Řešení domácího úkolu

Cvičení 1, analýza chování bezchybného obvodu. Spusťte verifikační simulaci (VS). Prohlédněte si časové průběhy hodnot signálů a všimněte si jak je modelována třístavová sběrnice a funkce řídícího automatu. Spusťte kompletní implementaci v ISE WebPack (KI), jako cílový obvod zvolte xc2s15-5cs144.

Jak velký je výsledný obvod?

Informace o velikosti jsou obsažené ve výstupu programu „map“, v souboru s koncovkou .mrp (např. top_level.mrp). Pro ukázkový příklad je velikost

```
Logic Utilization:
  Number of Slice Flip Flops:        14 out of    384    3%
  Number of 4 input LUTs:            55 out of    384   14%
Logic Distribution:
    Number of occupied Slices:                          31 out of    192   16%
    Number of Slices containing only related logic:     31 out of     31  100%
    Number of Slices containing unrelated logic:         0 out of     31    0%
        *See NOTES below for an explanation of the effects of unrelated logic
Total Number of 4 input LUTs:        55 out of    384   14%
   Number of bonded IOBs:            27 out of     86   31%
      IOB Flip Flops:                                9
   Number of GCLKs:                   1 out of      4   25%
   Number of GCLKIOBs:                1 out of      4   25%

Total equivalent gate count for design:  604
Additional JTAG gate count for IOBs:  1,344
Peak Memory Usage:  121 MB
```

Jaká je maximální dosažitelná hodinová frekvence a kudy prochází kritická cesta v obvodu?

Statickou časovou analýzu spustíme pomocí okna „Processes“ v ISE, je skrytá pod Implement Design/Place & Route/Generate Place & Route Static Timing/Analyze Post-Place & Route Static Timing (Timing Analyzer). Po spuštění aplikace vybereme z menu Analyze/Analyze against auto generated timing constraints (constraints žádné nemáme) a klikneme na OK.

Ve zprávě o statické časové analýze je napsáno následující:

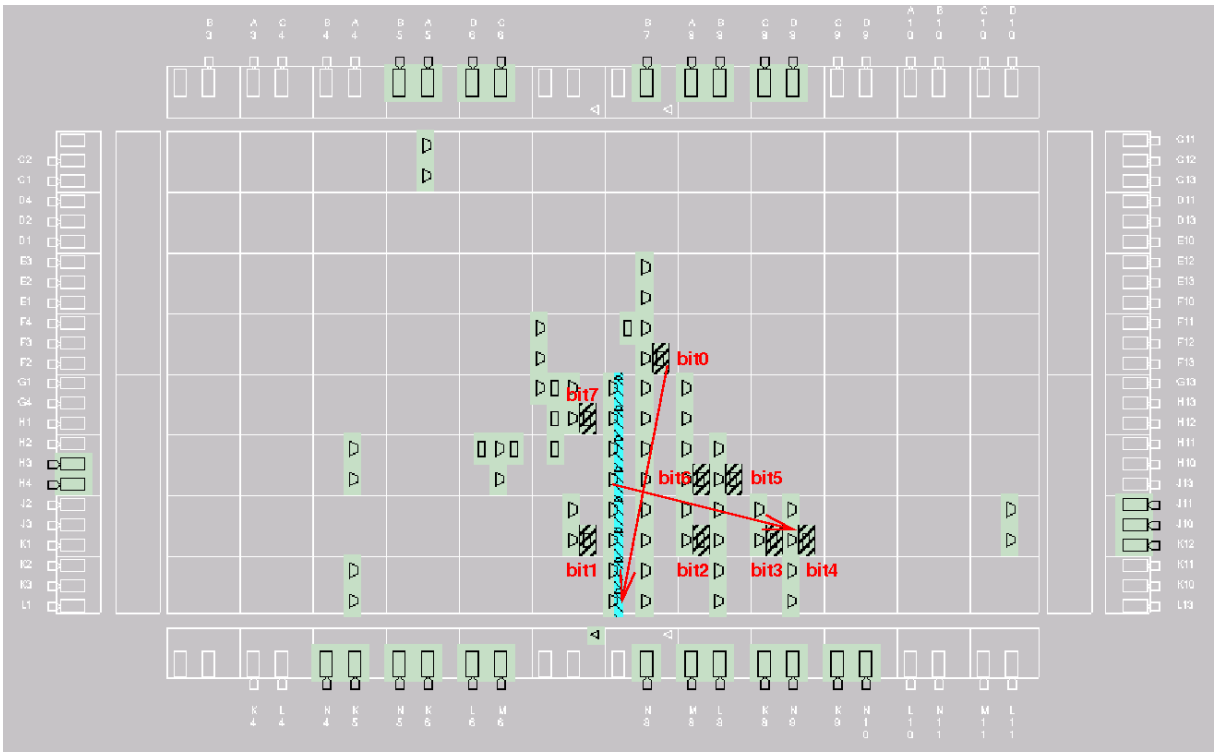
```
================================================================================
Timing constraint: Default period analysis for net "clk_BUFGP"

 282 items analyzed, 0 timing errors detected. (0 setup errors, 0 hold errors)
 Minimum period is   9.092ns.
```

