Escola Politécnica da Universidade de São Paulo
Departamento de Engenharia de Sistemas Eletrônicos - PSI

PSI-2553- Projeto de Sistemas Integrados

Exp. 1B: Simulações ….. (Teoria)

Objetivos

Esta experiência visa validar o projeto do processador Fibonacci através de 2 simulações:

1. A arquitetura RTL do processador Fibonacci sintetizado e capturado em VHDL na exp1A.
2. O código VHDL comportamental utilizado como ponto de partida deste projeto.

As simulações serão realizadas através do uso de um ***testbench***.

Parte Teórica

Ao longo de um fluxo de projeto, o circuito alvo é representado por diferentes modelos que se tornam cada vez mais detalhados. Os sucessivos modelos vão sendo criados a partir de um modelo inicial, comumente referido por *golden model*. Este é considerado como modelo de referência e correto por construção. Os sucessivos modelos situam-se nos eixos comportamental (módulos funcionais), estrutural (células de biblioteca) e geométrico (desenho do leiaute). Cada modelo precisa ser simulado a fim de se garantir que o circuito final esteja funcionando de acordo com as especificações desejadas.

VHDL- Testbench

O *testbench* é um código VHDL que descreve um conjunto de estímulos a ser aplicado nos portos (terminais) de entrada do (modelo do) circuito a fim de se verificar seu correto funcionamento através da correspondente simulação. O *testbench* pode também efetuar a comparação entre os resultados das simulações e um gabarito, indicando de automaticamente se o modelo do processador está correto ou se contém erros. Desta forma é possível testar cada um dos modelos do circuitos através de um grande número de estímulos de entrada sem a necessidade de se descrever tais estímulos através dos recursos fornecidos pelos simuladores, reduzindo o risco de erros durante as simulações.

Não deve subestimar a importância da elaboração de um *testbench*. Dada a necessidade de se certificar de que um projeto/circuito tenha sido realizado corretamente, é comum que se gaste mais tempo no desenvolvimento de um *testbench* do que no projeto do circuito. A tarefa de verificação funcional, constituída basicamente da construção e execução de *testbenches*, consome até 60-70% dos recursos humanos e materiais dentro do fluxo do projeto nos projetos correntes.

Nesta experiência, os *testbenches* serão escritos em VHDL (outras linguagens também são admissíveis) e as simulações farão uso do Modelsim nos ambientes FPGAdvantage e Quartus.

O modelo básico de um *testbench* é ilustrado na figura 1.

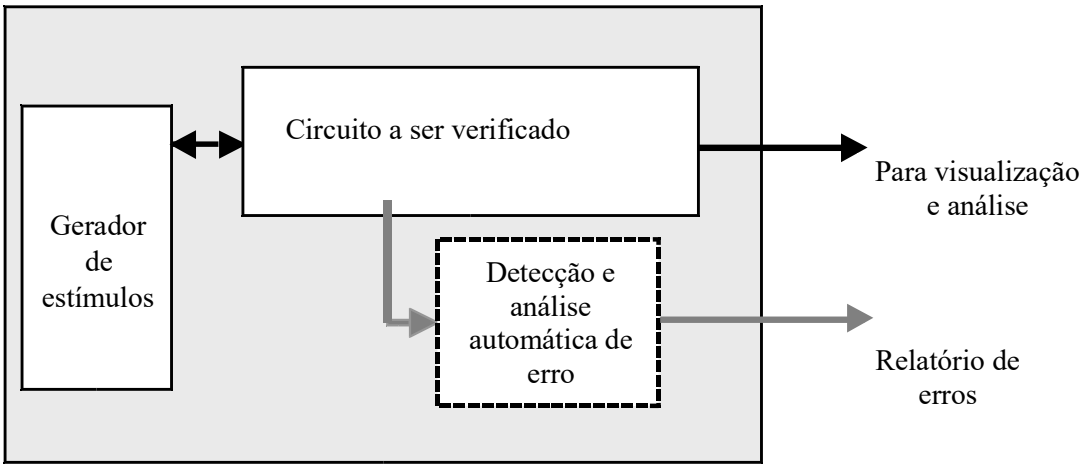

Fig.1: Modelo básico de um *testbench*

O código VHDL do *testbench* é hierárquico. Ele consiste de um módulo topo que contém 2 submódulos: o "*Gerador de estímulos"* e o "*Circuito a ser verificado"* (nesta experiência, o módulo "*Detecção e análise automática de erro*" não será considerado). O conjunto todo é compilado e rodado em simulador, sendo que o resultado pode ser, por exemplo, uma carta de tempos a ser analisada pelo projetista com o intuito de verificar se o funcionamento do circuito foi adequado. O *Gerador de estímulos* representa o ambiente de operação (não exaustiva) do circuito. Desta forma ele deve funcionar como uma FSM que, além de ser responsável pelo envio de estímulos, também recebe sinalizações do circuito sendo testado e realiza mudanças de estado frente a elas.

