Laboratório sobre Implementação de Sistemas Digitais com HDLs Ferramentas de Captura e Validação

Prática: Implementação de um Circuito Acumulador

Recursos: Ambiente de Desenvolvimento Active-HDL da Aldec, Inc.

Parte I – Introdução e Objetivos

Os laboratórios anteriores referiam-se às Unidade I e II da disciplina. Eles serviram para dar embasamento na captura de projeto, simulação e síntese de sistemas digitais baseados em diagramas de esquemáticos, diagramas de transição de estados e geradores automáticos de módulos. Também foi exercitada, nestes laboratórios, a síntese física de sistemas digitais visando sua implementação sobre FPGAs e a configuração e teste destes dispositivos sobre plataforma educacionais, em particular sobre a plataforma XS40/XST-1 da empresa Xess, disponível no ambiente da disciplina. O último destes laboratórios visou consolidar os conhecimentos adquiridos, através de um projeto específico. A partir do presente laboratório, inicia-se a Unidade III da disciplina, onde o foco passa a ser uma forma mais abstrata e mais moderna de desenvolver sistemas digitais, qual seja, aquela baseada no emprego de Linguagens de Descrição de Hardware (em inglês, *Hardware Description Languages*, donde deriva o acrônimo HDL).

Os objetivos específicos deste Laboratório são iniciar a utilização do ambiente de simulação Active-HDL, disponível para uso na disciplina, e realizar a simulação de um circuito exemplo, um acumulador de valores de 8 bits.

Parte II - Localização do Ambiente de Desenvolvimento Active-HDL

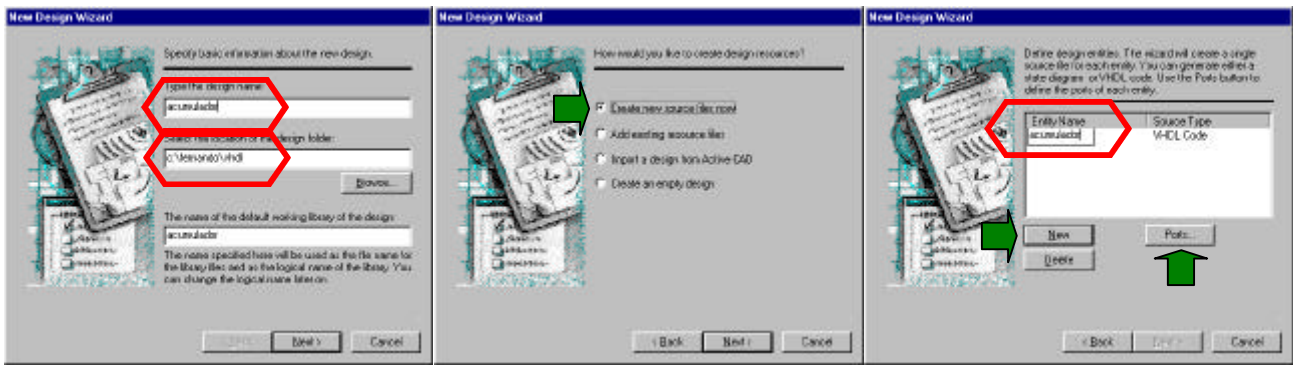
- Verificar se o simulador está instalado. Normalmente, há o ícone do "**Active-HDL 3.5**" no *desktop*, com a seguinte aparência:


Active-HDL 3.5.lnk

- Se o ícone não estiver disponível no seu ambiente de trabalho, verifique em **C:\Program Files\Aldec\Active-HDL 3.5\Bin\avhdl.exe**, ou tente localizá-lo via menu **Start** do Windows.

