T1

```
module sha256_round (input [31:0] Kt,
                     Wt,
                     input [31:0] a_in,
                     b_in,
                     c_in,
                     d_in,
                     e_in,
                     f_in,
                     g_in,
                     h_in,
                     output [31:0] a_out,
                     b_out,
                     c_out,
                     d_out,
                     e_out,
                     f_out,
                     g_out,
                     h_out);

wire [31:0] S0,S1,Ch,Maj,t1,t2;
sha256_S0 u_sha256_S0(
.x  (a_in),
.S0 (S0)
);
sha256_S1 u_sha256_S1(
.x  (e_in),
.S1 (S1)
);
Ch u_Ch(
.x  (e_in),
.y  (f_in),
.z  (g_in),
.Ch (Ch)
);
Maj u_Maj(
.x   (a_in),
.y   (b_in),
.z   (c_in),
.Maj (Maj)
);
assign t2    = (S0+Maj);
assign t1    = (h_in+S1+Ch+Kt+Wt);
assign a_out = (t1+t2);
assign b_out = a_in;
assign c_out = b_in;
assign d_out = c_in;
assign e_out = (d_in+t1);
assign f_out = e_in;
assign g_out = f_in;
assign h_out = g_in;
```

```
endmodule


module sha256_S0 (
    input wire [31:0] x,
    output wire [31:0] S0
    );
    assign S0 = ({x[1:0], x[31:2]} ^ {x[12:0], x[31:13]} ^ {x[21:0],
    x[31:22]});
endmodule

    module sha256_S1 (
        input wire [31:0] x,
        output wire [31:0] S1
        );
        assign S1 = ({x[5:0],x[31:6]} ^ {x[10:0],x[31:11]} ^
{x[24:0],x[31:25]});
    endmodule

        module Ch (
            input wire [31:0] x, y, z,
            output wire [31:0] Ch
            );
            assign Ch = ((x & y) ^ (~x & z));
        endmodule

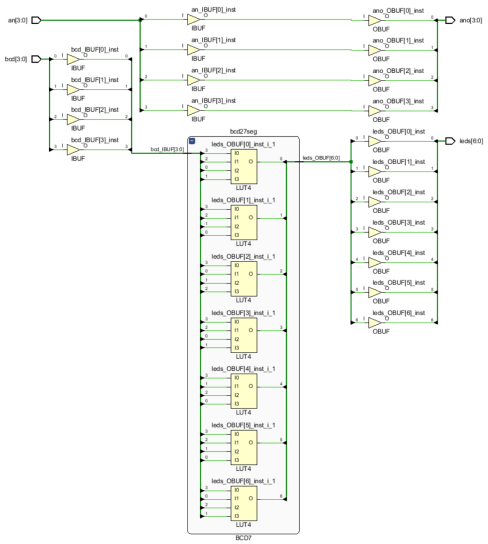
            module Maj (
                input wire [31:0] x, y, z,
                output wire [31:0] Maj
                );
                assign Maj = ((x&y)|(y&z)|(z&x));
            endmodule
```

