UFPE – Universidade Federal de Pernambuco
CIn – Centro de Informática

Tutorial Verilog

1ª Edição – 2011.1

Felipe de Assis Souza (fas5)

INTRODUÇÃO

Resolvi elaborar este documento após pagar a cadeira de Sistemas Digitais em 2011.1, já que a maior dificuldade que os alunos (me incluindo) sentiram durante o projeto da 2º unidade, foi em entender a sintaxe e semântica da linguagem Verilog.

O objetivo deste tutorial é esclarecer algumas dúvidas a respeito da linguagem, utilizada no projeto da 2ª Unidade de Sistemas Digitais, além de servir como revisão para a cadeira de Infraestrura de Hardware, onde ela será trabalhada mais a fundo.

Aqui você encontrará a explicação das principais *keywords* além de exemplos de como utilizá-las, tentei explicá-las de uma maneira menos formal para facilitar o entendimento. E não, você não vai encontrar o código entregue de bandeja neste tutorial (apesar de dar para pescar alguma coisa). A lógica é com você, ou como diz um amigo meu: “Dá teus ‘pulo’, boy!”.

fas5.

1. PRINCIPAIS *KEYWORDS*

1) module/endmodule – é a mesma coisa que método (ou função) em outras linguagens.
OBS.: *Um module sempre é encerrado com um endmodule.*

Sintaxe e Semântica:
module nome_do_modulo (input 1, input N, ..., output1, ..., outputN);

Ex1.: **module SomaSub (In1, In2, Seletor, Overflow, Out);** *// Início do módulo SomaSub*
(...)
endmodule *// Fim do módulo SomaSub*
In1, In2, Seletor = entradas.
Overflow, Out = saídas.

Ex2.: **module MaiorQue (A, B, Status);** *// Início do módulo MaiorQue*
(...)
endmodule *// Fim do módulo MaiorQue*
A, B = entradas.
Status = saída.

2) input – usado para declarar uma entrada.

Sintaxe e Semântica:
input nome_da_entrada;

Ex1.: **input In1, In2;** *// Declarando duas entradas, In1 e In2, de 1 bit*

Ex2.: **input [3:0] A;** *// Declarando uma entrada, A, de 4 bits*
input [3:0] B; *// Declarando uma entrada, B, de 4 bits*

Ex3.: **input [7:0] A, B;** *// Declarando duas entradas, A e B, de 8 bits*
input Clock; *// Declarando uma entrada, Clock, de 1 bit*

3) output – usado para declarar uma saída.

Sintaxe e Semântica:
output nome_da_saida;

Ex1.: **output Out;** *// Declarando uma saída, Out, de 1 bit*

Ex2.: **output [2:0] Out;** *// Declarando uma saída, Out, de 3 bits*
output Overflow, Status; *// Declarando duas saídas, Overflow e Status, de 1 bit*

4) reg – usado para quando se deseja guardar um valor em uma entrada ou saída até que outro valor seja enviado para essa entrada ou saída (registrador).
OBS.: *Entradas e saídas usadas dentro de blocos always/initial devem ser do tipo reg.*

Sintaxe e Semântica:
reg nome_da_entrada/saida;

```
Ex1.:   reg In1, In2, Out;          // Setando In1, In2 e Out como três registradores de 1 bit

Ex2.:   reg [2:0] Tx, Ty, Tz;       // Setando Tx, Ty e Tz como três registradores de 3 bits
        reg [1:0] Tula;             // Setando Tula como um registrador de 2 bits
        reg Conta, Reset;           // Setando Conta e Reset como dois registradores de 1 bit
```

5) wire – é o tipo padrão das entradas e saídas.
OBS.: *Não se pode usar entradas/saídas do tipo wire dentro de blocos always/initial.*

Sintaxe e Semântica:
wire nome_da_entrada/saida;

```
Ex.:    wire [3:0] Outtemp;         // Declarando um fio, OutTemp, que suporta 4 bits.
```

6) assign – usado para atribuir um valor a uma entrada ou saída.