Parte III - Início do Projeto

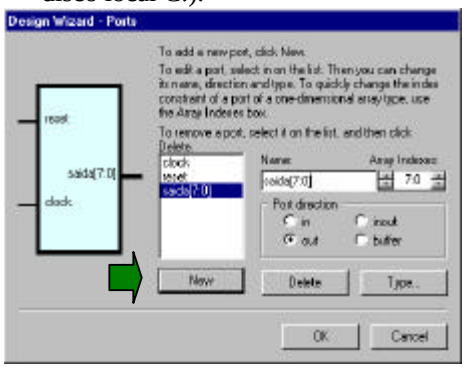
- Crie um novo projeto. Para tanto, na janela de diálogo inicial, escolha o marcador *Create New Design* e clique no botão Ok. Seu projeto deve ter como entradas os sinais de **reset** e **clock**, e como saída um vetor **saída** de 8 bits. O *Design Wizard* é automaticamente iniciado para ajudar na criação do texto que descreve seu projeto em VHDL. A Figura 1 detalha e comenta os passos da interação com o *Design Wizard* do Ambiente Active-HDL 3.5 para criação de um esqueleto de projeto:
- Observar que o arquivo criado já contém a referência à biblioteca IEEE e ao *package* pertinente, onde os tipos *std_logic* estão definidos. Também aparece a definição da *entity* do seu módulo principal, faltando apenas completar a parte referente à arquitetura (*architecture*), cujo cabeçalho já está disponível.

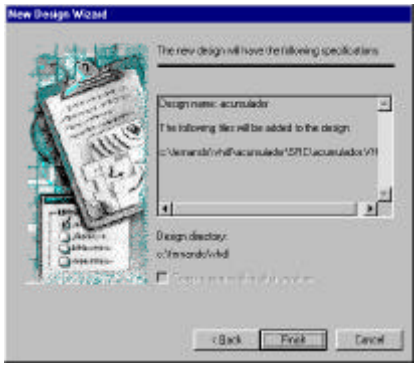


1. Defina o nome do projeto e o diretório onde este será armazenado (Use seu diretório H:/ ou algum diretório no disco local C:).

2. Indique que os fontes devem ser criados pelo *wizard*.

3. Indique o nome do arquivo VDHL e depois entre com o nome e o tipos (seta verde) das entradas/saídas.






```
--
--  File: D:\home\moraes\acumulador\SRC\acumulador.VHD
--  created by Design Wizard: 05/03/99 08:44:19
--
library IEEE;
use IEEE.std_logic_1164.all;
entity acumulador is
   port (
      clock: in STD_LOGIC;
      reset: in STD_LOGIC;
      saida: out STD_LOGIC_VECTOR (7 downto 0)
   );
end acumulador;
architecture acumulador of acumulador is
begin
  -- <<enter your statements here>>
end acumulador;
```

4. Entre com as portas de entrada e saída de seu módulo. No final clique em OK. O resultado será o retorno à janela de arquivos VHDL (item 3). Clique em "Next".

5. Clique em "finish".

6. Após o término da criação do projeto com o Design Wizard volta-se ao ambiente de simulação. No Design Browser, clique duas vezes no nome do arquivo (**acumulador.vhd)**. Será aberto o arquivo acima.

Figura 1 – Resumo de passos da interação com o Design Wizard do ambiente Active-HDL V3.5 para criação de um esqueleto de projeto do sistema digital acumulador.

Parte IV - implementação de um Circuito Acumulador

- Acumuladores são circuitos digitais básicos muitos empregados, compostos de um registrador que guarda resultados de um processo de acumulação e um operador combinacional que realizada as operações parciais cujos resultados são acumulados. Os operadores mais comumente usados são somadores ou multiplicadores ou combinações destes, mas outros são possíveis. No caso, trabalharemos com um somador.
- Um diagrama de blocos sucinto do circuito que deverá ser implementado está mostrado na Figura 2. Os sinais com nomes em caracteres pretos indicam entradas e/ou saídas externas (parte da entity do módulo VHDL), enquanto que os sinais com nome em azul são sinais internos do circuito. O somador e o registrador constituem o sistema acumulador, enquanto que o contador de oito bits é um exemplo muito simples de um gerador de parcelas da soma.

Tarefa 1: **Estude, Pense e Responda**: O circuito completo realiza o quê na sua opinião? Uma resposta possível é que se trata de um circuito capaz de calcular a soma dos **n** primeiros termos de uma **progressão aritmética** de **termo inicial** 0 e **razão** 1. Porquê? Que parte do circuito gera **n**?.

