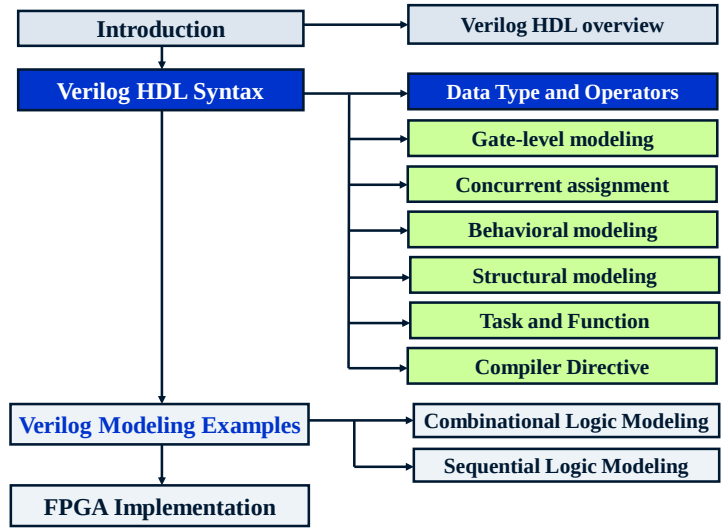
자료형과 연산자

Kyung-Wook Shin
kwshin@kumoh.ac.kr

School of Electronic Eng.,
Kumoh National Institute of Technology

Learning Map



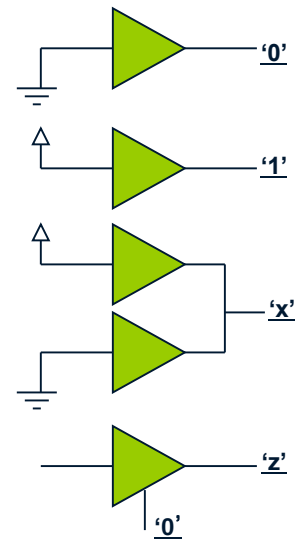
2.1.1 Verilog의 논리값

표 2.1 Verilog의 논리값 집합

논리값	의 미
0	logic zero, or false condition
1	logic one, or true condition
x	unknown logic value
z	high - impedance state

2.1.1 Verilog의 논리값

- Zero, low, false, logic low, ground, VSS
- One, high, true, logic high, power, VDD, VCC
- X, unknown : occurs at logical conflict which cannot be resolved
- HiZ, high impedence, tri-stated, disabled or disconnected driver



2.1.2 Verilog HDL의 자료형

- Net 자료형 : 소자간의 물리적인 연결을 추상화
  - wire, tri, wand, wor, triand, trior, supply0, supply1, tri0, tri1, trireg
  - Default 자료형 ; 1비트의 wire
- Variable 자료형 : 절차형 할당문 사이의 값의 임시 저장
  - 프로그래밍 언어의 variable과 유사한 개념
  - reg, integer, real, time, realtime

net 자료형과 variable 자료형의 할당 모드

자료형	할당 모드				
	게이트 프리미티브 출력	연속 할당문	절차형 할당문	assign… deassign PCA	force… release PCA
Net	Yes	Yes	No	No	Yes
Variable	Comb (No) Seq (Yes)	No	Yes	Yes	Yes

* PCA : Procedural Continuous Assignment

2.1.2 net 자료형

- net 자료형
  - 논리 게이트, 모듈 등의 하드웨어 요소들 사이의 물리적 연결을 나타내기 위해 사용
  - 연속 할당문(continuous assignment), 게이트 프리미티브 등과 같은 구동자(driver)의 값에 의해 net의 값이 연속적으로 유지됨
    - 값을 저장하지 않음 (단, trireg net는 예외)
  - 구동자가 연결되지 않으면, default 값인 high-impedance (z)가 됨
    - 단, trireg net는 이전에 구동된 값을 유지
  - default 자료형은 1비트의 wire
  - default 초기값은 z