Sintaxe e Semântica:
assign nome_da_entrada/saida = valor;

```
Ex1.:   input [3:0] In1;            // Declarando uma entrada, In1, de 4 bits
        (...)
        assign In1 = 4'b1001;       // In1 recebe a cadeia de 4 bits 1001.
        (...)

Ex2.:   input [7:0] In1, In2;       // Declarando duas entradas, In1 e In2, de 8 bits
        output Status;              // Declarando uma saída, Status, de 1 bit
        (...)
        assign Status = (In1 > In2);  /* Status recebe 1, se In1 for maior do que In2,
        (...)                          *  caso contrário, recebe 0 */

Ex3.:   input [3:0] In1, In2;       // Declarando duas entradas, In1 e In2, de 4 bits
        input [1:0] Sel;            // Declarando uma entrada, Sel, de 2 bits
        output [3:0] Out;           // Declarando uma saída, Out, de 4 bits
        (...)
        assign Out = (Sel == 2'b01)? In1: In2;   /* Se Sel for igual à cadeia de 2 bits 01, Out
        (...)                                     * recebe In1, caso contrário, recebe In2 */
```

7) begin/end – utilizado para iniciar uma sequência de comandos. Quando há apenas um comando a ser executado, não é necessária a utilização de begin/end, mas é recomendada, para melhorar a legibilidade do código.

Sintaxe e Semântica:

```
begin
  comando1;
  comando2;
  (...)
  comandoN;
end
```

Ex.: exemplos no próximo item.

8) initial – usado para inicializar inputs e/ou outputs.
OBS.: *Como o objetivo de inicializar inputs e outputs é fazer com que eles guardem algum valor, eles também têm que ser declarados como reg.*

Sintaxe e Semântica:

```
initial begin
  Saida1 = dado1;
  Saida2 = dado2;
  (...)
end
```

Ex1.:

```
output [3:0] Out;          // Declarando uma saída, Out, de 4 bits
reg [3:0] Out;             // Setando Out como registrador de 4 bits
(...)
initial Out <= 4'b0000;    // Inicializando Out com a cadeia de 4 bits 0000
(...)
```

Ex2.:

```
input [3:0] In1;           // Declarando uma entrada, In1, de 4 bits
input Sel;                 // Declarando uma entrada, Sel, de 1 bit
reg [3:0] In1;             // Setando In1 como registrador de 4 bits
reg Sel;                   // Setando Sel como registrador de 1 bit
(...)
initial begin              // Início do bloco
  In1 = 4b'1010;           // Inicializando In1 com a cadeia de 4 bits 1010
  Sel = 1;                 // Inicializando Sel com o bit 1
end                        // Fim do bloco
(...)
```

9) always – utilizado para realizar instruções sempre que houver mudança em alguma entrada/saída pré-determinada.

OBS.: posegde = o bloco always executará na descida do clock.
negedge = o bloco always executará na subida do clock.

Sintaxe e Semântica:

```
always @ (entrada1 or ... or entradaN) begin
  (...)
  código;
  (...)
end
```

ou

```
always @ (posedge Clock) begin
  (...)
  código;
  (...)
end
```

Ex1.:

```
input [2:0] In1, In2, In3;        // Declarando três entradas, In1, In2 e In3, de 2 bits
(...)
always (In1 or In2 or In3) begin  /* Início do bloco. Sempre que In1, In2 ou In3
  (...)                            * mudarem de valor, o código dentro do bloco
  código;                          * será executado */
  (...)
end                               // Fim do bloco
```

Ex2.:

```
input Clock;                      // Declarando uma entrada, Clock, de 1 bit
input [3:0] In1, In2;             // Declarando duas entradas, In1 e In2, de 4 bits
(...)
always @(posedge Clock) begin     /* Início do bloco. Na descida do clock, o
  (...)                            * código dentro do bloco será executado */
  código;
  (...)
end                               // Fim do bloco
```

OBS.: Se ao invés de posedge estivesse escrito negedge, o código dentro do bloco seria executado na subida do clock.

10) case/endcase – bloco condicional, semelhante ao bloco case em Java.

OBS.: A keyword default é usada para descrever uma ação padrão, ou seja, caso a entrada não seja nenhuma das previstas pelo programador, um código "padrão" também definido pelo programador será executado. Seu uso não é não é obrigatório.