Apesar de não utilizado nesta experiência, é muito comum que os *testbenches* sejam mais automatizados (em forma de CAD) para que a análise de erros não seja realizada visualmente pelo projetista. Desta forma, costuma ser adicionado um módulo de detecção e análise de erros a partir da simulação realizada e da análise do grau de progressão dos testes. Em geral, tal módulo é realizado em linguagem de verificação (SystemVerilog, C++, SystemC, etc.) diversa do VHDL ou Verilog.

Para explicar como deve ser a comunicação de dados entre o submódulo "*Gerador de estímulos*" e o submódulo "*Circuito a ser verificado*" vamos utilizar o exemplo da figura 2.

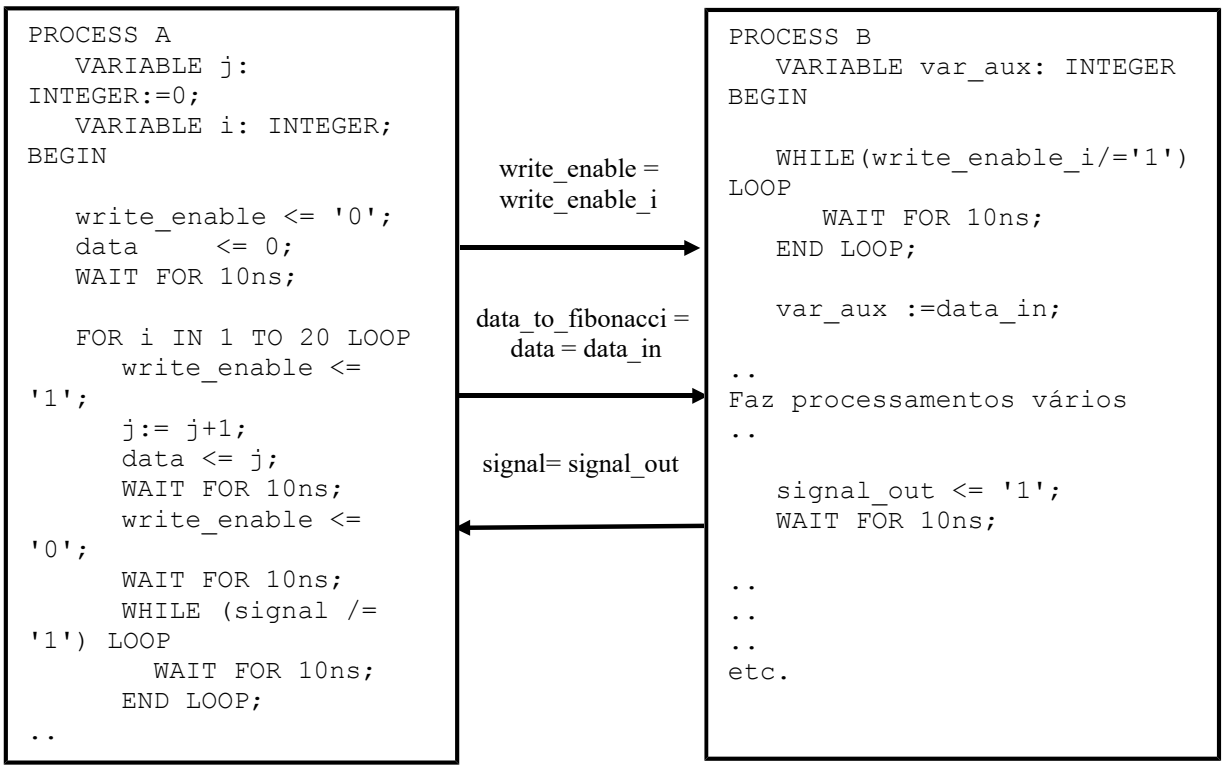

Fig.2: Comunicação entre módulo gerador e módulo projeto

O processo A corresponde ao gerador e o B ao circuito. Ambos os processos são disparados simultaneamente ao se iniciar a execução do *testbench*. Pode-se observar que que, enquanto no processo A, os sinais *write_enable_i* e *data_in* são iniciados no processo B a condição *write_enable_i* =1 é aguardada. Em todos os casos, os WAITs servem para sequencializar os blocos de comandos uma vez que as designações de sinal levam um tempo DELTA para ocorrerem (devido ao modelo de computação referente à linguagem VHDL). No processo A, quando o laço FOR é executado o valor de *data_in* é definido e colocado na saída. No processo B, quando *data_in* é recebido, ele é usado para processamento genérico que pode levar um tempo indefinido para ocorrer. Enquanto isto, o processo entra em estado de espera aguardando o flag *signal_out*.