2.1.2 net 자료형

표 2.2 Verilog net 자료형

자료형 이름	의 미
wire	함축된 논리적 동작이나 기능을 갖지 않는 단순한 연결을 위한 net
tri	함축된 논리적 동작이나 기능을 갖지 않는 단순한 연결을 위한 net이며, 하드웨어에서 3상태(tri - state)가 되는 점이 wire와 다름. Multiple driving sources가 허용됨
wand	다중 구동자를 갖는 net이며, 'wired-and'(즉, open collector logic)의 하드웨어 구현을 모델링하기 위해 사용
wor	다중 구동자를 갖는 net이며, 'wired-or'(즉, emitter coupled logic)의 하드웨어 구현을 모델링하기 위해 사용
triand	wand와 동일하게 다중 구동자를 갖는 net이며, 하드웨어에서 3상태(tri-state)를 갖는 점이 다름
trior	wor와 동일하게 다중 구동자를 갖는 net이며, 하드웨어에서 3상태(tri-state)를 갖는 점이 다름
supply0	회로접지(circuit ground)에 연결되는 net
supply1	전원(power supply)에 연결되는 net
tri0	저항성 pulldown (resistive pulldown)에 의해 접지로 연결되는 net
tri1	저항성 pullup (resistive pullup)에 의해 전원으로 연결되는 net
trireg	물리적인 net에 저장되는 전하를 모델링하는 net

2.1.2 net 자료형

예 2.1.1 net 자료형 선언의 예

```
wire w1, w2;        // 1비트 wire 자료형은 선언을 생략할 수 있음
wire [7:0] bus;     // a 8-bit bus
wire enable=1'b0; // wire with initial value of 0
wand w3;            // a scalar net of type wand
tri [15:0] busa;  // a three-state 16-bit bus
```

Verilog HDL 자료형과 연산자 K. W. SHIN

2.1.2 net 자료형

- wire와 tri
  - 회로 구성요소들 사이의 연결에 사용
  - wire : 단일 게이트 또는 단일 연속 할당문에 의해 구동되는 net에 사용
    모듈 인스턴스의 포트에 연결되는 신호
  - tri : 3상태 net에 사용. Multiple driving sources가 허용됨

표 2.3 wire, tri net의 진리표

wire/tri	0	1	x	z
0	0	x	x	0
1	x	1	x	1
x	x	x	x	x
z	0	1	x	z

2.1.2 net 자료형

- wired net
  - 다중 구동자를 갖는 설계를 지원하기 위해 사용

표 2.4 wand, triand net의 진리표

wand/ triand	0	1	x	z
0	0	0	0	0
1	0	1	x	1
x	0	x	x	x
z	0	1	x	z

표 2.5 wor, trior net의 진리표

wor/ trior	0	1	x	z
0	0	1	x	0
1	1	1	1	1
x	x	1	x	x
z	0	1	x	z

2.1.2 net 자료형

예 2.1.2

코드 2.1

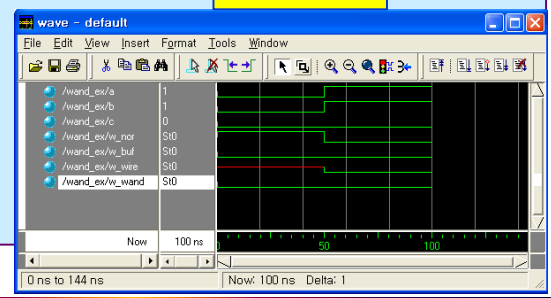
```
module wand_ex ;
  reg  a, b, c;
  wire w_nor, w_buf, w_wire;
  wand w_wand;

  initial  begin
           a=0; b=0; c=0;
       #50 a=1; b=1;
  end
  nor(w_nor, a, b);
  buf(w_buf, c);

  nor(w_wire, a, b);
  buf(w_wire, c);

  nor(w_wand, a, b);
  buf(w_wand, c);
endmodule
```