Sintaxe e Semântica:

```
case (entrada)
    possivel_entrada 1: begin
                            (...)
                            código;
                        end
    (...)
    possivel_entrada N: begin
                            (...)
                            código;
                        end
    default: begin
                (...)
                código;
             end
endcase
```

```
Ex1.:   input [1:0] In;              // Declarando uma entrada, In, de 2 bits
        output [3:0] Out;            // Declarando uma saída, Out, de 4 bits
        (...)
        case (In)                    // Início do bloco
           2'b00: Out <= 4'b0010;    // Caso In seja igual a 00, Out receberá 0010
           2'b01: Out <= 4'b1010;    // Caso In seja igual a 01, Out receberá 1010
           2'b10: Out <= 4'b0110;    // Caso In seja igual a 10, Out receberá 0110
           2'b11: Out <= 4'b0111;    // Caso In seja igual a 11, Out receberá 0111
        endcase                      // Fim do bloco

Ex2.:   input [1:0] In;              // Declarando uma entrada, In, de 2 bits
        input Sel;                   // Declarando uma entrada, Sel, de 1 bit
        output [3:0] Out;            // Declarando uma saída, Out, de 4 bits
        (...)
        case (In)                    // Início do bloco
           2'b00: Out <= 4'b0010;    // Caso In seja igual a 00, Out receberá 0010
           2'b01: Out <= 4'b1010;    // Caso In seja igual a 01, Out receberá 1010
           2'b10: begin              /* Caso In seja igual a 10:
                     Out <= 4'b0110;  * Out receberá 0110 e
                     Sel <= 1;        * Sel receberá 1 */
                  end
           default: Out <= 4'b0111;  // Caso In seja igual a 11, Out receberá 0111
        endcase                      // Fim do bloco
```

11) if/else – bloco condicional semelhante ao bloco if/else de Java. Só pode ser utilizado dentro de blocos always.

Sintaxe e Semântica:

```
if (condicao) begin        ou        assign Saida = (Entrada == cadeia_de_bits) ? Opcao1: Opcao2;
   (...)
   código
   (...)
end else begin
   (...)
end
```

```
Ex1.:   input [3:0] A, B;                  // Declarando uma entrada, In, de 4 bits
        output [3:0] Out;                  // Declarando uma saída, Out, de 4 bits
        (...)
        if (A > B) begin                   /* Início do bloco. Se A for maior do que B:
                Out <= A - B;               * Out recebe o valor da operação A – B,
        end else begin                      * caso contrário:
                Out <= B - A;               * Out recebe o valor da operação B - A */
        end                                // Fim do bloco
        (...)
```

```
Ex2.:   input [3:0] In;                          // Declarando uma entrada, In, de 4 bits
        input [1:0] Sel;                         // Declarando uma entrada, Sel, de 2 bits
        output [3:0] Out;                        // Declarando uma saída, Out, de 4 bits
        wire [3:0] OutTemp;                      // Declarando um fio que suporta 4 bits
        (...)
        assign Out = (Sel == 2'b11)? In: OutTemp;    /* Caso Sel seja igual a 11, Out recebrá
        (...)                                         * o mesmo valor de In1, caso contrário,
                                                      * receberá o valor de OutTemp */
```

12) parameter – usado para rotular dados que serão utilizados muitas vezes, contribuindo para a legibilidade do código.

Sintaxe e Semântica:
parameter ROTULO = cadeia de bits;

```
Ex.:    input [3:0] Instrucao;                  // Declarando uma entrada, Instrucao, de 4 bits
        (...)
        parameter CLRLD = 4b'0000;    // Rotulando a cadeia de 4 bits 0000 como CLRLD
        parameter ADDLD = 4b'0001;    // Rotulando a cadeia de 4 bits 0001 como ADDLD
        parameter HOLD = 4b'0010;     // Rotulando a cadeia de 4 bits 0010 como HOLD
        (...)
        case (Instrucao)                        // Início de um bloco do tipo 'case'
                CLRLD: begin                    /* Caso Instrucao seja igual a 0000, execute
                            (...)                * até o próximo 'end' */
                       end
                ADDLD: begin                    /* Caso Instrucao seja igual a 0001, execute
                            (...)                * até o próximo 'end' */
                       end
                HOLD:  begin                    /* Caso Instrucao seja igual a 0010, execute
                            (...)                * até o próximo 'end' */
                       end
                (...)
        endcase                                 // Fim do bloco
        (...)
```

OBS.: *As cadeias 0000, 0001, 0010 foram rotuladas como CLRLD, ADDLD, HOLD, respectivamente. Sempre que for preciso usar essas cadeias no código, não será necessário digitá-las novamente, podendo substituí-las por seus respectivos rótulos.*

FONTES

http://www.verilogtutorial.info/

http://www.asic-world.com/verilog/index.html