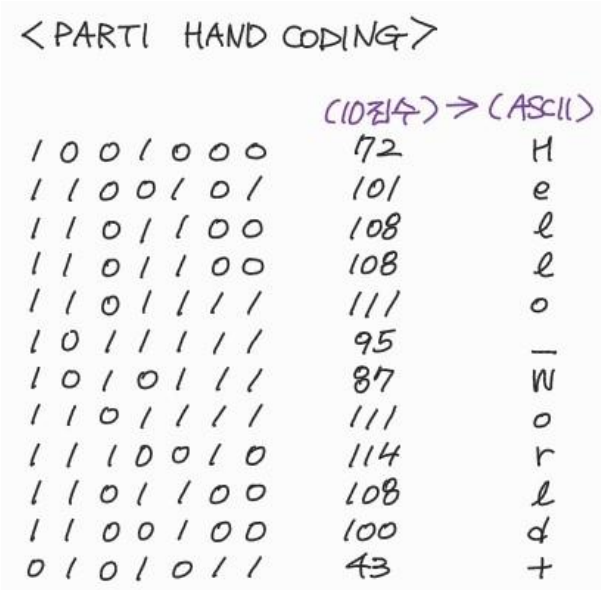
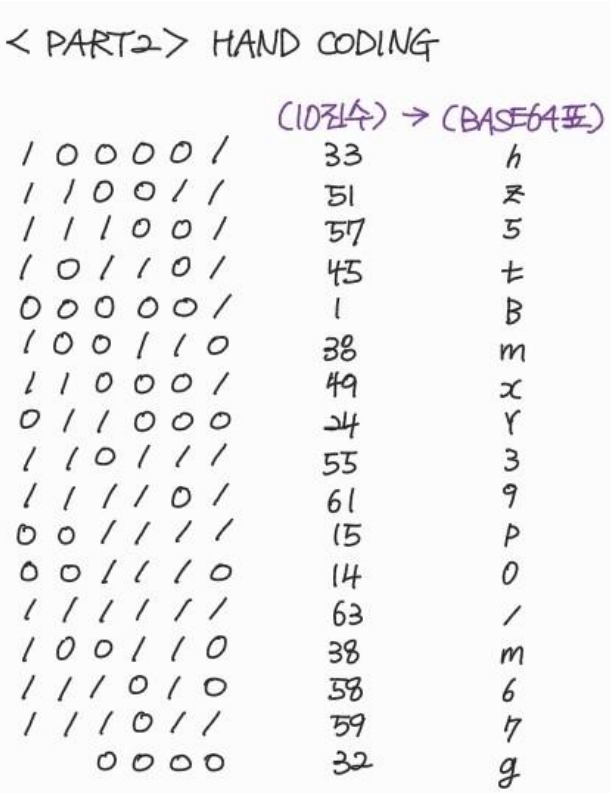
[Checksheet]

No.	Checksheet item	Done?
1.	Ascii code (part 1)	O
2.	Base64 code (part 2)	O
3.	Part1 ascii (iVerilog)	O
4.	Part2 6-bit printing	O
5.	Part2 Base64 printing	O
6.	Synthesizable	X

1. Ascii code (Hand)

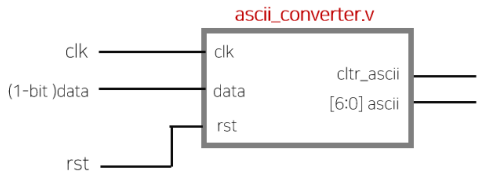


2. Base64 code (Hand)



3. ASCII Printing

1) Architecture



▲ 비트스트림 → ASCII 변환 과정

2) Description

```
module ascii_converter(clk,rst,data,ascii,complete);
  // Port Declaration
  input clk, rst, data;
  output reg complete;
  output reg [6:0] ascii;

  integer i;
  reg [6:0] tmp;

  // Merge
  always @(posedge clk, negedge rst) begin
    complete = 0;

    // 리셋 : 레지스터 값 초기화
    if (~rst) begin
      i=6; tmp=7'b0;
    end
    // 저장 : 들어오는 1-bit data를 tmp에 저장해둔다
    else begin
      tmp[i] = data;
      i = i-1;
    end

    // 출력 : 7개의 데이터가 들어올 때마다 저장되어 있던 7-bit 출력함과 동시에
    //       complete 제어 신호 ON, 레지스터 값 초기화
    if (i==-1) begin
      i = 6;
      complete = 1'b1;
      ascii = tmp;
      tmp = 7'b0;
    end
  end
endmodule
```