시뮬레이션 결과



2.1.3 variable 자료형

- variable 자료형 ; reg, integer, real, time, realtime
  - 절차적 할당문(procedural assignment)의 실행에 의해 그 값이 바뀌며, 할당에서부터 다음 할당까지 값을 유지
  - default 초기값
    - reg, time, integer 자료형 : x (unknown)
    - real, realtime 자료형 : 0.0
  - variable이 음의 값을 할당 받는 경우,
    - signed reg, integer, real, realtime 자료형 : 부호를 유지
    - unsigned reg, time 자료형 : unsigned 값으로 취급

2.1.3 variable 자료형

- reg
  - 절차적 할당문에 의해 값을 받는 객체의 자료형
  - 할당 사이의 값을 유지
  - 하드웨어 레지스터를 모델링하기 위해 사용될 수 있음
    - edge – sensitive (플립플롭 등)와 level – sensitive (래치 등)의 저장소자들을 모델링할 수 있음
    - reg는 조합논리회로의 모델링에도 사용되므로, reg가 항상 하드웨어적인 저장소자를 의미하지는 않음

예 2.1.5 reg 자료형 선언의 예

```
reg a;          // a scalar reg
reg[3:0] v;     // a 4-bit vector reg made up of(from most to
                // least significant) v[3], v[2], v[1], and v[0]
reg signed [3:0] signed_reg; // a 4-bit vector in range −8 to 7
reg [-1:4] b;                // a 6-bit vector reg
reg [4:0] x, y, z;           // declares three 5-bit regs
```


2.1.3 variable 자료형


```
module dff (clk, d, qout);
  input  d ,clk;
  output qout;
  reg    qout;

  always @(posedge clk) begin
     qout <= d;
  end
endmodule
```

D 플립플롭

```
module mux21_if(a, b, sel, out);
   input  [1:0] a, b;
   input        sel;
   output [1:0] out;
   reg    [1:0] out;

   always @(a or b or sel) begin
      if(sel == 1'b0)
         out = a;
      else
         out = b;
   end
endmodule
```

2 : 1 MUX

[코드 2.2] reg 자료형 선언이 사용된 모델링 예

2.1.3 variable 자료형

- integer 자료형
  - 정수형 값을 취급하며, 절차적 할당문에 의해 값이 변경됨
  - signed reg로 취급되며, 연산 결과는 2의 보수가 됨
- time 자료형
  - 시뮬레이션 시간을 처리하거나 저장하기 위해 사용됨
  - 64비트의 reg와 동일하게 작용
  - unsigned 값이고 unsigned 연산이 이루어짐
- real, reatime 자료형
  - 실수형 값을 취급

예 2.1.6 Variable 선언 예

```
integer a;          // integer value
time last_chng;     // time value
real float ;        // a variable to store real value
realtime rtime ;    // a variable to store time as a real value
```


2.1.4 벡터

- 벡터
  - 범위지정 [msb:lsb] 을 갖는 다중 비트의 net 또는 reg 자료형
  - signed로 선언되거나 signed로 선언된 포트에 연결되는 경우를 제외하고는 unsigned로 취급
  - 단일 할당문으로 값을 받을 수 있음

```
data_type [msb:lsb] identifier;
```

예 2.1.7 벡터 선언의 예

```
reg  [7:0]  rega;   // 8-bit reg
wire [15:0] d_out;  // 16-bit wire
```

2.1.5 배열

- 배열
  - 별도의 자료형이 없으며, reg 또는 wire 선언을 이용하여 선언
  - 배열 전체 또는 일부분은 단일 할당문에 의해 값을 할당 받을 수 없으며, 또한 수식에 사용될 수 없음
    - 배열을 구성하는 element만 단일 할당문으로 값을 할당 받을 수 있음
  - RAM, ROM, Register File 등의 메모리 모델링에 사용

```
data_type identifier [Uaddr:Laddr][Uaddr2:Laddr2];
```

```
data_type [msb:lsb] identifier [Uaddr:Laddr][Uaddr2:Laddr2];
                                  행(row)        열(column)
```


