

Nome: Yuri L. M. Lanzini Matrícula: 2221100006

Vídeo 1 - Introdução a Verilog - Visão geral

Nesse vídeo ele fala sobre três temas:

O que é Verilog?

- É uma linguagem de descrição de Hardware. Essa ferramenta suporta a projeção, verificação e implementação de projetos analógicos, digitais e híbridos em vários níveis de abstração.

Síntese X Simulação

A linguagem verilog ela serve tanto para gerar os circuitos reais que seriam uma etapa de síntese através de um modelo verilog, por exemplo fpga um compilador de síntese que pode gerar um Netlist que eventualmente pode virar um circuito. Quanto também possibilita a questão da simulação e isso é uma funcionalidade muito poderosa já que por simulação a gente consegue testar diversos cenários de uma forma bem mais rápida e bem mais barato.

Behaviour X Structural

- Behavior – Descreve somente as funcionalidades entrada/saída. A estrutura interna fica a cargo da ferramenta de síntese.
- Structural – Define as funcionalidades e estruturas internas dos circuitos. Estruturas de hardware são especificadas explicitamente.

Vídeo 2 - Introdução a Verilog – Módulos

Nesse vídeo ele fala sobre os módulos que são as unidades construtivas da linguagem verilog, cada modulo terá um conjunto de entradas, uma lógica de processamento dessas entradas e um conjunto de saídas com os resultados.

Declaração dos módulos

Os modulos verilog seguem sempre a mesma estrutura começam com a palavra chave módulo e terminam com endmodule, logo depois da palavra chave vem o nome do determinado módulo e a sua lista de portas de entrada e saída ou bidirecional, cada porta pode ser três tipos: input, output, inout.

Data Types

Net- Representa conexoes físicas entre componentes.

Variable – Elemento para armazenamento temporário de dados.

Arrays

Declaração de Arrays:

- <DataType> [a:b] nome [c:d]

Atribuições

Atrib. Contínua(assing)

-É sempre ativa.

Atrib. Procedural (always)

- Ativa de acordo com a lista de sensibilidade.

Blocking

- Executada linha a linha, na sequência.

Nonblocking

- O lado direito de todas as atribuições são avaliados no mesmo instante

Constantes Numéricas

<tamanho(quant. de BITS)>'<base numérica><valor da constante>

Vídeo 3 - Introdução a Verilog – Instanciação

Instanciação de Módulos

- Sempre instanciar pelo nome das portas.

```
<"tipo_do_modulo">  
<nome_da_Instância>  
(.nome_da_porta1(fio1_conectado),  
.nome_da_porta2(fio2_conectado),...,  
.nome_da_porta_n(fio_n_conectado));
```

- A ordem das portas não importa.

Vídeo 4 - Introdução a Verilog – Operadores

Nesse vídeo ele mostra e explica os operadores e o controle de fluxo segue abaixo as imagens com a funcionalidade e exemplos:

Operadores

Operator Symbol	Functionality	Examples
+	Add. Positive	ain = 5 ; bin = 10 ; cin = 2'b01 ; din = 2'b0z
-	Subtract. Negate	bin - cin => 9 -bin => -10 ain + din => x
*	Multiply	ain * bin => 50
/	Divide	bin / ain => 2
%	Modulus	bin % ain => 0
**	Exponent*	ain ** 2 => 25

Operator Symbol	Functionality	Examples
~	Invert each bit	~ain => 3b'010 ~cin => 3'b10x
&	AND each bit	ain & bin => 3'b100 bin & cin => 3'b010
	OR each bit	ain bin => 3'b111 bin cin => 3'b11x
^	XOR each bit	ain ^ bin => 3'b011 bin ^ cin => 3'b10x
^~ or ~^	XNOR each bit	ain ^~ bin => 3'b100 bin ^~ cin => 3'b01x