→ ascii_converter : 7CLK 동안 1-bit씩 입력을 받아 7-bit output 출력

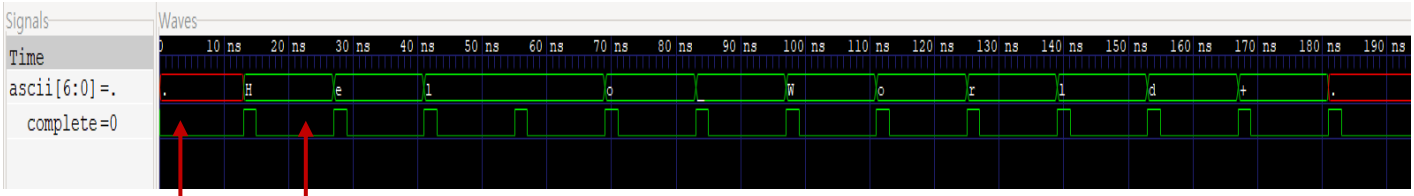
→ RST이 수행되고 나면, 모듈의 동작이 시작됩니다.

→ 매 CLK (=1ns) 마다 입력으로 들어온 데이터를 임시 저장공간(tmp)에 저장하고, 지금까지 들어온 비트 수를 카운팅합니다.

→ 7-bit가 다 들어오면(i=-1), tmp에 저장되어 있던 7-bit ASCII 값을 내보내고, complete = 1'b1로 설정한 다음 카운트를 초기화 합니다.

