BobAnkh

1. SHA_256 中的变换

```
// round compression function
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
    );
// 请在此补充完整
wire [31:0] t_1, t_2, ch, s_1, s_0, maj;
Ch ch_out(e_in, f_in, g_in, ch);
sha256_S1 s1_out(e_in, s_1);
Maj maj_out(a_in, b_in, c_in, maj);
sha256_S0 s0_out(a_in, s_0);
assign t_2 = (s_0 + maj);
assign t_1 = (h_in + s_1 + ch + Kt + Wt);
assign a_out = (t_1 + t_2);
assign e_out = (d_in + t_1);
assign {b_out, c_out, d_out, f_out, g_out, h_out } = {a_in, b_in,
 c_in, e_in, f_in, g_in };
endmodule

//Σ0(x)
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

//Σ1(x)
module sha256_S1 (
    input wire [31:0] x,
    output wire [31:0] S1
    );
//请在此补充完整
assign S1 = ({x[5:0], x[31:6]} ^ {x[10:0], x[31:11]} ^ {x[24:0],
x[31:25]});
endmodule
```

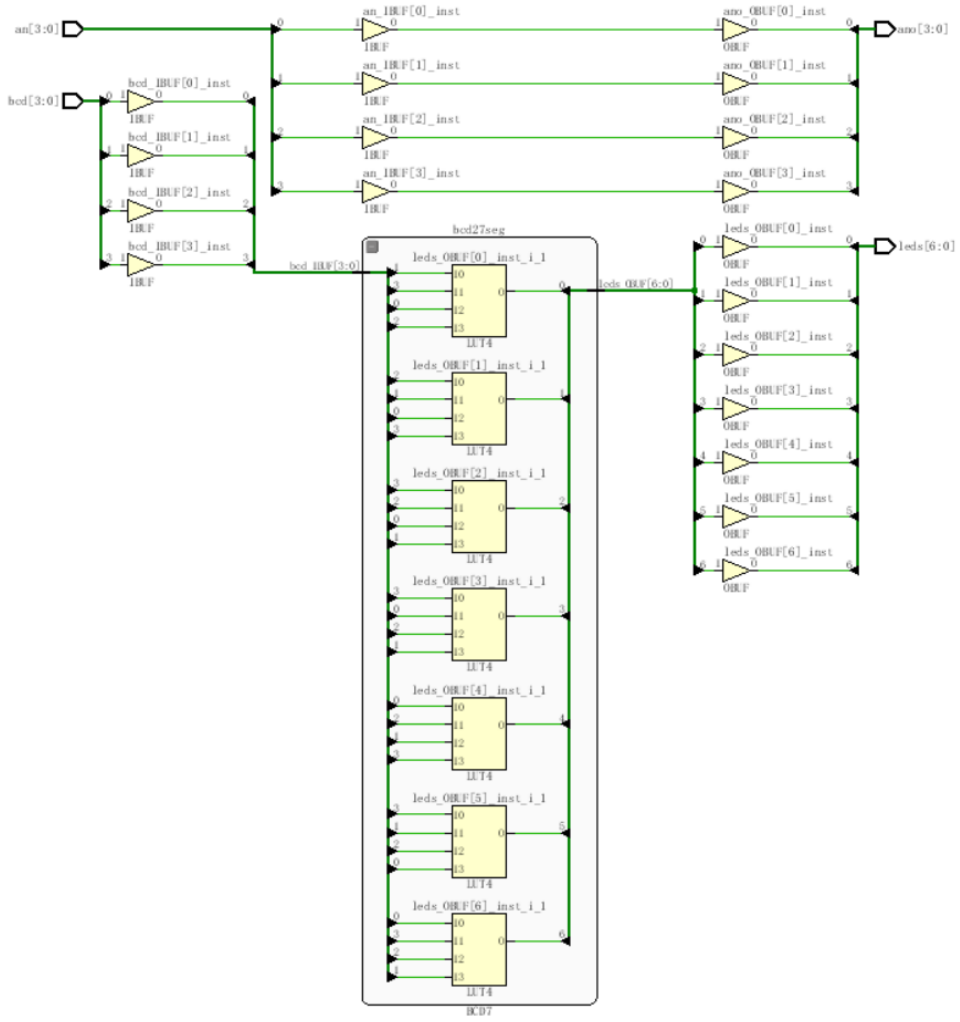
```
// Ch(x,y,z)
module Ch (
    input wire [31:0] x, y, z,
    output wire [31:0] Ch );
assign Ch = ((x & y) ^ (~x & z));
endmodule

// Maj(x,y,z)
module Maj (
    input wire [31:0] x, y, z,
    output wire [31:0] Maj
    );
//请在此补充完整
assign Maj = ((x & y) | (y & z) | (z & x));
endmodule
```