A figura 3 reproduz a descrição VHDL comportamenbtal do M_F adotada no início deste projeto.

```
ARCHITECTURE behavior_w OF fibonacci IS
  CONSTANT PERIOD : TIME := 10 ns;
BEGIN
        PROCESS
                VARIABLE        n_anterior1                     : NATURAL :=0;
                VARIABLE        n_anterior2                     : NATURAL :=0;
                VARIABLE        n_fibonacci                     : NATURAL :=0;
                VARIABLE        n_max                             : NATURAL :=0;


        BEGIN
                                status_o <= "00";
                                irq_o <= '0';
                                data_o <= 0;
                                WAIT FOR 1*PERIOD;

                                WHILE(write_enable_i/='1') LOOP
                                  WAIT FOR 1*PERIOD;
                                END LOOP;

                                n_anterior1 :=1;
                                n_anterior2 :=0;
                                n_max :=data_in;
                                status_o <= "01";
                                WAIT FOR 1*PERIOD;
                                IF (n_max =0) or (n_max=1) THEN
                                        n_fibonacci:=n_max;
                                ELSE
                                        WHILE (n_max  /= 1) LOOP
                                                n_fibonacci :=n_anterior1+n_anterior2;
                                                n_anterior2 :=n_anterior1;
                                                n_anterior1 :=n_fibonacci;
                                                n_max :=n_max-1;
                                        END LOOP;
                                END IF;

                                data_o <= n_fibonacci;
                                irq_o       <= '1';
                                status_o <= "10";
                                WAIT FOR 1*PERIOD;

                                WHILE (read_enable_i/='1') LOOP
                                  WAIT FOR 1*PERIOD;
                                END LOOP;

                                WAIT FOR 1*PERIOD;

        END PROCESS;
END behavior_w;
```

Fig 3: Algoritmo Fibonacci não-recursivo em VHDL

A Figura 4 ilustra a estrutura geral dos 2 *testbenches* fornecidos. A única diferença entre os dois é a presença dos sinais *clock* e *reset* no *testbench* RTL e a ausência destes dois sinais no *testbench* algorítmico.

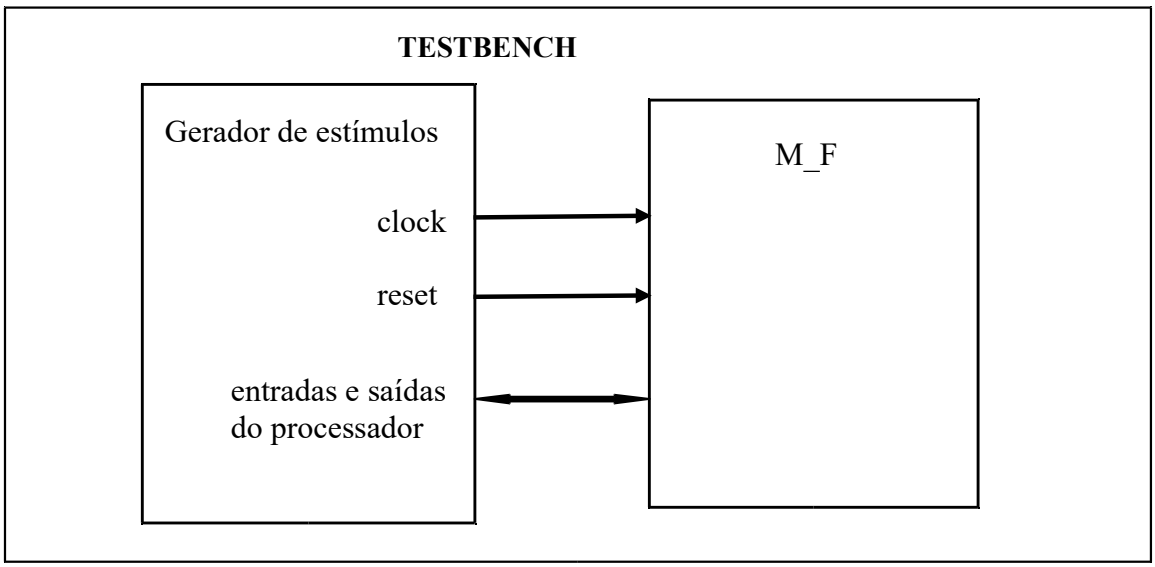

Figura 4 – Testbench para verificação do processador M_F

A figura 5 reproduz os portos da descrição VHDL comportamental do M_F adotada no início deste projeto.

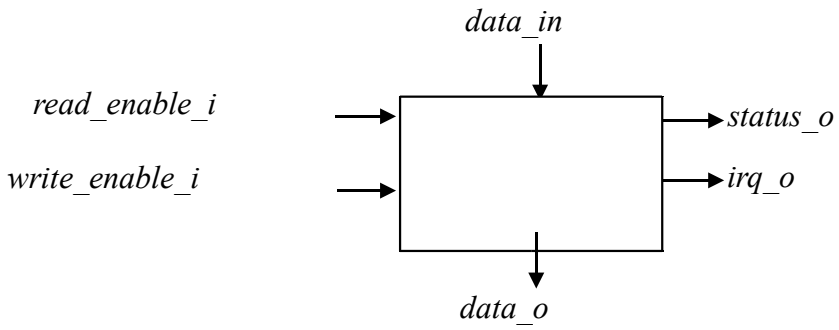

Figura 5. Interfaces do módulo Fibonacci