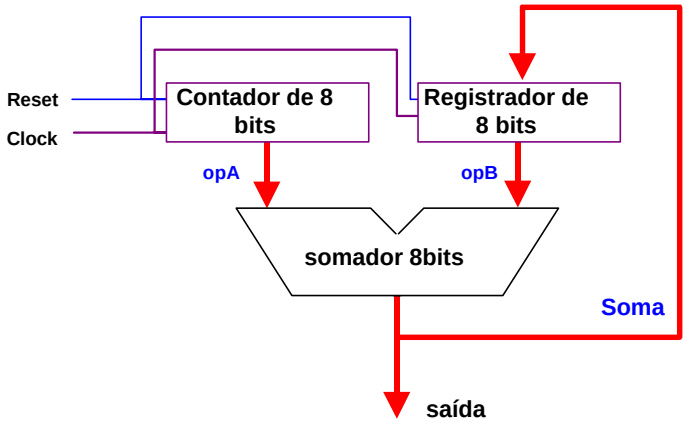


Figura 2 – Diagrama de blocos do sistema digital acumulador.

O projeto deste acumulador será decomposto em 3 partes: a definição dos elementos comuns, o circuito propriamente dito e o gerador de teste ou test bench.

- **Primeira parte do projeto**: *Definição do elementos comuns*. Aqui utilizaremos o conceito de "*package*" da linguagem VHDL, onde serão definitos o tipo *regsize* (vetor de 8 bits) e uma subrotina (*procedure*) para realizar a soma, denominada *somaAB*. O código encontra-se Na Figura 3. Utilizar o *help* da ferramenta para compreender o significado das palavras reservadas de VHDL.

  Importante: toda a descrição do *package* é inserida no início do arquivo, na linha 1, antes dos comentários inseridos pelo *wizard*. Não se esqueça de que a cada nova **design unit** do tipo *package* ou *entity*, é necessário definir as bibliotecas usadas pela entidade em questão. Ou seja, o contexto de uma declaração de uso de uma biblioteca restringe-se à unidade de projeto (design unit) definida imediatamente após a declaração.

```
library IEEE;
use IEEE.Std_Logic_1164.all;

package comum is

    subtype regsize is std_logic_vector(7 downto 0);
    procedure somaAB  (     signal A,B: in  regsize;   signal Cin: in STD_LOGIC;
                            signal S:   out regsize;   signal Cout:out STD_LOGIC);
end comum;

package body comum is

    procedure somaAB  (     signal A,B:  in  regsize;  signal Cin:  in STD_LOGIC;
                            signal S:    out regsize;  signal Cout: out STD_LOGIC) is
    variable carry : STD_LOGIC;
    begin
        for w in 0 to 7 loop
            if w=0 then carry:=Cin;  end if;
            S(w) <= A(w) xor B(w) xor carry;
            carry := (A(w) and B(w)) or (A(w) and carry) or (B(w) and carry);
        end loop;
        Cout <= carry;
    end somaAB;
end comum;
```

Figura 3 – Estado do texto VHDL após a definição dos elementos comuns.

- **Segunda parte do projeto**: *O circuito propriamente dito.* Iniciar logo após o término do package (*end comum*). Acrescentar, às bibliotecas inseridas pelo *Design Wizard*, a biblioteca "IEEE.std_logic_unsigned.all", que habilita o uso do operador '+' entre tipos *std_logic* e

std_logic_vector, a serem utilizados no contador, e o package "*comum*", que conterá o somador e a definição do tipo *regsize*. O texto deve ficar como na Figura 4:

```
library IEEE;                                          } Já inserido pelo wizard
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;                       } Bibliotecas adicionais que devem ser inseridas
use comum.all;

entity acumulador is                                   }
   port (                                              }
      clock: in STD_LOGIC;                             } Já inserido pelo wizard
      reset: in STD_LOGIC;                             }
      saida: out STD_LOGIC_VECTOR (7 downto 0)         }
   );                                                  }
end acumulador;
architecture acumulador of acumulador is ..... } Início do resto do projeto
```

Figura 4 – Estado do texto VHDL após a definição das bibliotecas adicionais.

- Declarar entre a *architecture* e o *begin* os sinais internos do circuito. Para este exemplo serão necessários três barramentos - *opA*, *opB* e *soma* - e dois sinais para o somador - *cin* e *cout*.

```
architecture acumulador of acumulador is
      signal soma, opA, opB : regsize;
      signal cin, cout : std_logic;
begin
```

- Agora deve-se montar o circuito entre o *begin* e o "*end acumulador*". Teremos 4 módulos executando em paralelo: dois registradores, o somador e a conexão da saída do registrador à saída do circuito. Ver Figura 5.

```
cin <= '0';                                            -- somador
s1: somaAB( opA, opB, cin, soma, cout);

saida <= soma;                                         -- conexão da saída do registrador à saída do circuito

process (clock, reset)                  --- registrador incrementador de 8 bits, com entrada e saída opA
begin
   if RESET='1' then   opA <= (others=>'0');
      elsif (clock'event and clock='1') then  opA <= opA + 1;
   end if;
end process;

process (clock, reset)                  ---- registrador de 8 bits, com entrada soma e saída opB
begin
   if RESET='1' then   opB <= (others=>'0');
      elsif (clock'event and clock='1') then  opB <= soma;
   end if;
end process;
```

Figura 5 – Estado do texto VHDL após a definição da arquitetura do acumulador.