2. 七段译码器的实现

（1）

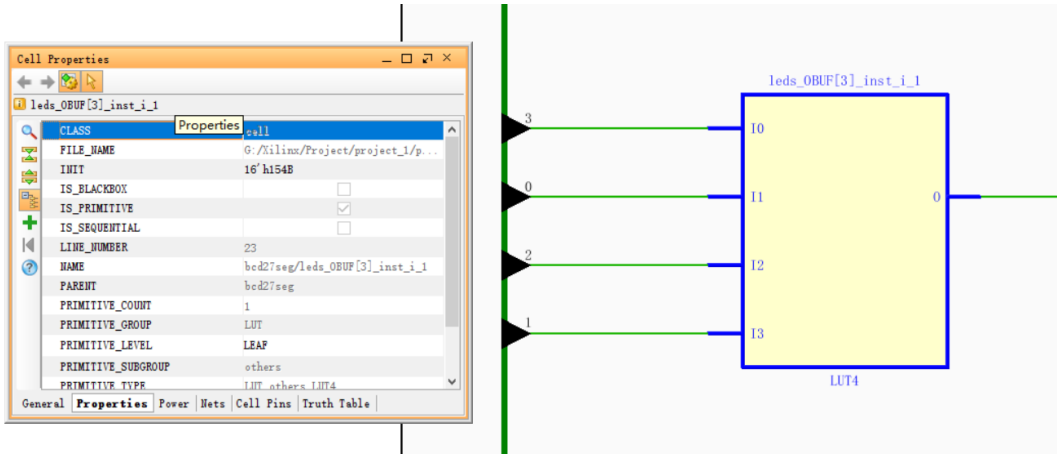
```
module BCD7(
    din,
    dout
);
input [3:0] din;
output [6:0] dout;

reg [6:0] dout;

always @(*)
begin
    if(din==4'h0) dout=7'b0111111;
    else if(din==4'h1) dout=7'b0000110;
    else if(din==4'h2) dout=7'b1011011;
    else if(din==4'h3) dout=7'b1001111;
    else if(din==4'h4) dout=7'b1100110;
    else if(din==4'h5) dout=7'b1101101;
    else if(din==4'h6) dout=7'b1111101;
    else if(din==4'h7) dout=7'b0000111;
    else if(din==4'h8) dout=7'b1111111;
    else if(din==4'h9) dout=7'b1101111;
    else dout=7'b0;
end
endmodule
```

（2）电路原理图结构



（3）验证配置字

查阅资料并分析可知，七段数码管中的 d 段对应的是代码中的 leds[3]，则找到其对应的 LUT 分析其配置字：



可以看到，该 LUT 的 16 位的对应地址输出为 INIT=16'h154B 转换为二进制即为 16'b0001_0101_0100_1011

再由图中的对应关系可以得知：I0=bcd_IBUF[3]=din[3]，I1=bcd_IBUF[0]=din[0]，I2=bcd_IBUF[2]=din[2]，I3=bcd_IBUF[1]=din[1]

则可进行验证：

din	bcd_IBUF[3:0]	I[3:0]	输出
4’h0	4’b0000	4’b0000	1
4’h1	4’b0001	4’b0010	0
4’h2	4’b0010	4’b1000	1
4’h3	4’b0011	4’b1010	1
4’h4	4’b0100	4’b0100	0
4’h5	4’b0101	4’b0110	1
4’h6	4’b0110	4’b1100	1
4’h7	4’b0111	4’b1110	0
4’h8	4’b1000	4’b0001	1
4’h9	4’b1001	4’b0011	1
4’h10	4’b1010	4’b1001	0
4’h11	4’b1011	4’b1011	0
4’h12	4’b1100	4’b0101	0
4’h13	4’b1101	4’b0111	0
4’h14	4’b1110	4’b1101	0
4’h15	4’b1111	4’b1111	0

可以看到输出与（1）中代码对应的相符合（即 dout 第四位），可以验证配置字。

3. LUT 的输入端子

我赞同这样的产品设计。商业界工业界显然也是要追求成本和效益的，而目前主流的商业 FPGA 产品主要使用 4-6 输入端的 LUT 也是有其原因的，主要从面积和性能的角度综合考量来决定的。

业界在过去已经通过实际的 CAD 实验表明：使用 4-LUT 可以获得最有效的面积[1][2]，从文献[1]的图中可以表明，4 输入左右的 LUT 的平均归一化面积是相对较小的，可以看到超过 6 输入的 LUT 的面积会有一个明显的陡升，这使得使用较多输入端的 LUT 的开销会明显增大，效用就会下降。