3. ASCII Printing

3) Results

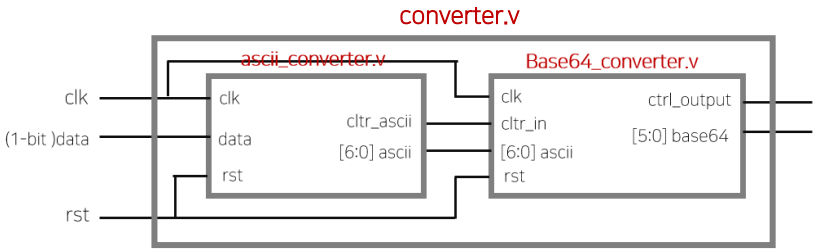


reset 이후
bitstream 입력 스타트

7개의 bit가 전달될 때마다
ASCII 값이 출력되고
(제어신호) complete = 1'b1

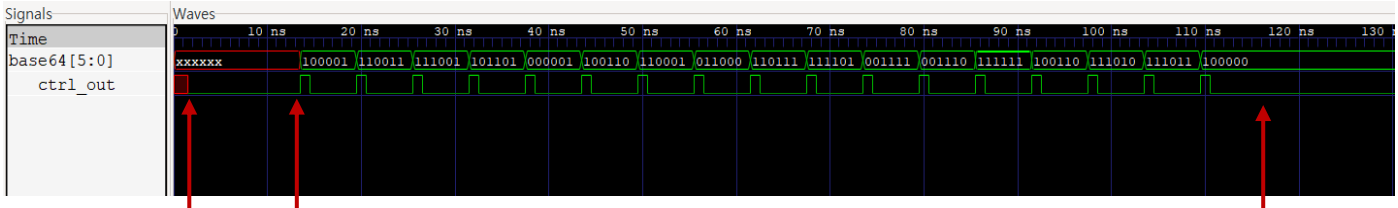
4. 6-bit printing

1) Architecture



▲ 비트스트림 → ASCII → 6bit 변환 과정

2) Results



reset 이후
bitstream 입력 스타트

6-bit output 나올 때마다
ctrl_out = 1'b1

더 이상 bitstream이 전달되지 않자,
Output 역시 출력되지 않는다

4. 6-bit printing

3) Description

```
module converter(clk,rst,data,base64,ctrl_base64);
  input clk, rst, data;
  output [5:0] base64;
  output ctrl_base64;

  wire [6:0] ascii;
  wire ctrl_ascii, ctrl_base64;

  ascii_converter a1(clk, rst, data, ascii, ctrl_ascii);
  base64_converter b1(clk, rst, ctrl_ascii, ascii, base64, ctrl_base64);
  print_base64 p1(ctrl_base64,rst,base64);
endmodule
```

```
module base64_converter(clk,rst,ctrl_in,ascii,base64,ctrl_out);
  /* Port Declaration */
  input clk, rst, ctrl_in;
  input [6:0] ascii;
  output reg ctrl_out;
  output reg [5:0] base64;

  reg [11:0] tmp;
  integer i, count, len;

  /* Convert */
  always @(posedge clk, negedge rst) begin
    if(~rst) begin // register initialization
      tmp = 12'b0; count = 0; len = 0;
    end
    else begin // not reset
      if(count==6) begin // 6-CLK마다 저장된 데이터 길이 체크해서 Base64 출력
        if (len>=6) begin // 저장되어 있는 ascii에서 6비트 출력
          base64 = tmp[11 -: 6];
          tmp = tmp << 6;
          len = len-6;
          ctrl_out = 1'b1;
        end
        else if(len>0 && len<6) begin// 0으로 채워진 값 출력
          base64 = tmp[11 -: 6];
          tmp = tmp << 6;
          len = 0;
          ctrl_out = 1'b1;
        end
        count = 1;
      end
      else begin
        count = count + 1;
        ctrl_out = 1'b0;
      end
    end
  end

  always @(posedge ctrl_in) begin
    for(i=6;i>=0;i=i-1) begin  // 제대로 된 입력(0또는1)에 대해서만 저장
      if ((ascii[i]==1'b0) || (ascii[i]==1'b1)) begin
        tmp[11-len] = ascii[i]; len = len + 1;end
    end
  end

endmodule
```

→ base64_converter : 6 CLK 마다 7-bit 입력을 받아 6-bit output 출력

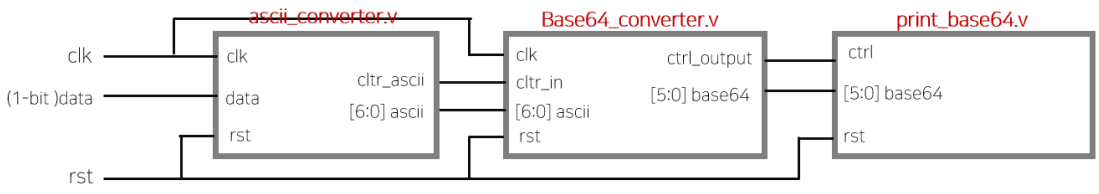
→ RST이 수행되고 나면, 모듈의 동작이 시작됩니다.

→ 이전 모듈에서 7-bit ASCII 가 들어오면 (= posedge ctrl_in), X, Z가 아닌 올바른 값이 들어왔는지 체크합니다.
입력에 문제가 없다면 들어온 ASCII 값을 임시 저장공간(tmp)에 저장합니다.

→ 이와 별개로, 모듈에서는 매 CLK마다 카운팅을 수행하고, 6번 카운트 할 때 마다 6-bit를 내보냅니다. 만약 tmp에 저장되어 있는 값이 6 이상이라면, 남아있는 데이터가 충분하므로 tmp [11:6]을 내보내고, ctrl_out = 1'b1로 설정합니다.
출력된 비트 수 만큼 tmp의 길이인 len 값을 len = len - 6으로 설정해주고, tmp = tmp << 6; 을 수행하여 이미 출력된 데이터를 제거해줌과 동시에, 0을 패딩시켜줍니다. 하지만 만약 tmp에 저장되어 있던 값이 0과 6 사이 값이라면, 데이터가 더 이상 들어오지 않고 있다는 말이므로, 패딩이 포함된 6 bit를 내보내고 len 값을 0으로 바꿔줍니다.
len = 0 이면 새로운 ASCII 값이 들어오기 전까지 새로운 6-bit 출력이 발생하지 않습니다.

5. BASE64 printing

1) Architecture



▲ 비트스트림 → ASCII → 6bit→ BASE64 까지의 과정

2) Description

```
module print_base64(ctrl, rst, base64);
  // Port Declaration
  input ctrl,rst;
  input [5:0] base64;

  integer idx;
  reg [511:0] dict;

  always @(posedge ctrl,negedge rst) begin

    // Initialize Base64 Table
    if (~rst) begin
      dict = {"/","+","9","8","7","6","5","4","3","2","1","0","z","y","x","w","v","u","t","s","r",
      "q","p","o","n","m","l","k","j","i","h","g","f","e","d","c","b","a","Z","Y","X","W","V",
      "U","T","S","R","Q","P","O","N","M","L","K","J","I","H","G","F","E","D","C","B","A"};
    end
    else begin
      // Select corresponding index
      idx = base64; idx = idx * 8;

      // Print Base64
      $write("%s  ",dict[idx +: 8]);
    end
  end
endmodule
```

→ print_base64 : 6-bit 입력을 받아 BASE64 코드 출력

→ dict 레지스터에 BASE64 Table을 저장해두고 사용합니다.

6-bit가 전달될 때마다(= posedge ctrl) 해당하는 레지스터 인덱스를 계산하여 Base64 코드를 CMD에 출력합니다

3) Results