Curso de Introdução à Design de Hardware Reconfigurável em FPGA

Profs. Amanda Martinez e Zoé Magalhães

22/07/2023

Lista de Exercícios 1

O objetivo desta lista de exercícios é conduzir os alunos através de uma série de desafios que, ao serem concluídos, resultarão no desenvolvimento de um projeto funcional: um cronômetro. Em todos os exercícios da lista fornecemos dicas para auxiliar no desenvolvimento, mas é importante salientar que existem outras opções de desenvolvimento, ou seja, não é mandatório seguir o fluxo de desenvolvimento que estamos sugerindo.

Exercício 1 - Criar um contador de 1 segundo (***contador decrescente***), sabendo que o clock do FPGA é 100 MHz.

Dicas:

1. Crie um projeto chamado **list1** seguindo os passos apresentados no Laboratório 1 da aula.
2. Adicione um design source chamado list1.vhd. Na definição do módulo, defina o nome da entidade como **list1** da arquitetura como **RTL**. Na definição das portas, inclua as seguinte portas:
    - clk sinal de clock
    - rst reset síncrono ativo em alto
    - cnt porta de saída
3. Para que o contador seja equivalente a 1 segundo, sabendo-se que o clock do sistema é 100 MHz, devemos alterar o tipo da porta cnt para o tipo que representa valores inteiros de 0 a 99.999.999. Porém, como neste exercício faremos a simulação somente desse contador de 1 segundo, para que a simulação não demore tanto, podemos usar apenas para fins de teste o valor de 999.

```
cnt : out integer range 0 to 999
```

4. Declare na arquitetura um sinal count_reg do tipo inteiro para representação entre 0 a 999 (pelo mesmo motivo descrito no item 3, trocamos o valor de 99.999.999 por 999).

```
signal count_reg: integer range 0 to 999;
```

5. Adicione um processo que decrementa o sinal count_reg a cada borda de subida do clock

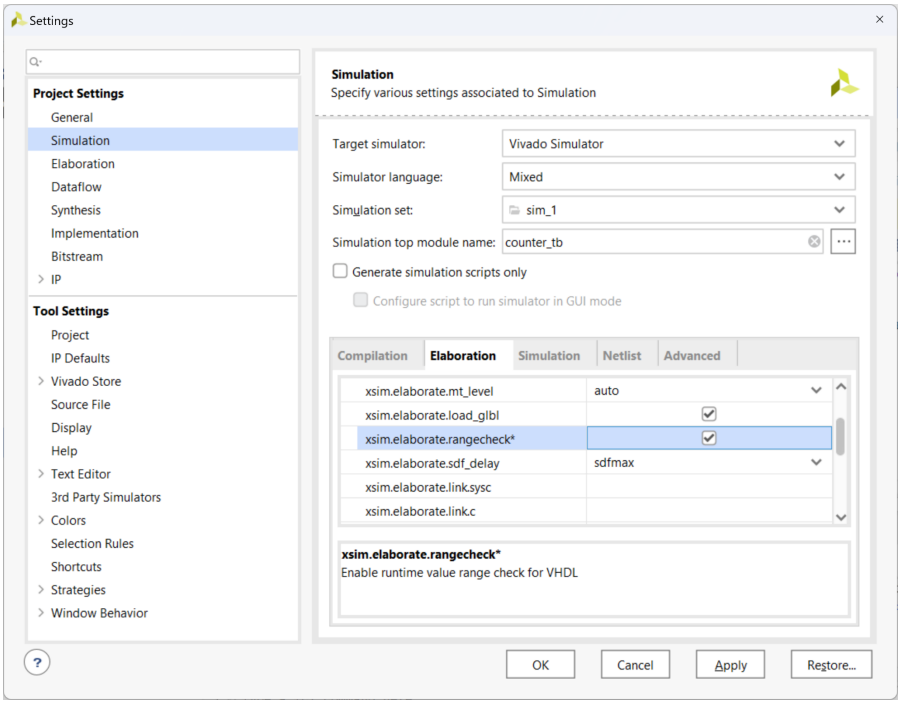
6. Adicione a esse processo uma lógica para iniciar a contagem em 999 (pelo mesmo motivo descrito no item 3, trocamos o valor de 99.999.999 por 999) na condição de reset (borda de subida do clock em que a porta rst está em nível lógico alto).
7. Conecte a saída cnt ao sinal count_reg
8. Utilize a ferramenta de *RTL ANALYSIS* do Vivado para verificar o RTL gerado.
9. Execute a simulação com o testbench modelo do **Anexo I**.

Estouro do tipo inteiro

O estouro do contador do tipo inteiro deve ser explicitamente tratado pela lógica. Por isso, antes de decrementar o contador, verifique se ele é igual a zero, caso o contador seja igual a zero ele deve ser configurado novamente para 999 (pelo mesmo motivo descrito no item 3, trocamos o valor de 99.999.999 para 999).

A ferramenta de simulação é capaz de detectar erros de estouro do *range* definido para dados do tipo *integer*. Essa funcionalidade vem desabilitada por padrão porque ela deixa a simulação mais lenta. Assim, o simulador simplesmente ignora o range definido para dados do tipo *integer*.

As opções de simulação podem ser configuradas em *Flow Navigator -> Settings -> Simullation*. Para habilitar a verificação do *range* de valores inteiros, marque a *checkbox* *xsim.elaborate.rangecheck* localizada na aba *Elaboration* e clique em *Apply*. Com isso o simulador passará a notificar erros se em tempo de simulação algum inteiro ultrapassar os limites de seu *range*.



ANEXO 1

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

use std.env.finish;

entity list1_tb is
end list1_tb;

architecture sim of list1_tb is
    constant clk_hz : integer := 100e6;
    constant clk_period : time := 1 sec / clk_hz;

    signal clk : std_logic := '1';
    signal rst : std_logic := '1';
    signal res : integer;
begin
    clk <= not clk after clk_period / 2;

    DUT : entity work.list1(rtl)
    port map (
        clk => clk,
        rst => rst,
        count => res
    );

    process
    begin
        wait for clk_period * 2;
        rst <= '0';
        wait for clk_period * 1_000_000_000;
        finish;
    end process;
end architecture;
```