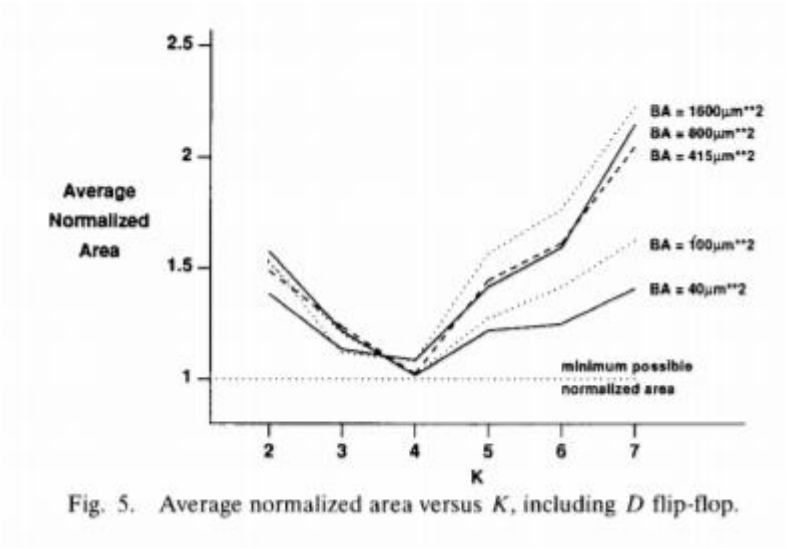


Fig. 5. Average normalized area versus K, including D flip-flop.

此外，根据文献[3]所示图，可以发现，在各种情况下，相对偏小的输入端数目的关键路径延时较为良好。

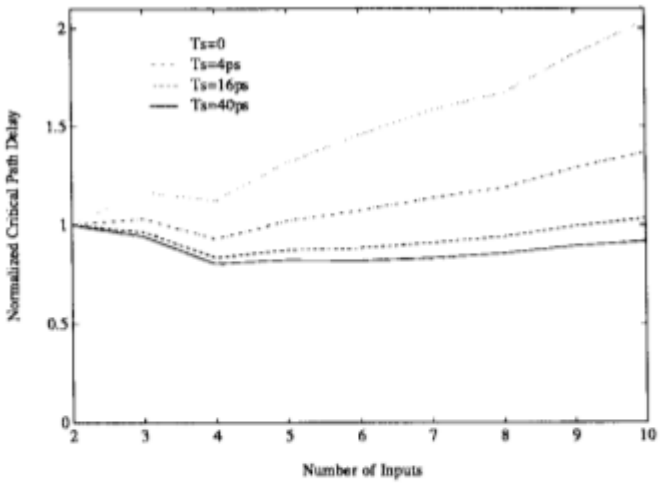


Figure 5: Critical Path Delay vs. Number of Inputs vs. τ_s. The top curve, $\tau_s = 0$, shows the critical path delay with zero interconnect delay. The curves for $\tau_s = 4$ps. and $\tau_s = 16$ps. correspond to programming technologies such as the antifuse and pass transistor, respectively.

文献[5]中的表格也展示了某一种 LUT 的延时，随着输入端数目的增加而不断增大。

TABLE II
LUT DELAYS USING 0.18-μm CMOS PROCESS

LUT Size (K)	C to D (ps)
2	199
3	283
4	401
5	534
6	662
7	816

输入数目更多虽然可以带来更容易的实现，可以更容易的适配各种电路的设计，但是数目的增加也会引入一些电路性能上的下降，所以需要综合考量，即希望能够尽量选择实现设计上方便的，也要控制延时和面积。显然，2-3 输入端的面积实现可以接受，延时较小，也有文献[6]表明，混合使用 2-LUT 和 3-LUT 也可以获得与 4-LUT 相当的面积有效性，不过这样显然相对复杂而并非目前的主流使用。而超过 6 输入的更大输入端树木的 LUT 带来的延时提高和占用面积过大的问题也使得性能相对下降而不被主流使用。文献[3]和[4]也表明，使用 4-6 输入的 LUT 可以获得最好性能，而文献[5]也说明了，在深亚微米工艺之下，输入端数目在 4-6 之间时面积最为有效，6-LUT 可以获得最好性能。

所以在综合考量占用芯片面积以及延时等电路性能和实际使用等因素情况下，4-6 输入端数目的 LUT 是一种合理的、较优的选择。

参考文献：

[1] Rose, Jonathan, et al. "Architecture of field-programmable gate arrays: The effect of logic block functionality on area efficiency." *IEEE Journal of Solid-State Circuits* 25.5 (1990): 1217-1225.

[2] Kouloheris, J., and Abbas El Gamal. "FPGA area versus cell granularity-lookup tables and PLA cells." *FPGA*. Vol. 92. 1992.

[3] Kouloheris, Jack L., and Abbas El Gamal. "FPGA performance versus cell granularity." *IEEE Custom Integrated Circuits Conference*. Vol. 6. No. 1. 1991.

[4] Singh, Satwant, et al. "The effect of logic block architecture on FPGA performance." *IEEE Journal of Solid-State Circuits* 27.3 (1992): 281-287.

[5] Ahmed, Elias, and Jonathan Rose. "The effect of LUT and cluster size on deep-submicron FPGA performance and density." *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 12.3 (2004): 288-298.

[6] Kaptanoglu, Sinan, et al. "A new high density and very low cost reprogrammable FPGA architecture." *Proceedings of the 1999 ACM/SIGDA seventh international symposium on Field programmable gate arrays*. 1999.