Projeto com Circuitos Reconfiguráveis

Erros comuns em VHDL

Prof. Daniel M. Muñoz Arboleda

FGA - UnB

Esta aula é útil se você costuma falar:

“Funciona em simulação mas não no hardware”

Lembrando

• Processos sem clock atualizam sinais a qualquer momento. Os sinais que disparam o processo precisam estar na lista de sensibilidade.

• Processos com clock só podem atualizar sinais síncronos na borda de clock. Os sinais assincronos, ou seja fora do rising_edge(clk), precisam estar na lista de sensibilidade.

1. Lista de sensibilidade incompleta em processos non-clocked: Todos os sinais que controlam a(s) saída(s) de um processo sem clock devem aparecer na lista de sensibilidade.

Exemplo:

```
process (sel, a, dado)
begin
      a_out <= 0x00;
      data_out <= 0x00;
      if sel='1' then
            a_out <= a;
            data_out <= dado;
      end if;
end process;
```

Se a lista não está completa você pode experimentar a seguintes situações:

a) A simulação não se comporta como esperado (os sinais não se atualizam quando deveriam).
b) Tudo parece funcionar bem em simulação mas não no hardware.

2. Sinal atribuído por múltiplos processos: Cada sinal deve ter uma única fonte. Se o sinal é atribuído por multiplas fontes então o simulador não saberá qual valor atribuir, portanto, o sinal será apresentado como 'X'

Exemplo:

```
process (i1, i2)
begin
    saida <= i1 and i2;
end process;

process (i1, i3)
begin
    saida <= i1 and i3;
end process;
```

Se seu sinal tem mais do que uma fonte (por exemplo atribuído por mais de um processo), você poderá observar:

a) X's atribuídos ao sinal de interesse
b) Uma mensagem de erro no processo de síntese ou na simulação.

3. Esquecer o valor padrão de um sinal atribuído em processo non-clocked: isto pode resultar na inferência de um latch. A ferramenta interpreta que você deseja manter ou armazenar um valor de um sinal no processo non-clocked.

Exemplo: inferindo latch por IF incompleto (falta um valor padrão para out).

```
process (sel)
begin
      if sel='0' then
            saida <= '1';
      end if;
end process;
```

Evite criar registros em processos non-clocked.

Este processo tentará manter (usando latch) o último valor atribuído ao sinal 'saida' quando 'sel' não seja zero.

Forma correta:

```
process (sel)
begin
      saida <= '0'; -- valor padrão
      if sel='0' then
            saida <= '1';
      end if;    -- ou use else para completar o condicional
end process;
```

Se deseja esse tipo de comportamento então use processos com relógio, inferindo um registrador.

4. Atribuir valor padrão de um sinal usando outro sinal do mesmo processo ou de outros processos non-clocked:

Exemplo: neste exemplo o projetista está tentando manter (inferência de latch) o valor de 'saída' no sinal aux.

```
process (sel,saida)
begin
      saida <= '0';
      aux_reg <= saida; - - problema
      if sel='0' then
            aux_reg <= '0'
            saida <= '1';
      end if;
end process;
```

Verifique que em cada branch de um processo non-clocked está efetivamente sendo atribuído um valor aos sinais atualizados.

5. Tentar criar contadores em processos assíncronos.

Contadores são baseados em registradores e, portanto, devem apenas ser criados em processos síncronos. Um contador soma ou subtrai do valor anterior, portanto, é necessário usar um registrador.

```
process (incremento)
begin
      if incremento='1' then
            contador <= contador + 1;
      end if;
end process;
```

Não faça isso! Tal vez funcione em simulação, mas não vai funcionar no hardware. O contador vai incrementar descontroladamente. Também vai inferir um latch.

```
process (incremento,counter)
begin
      if incremento='1' then
            contador <= contador + 1;
      end if;
end process;
```

Isto fica pior! Cada vez que incrementa o processo dispara novamente. Isso cria um loop combinacional e tipicamente causa um erro de simulação. Novamente está inferindo um latch.

5. Tentar criar contadores em processos assíncronos.

```
process (incremento,counter)
begin
      contador <= contador;
      if incremento='1' then
            contador <= contador + 1;
      end if;
end process;
```

Pelamor de Deus!

Similar ao exemplo anterior, mas desta vez o projetista tenta atribuir um valor padrão a ‘contador’ usando ‘contador’ . Também vai inferir latch.

5. Tentar criar contadores em processos assíncronos.

Forma correta:

```
process (clk)
begin
	if rising_edge(clk) then
		if reset='1' then
			contador <= (others=>'0');
		else
			if incremento='1' then
				contador <= contador + 1;
			end if;
		end if;
	end if;
end process;
```

6. Usar um sinal que ainda não foi atualizado: sinais se atualizam de forma concorrente no final do processo. Variáeis se atualizam de forma imediata.

Exemplo:

```
process (clk)
begin
      if rising_edge(clk) then
            if reset='1' then
                  contador <= (others=>'0');
            else
                  a <= ‘1’;
                  if a='1' then
                        contador <= contador + 1;
                  end if;
            end if;
      end if;
end process;
```

Supondo que o valor inicial de a <= ‘0’. O sinal ‘a’ vai se atualizar no final do processo. Somente no próximo ciclo de clock o condicional if ‘a’=’1’ vai se cumprir e o contador será atualizado.

Erros comuns em VHDL