2.1.5 배열

예 2.1.8 배열 선언

```
reg [7:0] mema[0:255];  // a memory mema of 256 8-bit registers
reg arrayb[7:0][0:255]; // a 2-D array of 1-bit registers
wire w_array[7:0][5:0]; // an array of wires
integer inta[1:64];     // an array of 64 integer values
time chng_hist[1:1000]; // an array of 1000 time values
```

예 2.1.9 배열요소에 대한 할당

```
mema = 0;           // Illegal syntax- Attempt to write to entire array
arrayb[1] = 0;      // Illegal Syntax - Attempt to write to elements
                    // [1][0], [1][1], [1][2], ...[1][255]
arrayb[1][12:31] = 0;    // Illegal Syntax - Attempt to write to
                         // elements [1][12], [1][13], ...[1][31]
mema[1] = 0;             // Assigns 0 to the second element of mema
arrayb[1][0] = 0;    // Assigns 0 to the bit referenced by indices [1][0]
inta[4] = 33559;     // Assign decimal number to integer in array
chng_hist[t_index] = $time; // Assign current simulation time to
                            // element addressed by integer index
```

2.1.5 배열

- 메모리
  - reg형 요소를 갖는 1차원 배열
  - 메모리 전체가 단일 할당문으로 값을 할당 받을 수 없음
    - 인덱스로 지정되는 워드 단위로만 값을 할당하거나 수식에 사용될 수 있음

예 2.1.10 메모리

```
reg [1:n] rega;   // An n-bit register
reg mema [1:n];   // A memory of n 1-bit registers
```


2.1.6 parameter

- parameter
  - variable 또는 net 범주에 속하지 않는 상수값
  - 회로의 비트 크기 또는 지연값을 지정하기 위해 사용
  - defparam 문 또는 모듈 인스턴스 문의 parameter overriding에 의해 값을 변경시킬 수 있음
  - 자료형과 범위지정을 가질 수 있음
    - 범위가 지정되지 않은 경우, 상수 값에 적합한 크기의 비트 폭을 default로 가짐

예 2.1.11 parameter 선언

```
parameter msb = 7;          // defines msb as a constant value 7
parameter e = 25, f = 9;    // defines two constant numbers
parameter r = 5.7;          // declares r as a real parameter
parameter byte_size = 8, byte_mask = byte_size - 1;
parameter average_delay =(r + f) / 2;
parameter signed [3:0] mux_selector = 0;
parameter real r1 = 3.5e17;
parameter p1 = 13'h7e;
parameter [31:0] dec_const = 1'b1;  // value converted to 32 bits
parameter newconst = 3'h4;          // implied range of [2:0]
```

2.1.6 parameter

모듈 인스턴스의 parameter overriding

```
module modXnor (y_out, a, b);
  parameter  size=8, delay=15;
  output     [size-1:0] y_out;
  input      [size-1:0] a, b;
  wire       [size-1:0] #delay  y_out= a ~^ b;    // bit-wise XNOR
endmodule
```

```
module Param;
  wire  [7:0] y1_out;
  wire  [3:0] y2_out;
  reg   [7:0] b1, c1;
  reg   [3:0] b2, c2;
  modXnor           G1 (y1_out, b1, c1);    // use default parameters
  modXnor  #(4, 5)  G2 (y2_out, b2, c2);    // overrides default parameters
endmodule
```

primitive gate의 delay

```
// Primitive instantiation with 3 units of delay
nand #3 G1 (out_nd2, in0, in1);
```

- 모듈 인스턴스는 지연을 가질 수 없음
- 게이트 프리미티브의 인스턴스는 파라미터 **overriding**을 가질 수 없음