Tím je určena minimální perioda hodin, maximální hodinová frekvence je tedy cca 109MHz. Kritická cesta je určena v dalším textu:

```
Data Path: i_alu/a_q_0 to i_alu/a_q_4
  Delay type         Delay(ns)  Logical Resource(s)
  -------------------------  -------------------
  Tcko                  1.292   i_alu/a_q_0
  net (fanout=4)        1.801   i_alu/a_q<0>
  Topcyf                1.487   i_alu/alu__n0002<0>lut
                                i_alu/alu__n0002<0>cy
                                i_alu/alu__n0002<1>cy
  net (fanout=1)        0.000   i_alu/alu__n0002<1>_cyo
  Tbyp                  0.096   i_alu/alu__n0002<2>cy
                                i_alu/alu__n0002<3>cy
  net (fanout=1)        0.000   i_alu/alu__n0002<3>_cyo
  Tcinx                 0.463   i_alu/alu__n0002<4>_xor
```

```
    net (fanout=1)        0.925   i_alu/_n0002<4>
    Tilo                  0.653   i_alu/alu_op_sel<0>9
    net (fanout=1)        0.695   i_alu/mux_2_alu_op_sel<1>_MUXF54
    Tdick                 1.680   i_alu/Mmux__n0001__n0001<0>__n0001<0>_rn_2111
                                  i_alu/a_q_4
    ----------------------------  --------------------------
    Total                 9.092ns (5.671ns logic, 3.421ns route)
                                  (62.4% logic, 37.6% route)
```

Vidíme, že začíná v registru a_q(0) prochází sčítačkou v ALU, a končí v registru a_q(4). Může se zdát zvláštní, že kritická cesta neprochází celou sčítačkou (končila by pak v registru a_q(7)). To je způsobeno rozložením jednotlivých registrů registru a_q na skutečném FPGA obvodu, viz níže. Všimněte si že bity registru 0 a 4 jsou daleko od vlastní sčítačky. Šipky v obrázku představují schematické znázornění spojů v FPGA obvodu, jsou jen orientační, skutečné spoje prochází přes řadu dalších prvků (jak infrastrukturních – podpora pro propojovací kanály, tak funkčních – např. multiplexer za sčítačkou).



Obrázek 1: Floorplan FPGA obvodu. Jednotlivé bity akumulátoru – registru a_q – jsou označeny červenými texty.

Jak jsou kódovány stavy řídícího automatu ve skutečném obvodu?

Kódování stavů lze zjistit ze zprávy o syntéze ve které můžeme nalézt následující informace:

```
    Optimizing FSM <i_ctrl/curr_state> on signal <curr_state[1:3]> with
gray encoding.
    --------------------
     State  | Encoding
    --------------------
```

```
 reset  | 000
 ofset  | 001
 invert | 011
 secti1 | 010
 a_do_b | 110
 adc    | 111
 secti2 | 101
 hotovo | 100
--------------------
```

Všimněte si, že nástroj automaticky zvolil kódování, v tomto případě stavům přiřadil sekvenci kódových slov v Grayově kódu.

Cvičení 2, chybné vložení hladinového klopného obvodu. Ve zdrojovém kódu ve výpisu 1 zakomentujte řádky 39 a 40 – větev *ELSE* příkazu *IF*. Spusťte VS.

Odhalila VS změnu chování?

Ne, proběhla bez problémů.

Spusťte KI a prostudujte výstup z nástroje pro syntézu. Najděte varování, které upozorňuje na chybu v kódu.

Varování lze nalézt ve zprávě o syntéze:

WARNING:Xst:737 - Found 8-bit latch for signal <b_d>.

Cvičení 3, chybný citlivostní seznam. Ponechte na řádku 26 ve výpisu 1 v CS pouze signál *op_res*, ostatní vymažte. Spusťte VS i KI.

Odhalila VS změnu v chování návrhu?

Ano, v tomto případě chybu detekoval mechanizmus automatické kontroly odezev obvodu zabudovaný v testbenchi:

```
# ** Failure: tesbench.vhd: chybny vypocet - na vystupu je 00000000 misto 01101111!
#    Time: 1500 ns  Iteration: 0  Process: /testbench/testbench_control File: testbench.vhd
```

Prohlédněte si také výstup z nástroje pro syntézu. Upozorňuje syntezátor na potenciální rozdíl v chování mezi návrhem na RTL úrovni a skutečným obvodem?

Varování lze nalézt ve zprávě o syntéze:

WARNING:Xst:819 - "F:/tmp/automatizace/automatizace/alu.vhd" line 26: The following signals are missing in the process sensitivity list:

reg_a_we, dbus, a_q, reg_b_we, b_q.

Krátký úvod do práce se simulátorem ModelSim

Pro zájemce o práci v číslicovém simulátoru ModelSim je přiložen i základní kompilační skript spolu s příkazy pro spuštění simulace. Po spuštění simulátoru se nejprve přepněte do adresáře s rozbaleným balíčkem s příkladem. To provedete pomocí příkazu cd zadaného v okně Transcript:

```
ModelSim>cd C:/dokumenty/jakub/prispevky2010/kniha/priklady/04_jazyk_vhdl
```

V balíčku je připraven skript který vytvoří pracovní knihovnu pro kompilaci, zkompiluje kódy a spustí simulaci. Skript spustíte příkazem do:
ModelSim>do compile.do

Po úspěšné kompilaci se spustí simulátor a proběhne simulace demonstračního obvodu. Pokud budete chtít změnit zdrojové kódy a opětovně spustit simulaci, proveďte změny libovolným textovým editorem a opět spusťte skript compile.do.