Tco 探秘（原创）

getmoon@gmail.com

希望大家喜欢，如果有什么不正确之处，请指出，必改。

1. 何为 Tco

在 FPGA 中，Tco 有两种：

(1) 触发器 Tco

(2) 管脚输出 Tco

触发器 Tco 由 FPGA 的器件速度等级，工艺决定。一般在几百 ps 左右。管脚输出 Tco 是指从输出触发器信号从管脚输出的延迟。本文指讨论管脚 Tco。

2. Tco 的作用

在 FPGA 和外部芯片由同步通信时，Tco 是保证系统能够工作与设定频率的重要因素。假设当前 A 芯片输出信号到 B 芯片。为了能够使 A 芯片的数据达到 B 芯片，并且满足 B 芯片的 setup/hold 时间要求。必须保证 ，

A 芯片的 Tco + B 芯片的 Tsu < T

3. Tco 的组成

Tco 的延迟有三部分组成：

- 输出触发器的触发器 Tco
- 输出触发器输出管脚到 IOE 的走线延迟
- IOE 内部延迟

在这 3 个延迟中，触发器内部 Tco 非常小，只有几百个 ps，相对于其他两个延迟，可以忽略不计。

4. Tco 的优化

为了优化Tco , quartus 提供了一个优化选项，就是“ Fast Output Register “ 。 意思是使用IOE中的输出寄存器直接用于逻辑寄存器。这样可以减少输出寄存器到pad的走线距离，达到优化Tco的目的。

另外，quartus 的 Tco 的计算方法和前面有所不同, quartus 的 Tco 的计算如下所示：

- 输入时钟管脚和输出触发器时钟之间的 skew
- 触发器内部 Tco
- 触发器-> IOE -> pad 延迟

可以看出，3 小节提到的计算方法是以输出触发器的时钟为参考的延迟。 而 quartus 计算的方法是以时钟输入管脚为参考。

Quartus 的分析如下所示：

Info: Slack time is 15 ps for clock "Clk[0]" between source register "out[3]~reg0" and destination pin "out[3]"
Info: + tco requirement for source register and destination pin is 5.000 ns
Info: - tco from clock to output pin is 4.985 ns
Info: + Longest clock path from clock "Clk[0]" to source register is 2.401 ns
Info: 1: + IC(0.000 ns) + CELL(1.469 ns) = 1.469 ns; Loc. = PIN_29; Fanout = 4; CLK Node = 'Clk[0]'
Info: 2: + IC(0.723 ns) + CELL(0.209 ns) = 2.401 ns; Loc. = IOC_X0_Y1_N0; Fanout = 1; REG Node = 'out[3]~reg0'
Info: Total cell delay = 1.678 ns (69.89 %)
Info: Total interconnect delay = 0.723 ns (30.11 %)
Info: + Micro clock to output delay of source is 0.664 ns
Info: + Longest register to pin delay is 1.920 ns
Info: 1: + IC(0.000 ns) + CELL(0.000 ns) = 0.000 ns; Loc. = IOC_X0_Y1_N0; Fanout = 1; REG Node = 'out[3]~reg0'
Info: 2: + IC(0.000 ns) + CELL(1.920 ns) = 1.920 ns; Loc. = PIN_59; Fanout = 0; PIN Node = 'out[3]'
Info: Total cell delay = 1.920 ns (100.00 %)

5. **Tco 使用分析**

为了分析 Tco 写了如下一个例子来进行分析。

```
module cnt1(Clk,Reset_,in , out);
    input    [0:0] Clk;
    input    [0:0] Reset_;
     input [3:0] in;
    output [3:0] out;

reg       [3:0] out;

always @(posedge Clk)
begin
    if (!Reset_)
        out<=0;
    else
        out<=in;
end

endmodule
```

该例子非常简单，仅仅是将输入数据打一拍输出。

Step1:

选用器件 EP1C6Q240C8 . Fmax = 80M. 没有使用任何约束和优化选项。

我们获得的结果 Tco 是：

Slack	Required tco	Actual tco	From	To	From Clock
N/A	None	6.838 ns	out[0]~reg0	out[0]	Clk[0]
N/A	None	6.735 ns	out[3]~reg0	out[3]	Clk[0]
N/A	None	6.734 ns	out[2]~reg0	out[2]	Clk[0]
N/A	None	6.396 ns	out[1]~reg0	out[1]	Clk[0]

Step2:

选用器件 EP1C6Q240C6 . Fmax = 80M.没有使用任何约束和优化选项。

我们获得的结果 Tco 是：

Slack	Required tco	Actual tco	From	To	From Clock
N/A	None	5.254 ns	out[0]~reg0	out[0]	Clk[0]
N/A	None	5.177 ns	out[3]~reg0	out[3]	Clk[0]
N/A	None	5.175 ns	out[2]~reg0	out[2]	Clk[0]
N/A	None	4.916 ns	out[1]~reg0	out[1]	Clk[0]

小结：

通过对比 step1 和 step2 可以看出，选用的器件不同，Tco 有着较大的差别。

Step3：

选用器件 EP1C6Q240C6 . Fmax = 80M. Tco 约束 5ns . 没有优化选项。

我们获得的结果 Tco 是：

Slack	Required tco	Actual tco	From	To	From Clock
N/A	5ns	4.985 ns	out[0]~reg0	out[0]	Clk[0]
N/A	5ns	4.985 ns	out[3]~reg0	out[3]	Clk[0]
N/A	5ns	4.985 ns	out[2]~reg0	out[2]	Clk[0]
N/A	5ns	4.985 ns	out[1]~reg0	out[1]	Clk[0]

Step4:

选用器件 EP1C6Q240C6 . Fmax = 80M. Tco 约束 5ns，使能 “ Fast Output Register” .

我们获得的结果 Tco 是：

Slack	Required tco	Actual tco	From	To	From Clock
N/A	None	4.985 ns	out[0]~reg0	out[0]	Clk[0]
N/A	None	4.985 ns	out[3]~reg0	out[3]	Clk[0]
N/A	None	4.985 ns	out[2]~reg0	out[2]	Clk[0]
N/A	None	4.985 ns	out[1]~reg0	out[1]	Clk[0]

Step4:

选用器件 EP1C6Q240C6 . Fmax = 80M. 没有约束，使能 “ Fast Output Register” .

我们获得的结果 Tco 是：

Slack	Required tco	Actual tco	From To	From Clock
N/A	5ns	4.985 ns	out[0]~reg0	out[0]	Clk[0]
N/A	5ns	4.985 ns	out[3]~reg0	out[3]	Clk[0]
N/A	5ns	4.985 ns	out[2]~reg0	out[2]	Clk[0]
N/A	5ns	4.985 ns	out[1]~reg0	out[1]	Clk[0]

小结：

从 step3 , 4 , 5 和 step1 , 2 看出， 加上约束或者使能“ Fast Output Register” .使得 Tco 得到了很大的改善。

6. 结论

本文通过分析 Tco 的作用，组成，并且通过实际例子对比，获得使用约束，或者“ Fast Output Register” 能较好的改善 Tco 的方法.

getmoon@gmail.com