Operadores

Operator Symbol	Functionality	Examples
>	Greater than	ain > bin => 1'b0 bin > cin => 1'bx
<	Less than	ain < bin => 1'b1 bin < cin => 1'bx
>=	Greater than or equal to	ain >= bin => 1'b0 bin >= cin => 1'bx
<=	Less than or equal to	ain <= bin => 1'b1 bin <= cin => 1'bx

Operator Symbol	Functionality	Examples
==	Equality	ain == bin => 1'b0 cin == cin => 1'bx
!=	Inequality	ain != bin => 1'b1 cin != cin => 1'bx
===	Case equality	ain === bin => 1'b0 cin === cin => 1'b1
!==	Case inequality	ain !== bin => 1'b1 cin !== cin => 1'b0

Operadores

Operator Symbol	Functionality	Examples
!	Expression not true	!ain => 1'b0 !bin => 1'b1 !cin => 1'bx
&&	AND of two expressions	ain && bin => 1'b0 bin && cin => 1'bx
	OR of two expressions	ain bin => 1'b1 bin cin => 1'bx

Operator Symbol	Functionality	Examples
&	AND all bits	&ain => 1'b0 &bin => 1'b0 &cin => 1'bx
~&	NAND all bits	~&ain => 1'b1 ~&bin => 1'b1 ~&cin => 1'bx
	OR all bits	ain => 1'b1 bin => 1'b0 cin => 1'b1
~	NOR all bits	~ ain => 1'b0 ~ bin => 1'b0 ~ cin => 1'b0
^	XOR all bits	^ain => 1'b0 ^bin => 1'bx ^cin => 1'bx
^~ or ~^	XNOR all bits	~^ain => 1'b1 ~^bin => 1'bx ~^cin => 1'bx

Operadores

Operator Symbol	Functionality	Examples
<<	Logical shift left	ain << 2 => 3'b100 bin << 2 => 3'bx00
>>	Logical shift right	ain >> 2 => 3'b001 bin >> 2 => 3'b000
<<<	Arithmetic shift left	ain <<< 2 => 3'b100 bin <<< 2 => 3'bx00
>>>	Arithmetic shift right	ain >>> 2 => 3'b111 (signed) bin >>> 2 => 3'b000 (signed)

Operator Symbol	Functionality	Format & Examples
?:	Conditional test	(condition) ? true_value : false_value ag_out = (sel == 2'b01) ? a : b
{ }	Concatenate	ain = 3'b010 ; bin = 3'b110 {ain bin} => 6'b010110
{ () }	Replicate	{3 (2'b101)} => 9'b101101101

Controle de Fluxo

Controle de fluxo

• If - else

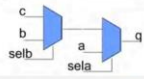
■ Format:

```
if <condition1>
  (sequence of statement(s))
else if <condition2>
  (sequence of statement(s))
...
else
  (sequence of statement(s))
```

■ Example:

```
always @* begin
  if (sel)
    q = a;
  else if (selb)
    q = b;
  else
    q = c;
end
```

Sempre dentro de um Always



Controle de fluxo

• Case


■ Format:

```
case (expression);
  <condition1> :
    (sequence of statements)
  <condition2> :
    (sequence of statements)
  ...
  default : -- (optional)
    (sequence of statements)
endcase
```

■ Example:

```
always @* begin
  case (sel)
    2'b00 : q = a;
    2'b01 : q = b;
    2'b10 : q = c;
    default : q = d;
  endcase
end
```

Sempre dentro de um Always



Controle de fluxo

• Repeat

■ repeat loop - executes a fixed number of times

```
if (rotate == 1)
  repeat (8) begin
    tmp = data[15];
    data = {data << 1, tmp};
  end
```

Sempre dentro de um Always

Repeats a rotate operation 8 times

Vídeo 5 - Introdução a Verilog - Primeiro projeto Quartus II

Nesse vídeo ele ensina a baixar o Quartus II, fazer as configurações iniciais, utiliza a placa Altera DE2 e passa três exemplos.

Vídeo 6 - Simulação no Quartus II – Modelsim

Nesse vídeo ele ensina como simular em verilog utilizando a ferramenta Modelsim no Quartus II, utiliza um módulo de Mux 4x1.