Tarefa 2: É possível usar na descrição, ao invés de usar dois processos (*process*), usar apenas um que cria simultaneamente os dois registradores dentro de um só processo, uma vez que ambos são controlados pelos mesmos sinais de clock e o reset. Faça isto.

- **Terceira parte do projeto**: *O gerador de teste ou test bench*. Sua função é gerar os estímulos para o circuito descrito. Para isto, declara-se o circuito implementado (no caso o acumulador) como um componente do test bench, utilizando-o na *architecture* deste último. Depois gera-se o *reset* com uma atribuição e o *clock* com um processo. O processo de geração do test bench está descrito na Figura 6.

```
library IEEE;
use IEEE.std_logic_1164.all;        } Bibliotecas a serem utilizadas
use comum.all;

entity tb is                        -- entidade sem pinos
end tb;

architecture teste of tb is

   component acumulador             -- declara que vai utilizar a entidade acumulador
     port (copiar aqui as portas do acumulador );
   end component;

   signal reset, ck : std_logic;    -- sinais internos
   signal saida : regsize;
begin

  a1:  acumulador port map(clock=>ck, reset=>reset, saida=>saida);    -- instância do acumulador

  reset <= '1' , '0' after 10 ns;   -- gera o reset com atribuição
  process                           -- gera o clock
     begin
         ck <= '1' after 10ns, '0' after 20ns;
         wait for 20ns;
    end process;

end teste;
```

Figura 6 – Conteúdo do arquivo VHDL que descreve o test bench do circuito acumulador.

Tarefa 3: Qual a freqüência gerada pelo processo do teste bench que produz o sinal de clock? Altere o teste bench para produzir uma freqüência de clock de 33,33...MHz e refaça a simulação. Salve ambas as formas de onda de simulação para entrega junto com o projeto.

Parte V - Compilação e Simulação

- Uma vez descrito o test bench, compila-se a descrição até não haverem mais erros. Use a opção Menu Design → Compile All.
- Depois deve-se realizar a simulação. Os passos para isto são:

1. Opção de menu *Simulation* → *Initialize Simulation*. Nesta parte deve-se escolher qual módulo iremos simular. Deve-se escolher o módulo *tb*, pois este é o módulo de mais alto nível na hierarquia.
2. Abrir o diagrama de tempos, clicando no botão relativo a esta função (ver a figura da Parte IV deste texto).
3. Inserir na coluna da esquerda do diagrama de tempos os sinais que se deseja visualizar. O procedimento é simples: seleciona-se o módulo desejado e arrasta-se seu ícone com o mouse apertado para o diagrama de tempos, o que acrescenta todos os sinais na janela de visualização da simulação. Processo similar pode ser usado para mostrar um ou mais sinais individuais.
4. Escolha a opção de menu *Simulation* → *Run For* tantas vezes quantas for necessário para enxergar uma porção significativa do comportamento do circuito.

- Verifique se a simulação está funcionando corretamente, e relate as observações no relatório da aula.

