Module #14 : Verilog HDL Array & Memories

14.1: Verilog Array and Memories:

- An array declaration in Verilog can be either scalar or vector.
- Arrays are allowed in Verilog for ‘reg’, ‘wire’, ‘integer’ and ‘real’ data types

Examples :

1) reg X [0 : 7]; // X is a scalar reg array of depth = 8 and each 1 - bit wide

2) reg [7 : 0] X1 [0 :3]; // X1 is an 8 bit reg vector array with a depth of 4

3) reg [7 :0] X2 [0 : 1] [0 : 3] ; X2 is a 2-Dimensional Array with Rows = 2 and Columns = 4 , each 8 bit wide

Module #14 : Verilog HDL Array & Memories

14.2: Verilog Array Assignment:

X = 1'b0; // Illegal assignment as all elements of an array can't be assigned in a single GO !!

X[1] = 1'b0 // Assign 1st element (Index = 1) of X a 0 (1 – Bit Value).

X1 [2] = 8'hAA; // Assign 8'hAA to Index = 2 of X1 Array

X2 [1] [2] = 8'hBB; Assign 8'hBB to Row = 1; Column = 2 of 2 – D Array X2

Module #14 : Verilog HDL Array & Memories

14.3: Few Ky Points :

- Two dimensional arrays can be declared, but can only be accessed by word.
- To get at a bit(s) one must send the output to a register or wire and select the bits from this new variable
- To change one bit, one must read the whole word, change the bit, and write back the word.

Module #14 : Verilog HDL Array & Memories

14.4: Initializing Memory From a File :

- The command **$readmemb** will read a file of binary numbers into the array.
- The data file consists of addresses and data.
- An address written in hex as @hhh...and indicates the address of the first word in a block of data.
- It is followed by binary data words separated by blanks. Legal binary bits are “0 1 X Z _”.
- Data not included in the file will be given xxx... values.
- The data may be given in noncontiguous blocks if an address proceeds each block.
- If no initial address is given, @000 is assumed for the first data word.
- Comments are allowed in data files.
- If ***start_addr*** is given the memory array will be filled starting at that address and continue until ***finish_addr*** (or the end of the array) is reached.
- The command **$readmemh** is similar except the data must contain hexadecimal numbers.

@00A // Start address
10101100 11110000 1x000x11 11110101 01011010 01001100 XxxxZzzz 00000000

Module #14 : Verilog HDL Array & Memories

14.4: Initializing Memory From a File :

Syntax:

reg [wordsize:0] array [0:arraysize]
readmemb("file_name", array_name);
readmemb("file_name", array_name, start_addr);
readmemb("file_name", array_name, start_addr, finish_addrs);
readmemh("file_name", array_name);

// Note: start_addr and finish addr are optional

Module #14 : Verilog HDL Array & Memories

14.4: Initializing Memory From a File :

Example 14.1:

```
reg [7:0] memry [0:31]; // 32 byte memory.
wire [7:0] memwrd;
wire x;
initial begin
// Initialize memory contents from file.
$readmemb("init.dat", memry, 8);
// Start reading at word 8. However the first word in the file
// is 10 (@00A), so words 8 and 9 will default to x.
end
- - -
// Extract last word in memory.
assign memwrd= memry[31];
// Extract most sig bit in word 31
assign x= memwrd[7];
```

```
------------------------------ file init.dat------------------------------
// Since start_addr =8 memry[0:9] will all be stored as
xxxxxxxx.
@00A //
 10101100 11110000 1x000x11 11110101 01011010 01001100
 XxxxZzzz 00000000
@01E // 5'h1E = 5'd30. Underscore gives readability.
 1100_1010 0011_0001
```