T2

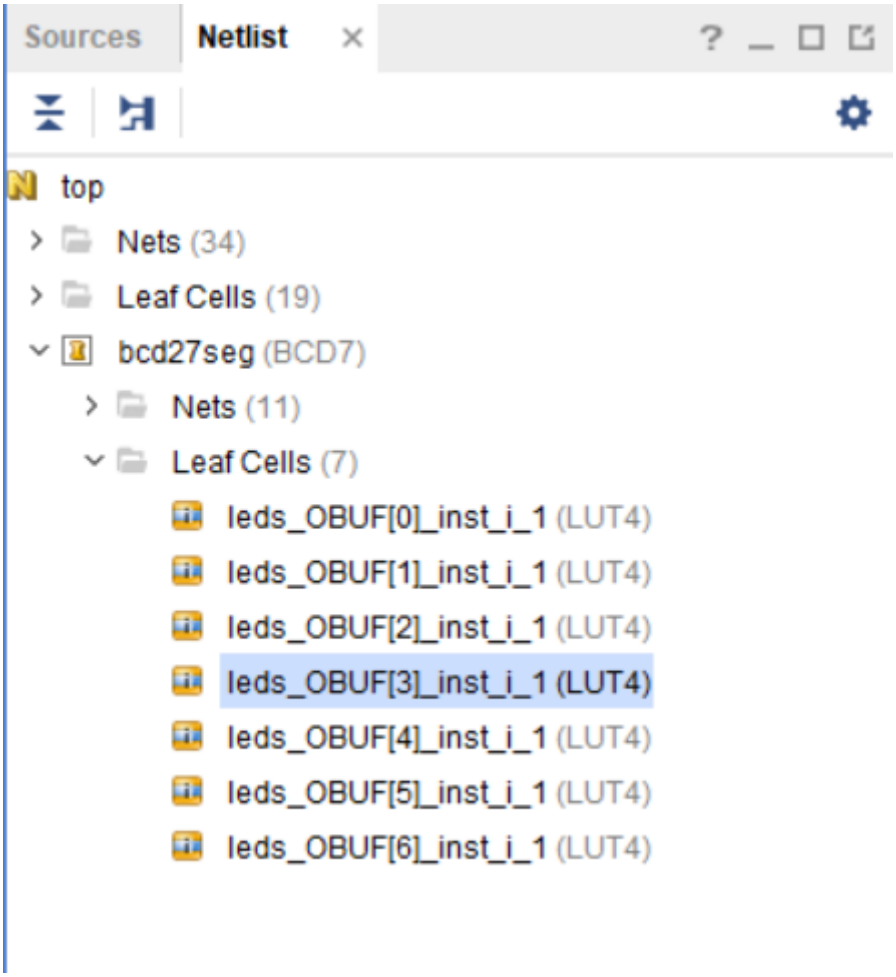
1. ```
   module BCD7(din,
               dout);
       input [3:0] din;
       output [6:0] dout;
       reg [6:0] dout;

       always @(*) begin
           case (din)
               4'h0: dout   <= 7'b1000000;
               4'h1: dout   <= 7'b1111001;
               4'h2: dout   <= 7'b0100100;
               4'h3: dout   <= 7'b0110000;
               4'h4: dout   <= 7'b0011001;
               4'h5: dout   <= 7'b0010010;
               4'h6: dout   <= 7'b0000010;
               4'h7: dout   <= 7'b1111000;
               4'h8: dout   <= 7'b0000000;
   ```
   ```

```
            4'h9: dout   <= 7'b0010000;
            default:dout <= 7'b1111111;
        endcase
    end
endmodule
```

2. 

3. d管的LUT的相对结构如下图：



查找表为16'hEA9C，即16b’1110_1010_1001_1100

验证：由于有bcd_IBUF[3]=I0,bcd_IBUF[2]=I1,bcd_IBUF[0]=I2,bcd_IBUF[1]=I3

可以发现查找结果与dout中第4位相同（控制d管），验证成立

din	bcd_IBUF	I[3:0]	查找结果	dout
4'h0	4'b0000	4'b0000	0	7'b100**0**000
4'h1	4'b0001	4'b0100	1	7'b111**1**001
4'h2	4'b0010	4'b1000	0	7'b010**0**100
4'h3	4'b0011	4'b1100	0	7'b011**0**000
4'h4	4'b0100	4'b0010	1	7'b001**1**001
4'h5	4'b0101	4'b0110	0	7'b001**0**010
4'h6	4'b0110	4'b1010	0	7'b000**0**010
4'h7	4'b0111	4'b1110	1	7'b111**1**000
4'h8	4'b1000	4'b0001	0	7'b000**0**000
4'h9	4'b1001	4'b0101	0	7'b001**0**000

T3

同意。商业FPGALUT的输入端子数目限制在4到6，可以平衡电路性能和芯片面积的需求。

如果输入端子数目更少，那么其能够实现的逻辑功能就会受到限制，需要更多的LUT来组合成复杂的逻辑单元，从而增加了布线延迟和功耗。这对大规模集成电路而言不利于集成；如果输入端子数目更多，虽然能实现更多逻辑功能，但会增加LUT内部和外部的复杂度和延迟。而且对于高速应用来说，LUT内部和外部的延迟往往是关键因素。综合来看，不同的应用场景有不同的最优选择，但是一般来说，限制在4到6之间选择可以达到一个较好的折中效果。