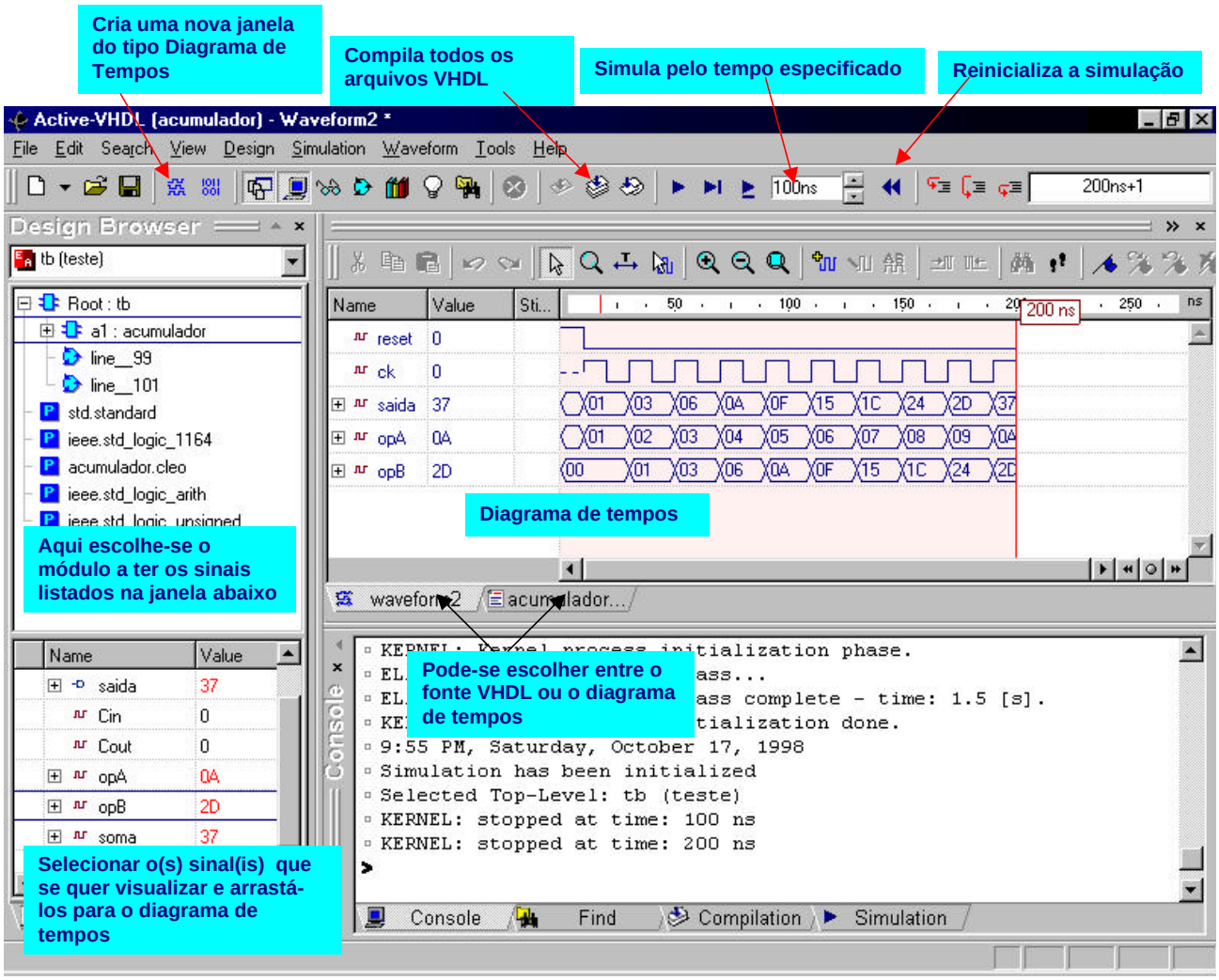
Parte VI - A fazer e a entregar

- Projeto completo, arquivado (opção de menu Design → Archive Design... do Ambiente Active-HDL), com as formas de onda salvas para duas freqüências distintas de simulação (conforme Tarefa 3);
- Relatório com as respostas às questões aqui colocadas dentro das Tarefas;

Percentuais de nota atribuídos às Tarefas:

Item Avaliado	Percentual
Projeto completo e simulações	**40%**
Respostas às questões e tarefas realizadas	**60%**

Apêndice – Visão Geral do Ambiente de Desenvolvimento Active-HDL:



- Para melhorar a compreensão do tema, aconselha-se a realização, fora do horário de aula, de alguns tutoriais disponíveis a partir do ambiente Active-HDL, bem como a análise de documentações específicas. Exemplos importantes são:
  - O tutorial Evita, acessível a partir da opção de menu Help → Interactive VHDL Tutorial do ambiente Active-HDL. Este tutorial também está disponível na área de download da disciplina;
  - Os tutoriais HDL Entry and Simulation Tutorial e VHDL Testbench Tutorial, disponíveis a partir da opção de menu Help → On-Line Documentation;
  - Exemplos selecionados do material denominado Application Notes e Package References, disponíveis a partir do mesmo local que os tutoriais supracitados.