DAY-56
#100DAYSRTL

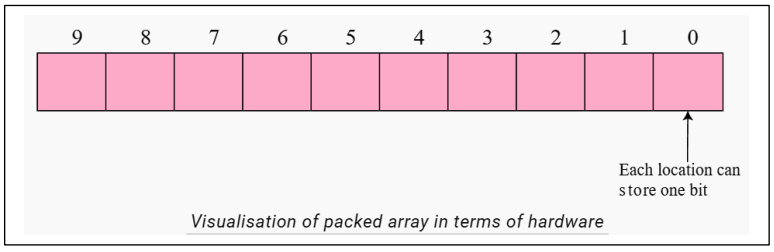
"System Verilog: Introduction to Arrays"

Introduction:-

In Verilog, we have seen that only static arrays can be created. Static arrays have a major drawback as the size of the arrays once defined cannot be changed. This wasted a lot of memory space as at times the entire size of the array is not used. Dynamic arrays were introduced in the System Verilog along with a few other array constructs like associative arrays and queues.

Packed Array:-

Packed arrays can be considered as a single dimensional memory, in which the index i represents the ith bit of the memory. Packed arrays can only be used with data types having width of one bit, i.e., only logic, bit, reg can be used. Any other data type cannot be used.



Visualisation of packed array in terms of hardware

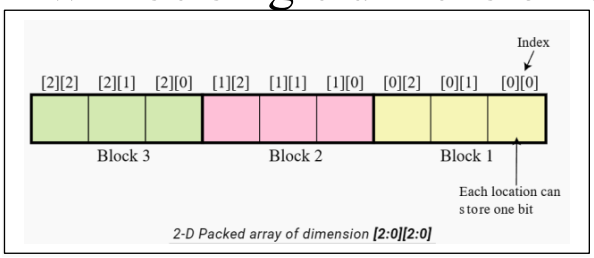
In the figure below, we see that all the data are closely packed with each address or index having one bit data. Thus, we cannot use multiple bits data type such as int, byte, etc to declare a packed array.

Syntax:-

```
bit [<index of MSB>:<index of LSB>] <identifier>;
```

Multidimensional Packed Arrays:-

In System Verilog, packed arrays can be of multiple dimensions. Multiple dimensions should not be confused with the physical layout. Physically it will be single dimension only



So, multi-dimensional packed array just means that the single block of 1-D memory is split into different blocks inside the simulator. This can be beneficial when the size of packed array is large.

Code Practising:-

```
module packed_arr_ex;
    //  declaring 3-D packed array
    //  total size - 256 bits
  bit [3:0][3:0][15:0] arr;
    initial begin
        //  assignment with only one index
        arr [1] = 64'h2548AA54;
        arr [3:2] = 128'h458AAC445AD;
        $display("Assignment with only one index");
        $display("Arr = %h", arr);
        $display("Arr[1] = %h", arr[1]);
        $display("Arr[3:2] = %h\n", arr[3:2]);
        //  assignment with 2 indices
        arr [0][0] = 16'h12;
        arr [0][3:2] = 16'h34;
        $display("Assignment with  2 indices");
        $display("Arr = %h", arr);
        $display("Arr[0][0] = %h", arr[0][0]);
        $display("Arr[0][3:2] = %h\n", arr[0][3:2]);
        //  assignment with 3 indices
        arr [0][1][9] = 1'b1;
        arr [0][1][8:0] = 9'h4;
        $display("Assignment with  3 indices");
        $display("Arr = %h", arr);
        $display("Arr[0][1][9] = %h", arr[0][1][9]);
        $display("Arr[0][1][8:0] = %h", arr[0][1][8:0]);
    end
endmodule
```

Result:-

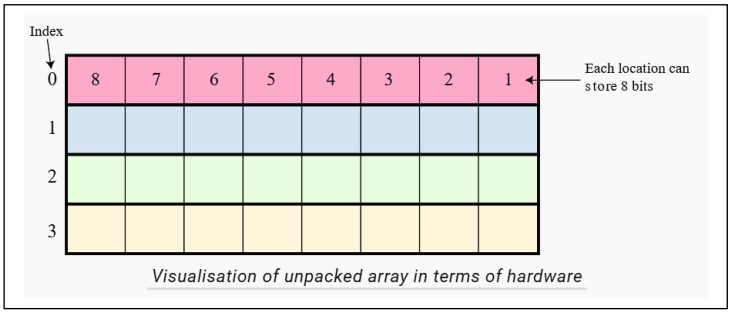
```
Arr = 00000000000000000000458aac445ad000000002548aa540000000000000000
Arr[1] = 000000002548aa54
Arr[3:2] = 00000000000000000000458aac445ad

Assignment with  2 indices
Arr = 00000000000000000000458aac445ad000000002548aa540000003400000012
Arr[0][0] = 0012
Arr[0][3:2] = 00000034

Assignment with  3 indices
Arr = 00000000000000000000458aac445ad000000002548aa540000003402040012
Arr[0][1][9] = 1
Arr[0][1][8:0] = 004
```

Unpacked Arrays:-

Unpacked arrays can be considered as a 2-dimensional memory. A 2 -dimensional memory means that each address of the memory can hold one or more than one bits of data, i.e., there is a width of memory. In simpler terms, it is the same type of array we are acquainted with in other programming language like C or Java. Unpacked arrays can be used with any data types, even 1bit data types.



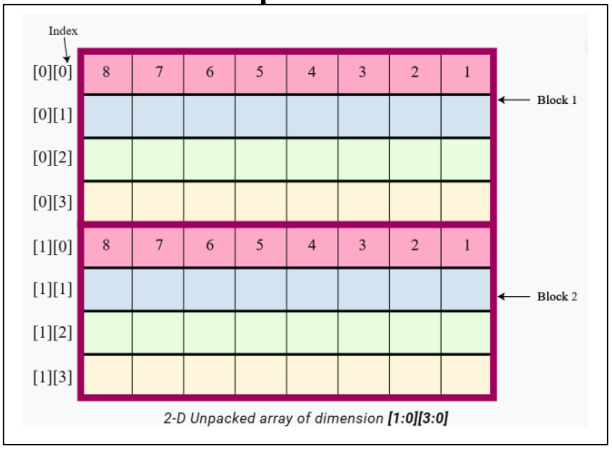
Visualisation of unpacked array in terms of hardware

Syntax:-

```
bit <identifier> [<index of MSB>:<index of LSB>];
```

Multidimensional UnPacked Arrays:-

Just as the packed array, unpacked arrays can also be of multiple dimensions. Again, physically it can be visualized as 2-dimensional matrix which is split in various blocks



2-D Unpacked array of dimension **[1:0][3:0]**

Array slicing and Assignment:-

Array slicing is the same as that of the packed arrays.

While assigning unpacked arrays, we must keep in mind that each index will hold certain number of bits and the assignment should be made keeping in that mind.We use loops or '{} operator to assign elements to an unpacked array. Loops will be discussed later, thus that method will be explained in future articles. In this

article we will focus on the operator method. '{} represents a element thus for arrays of multiple dimension, the '{} operator is nested within the same operator.

Code Practising:-

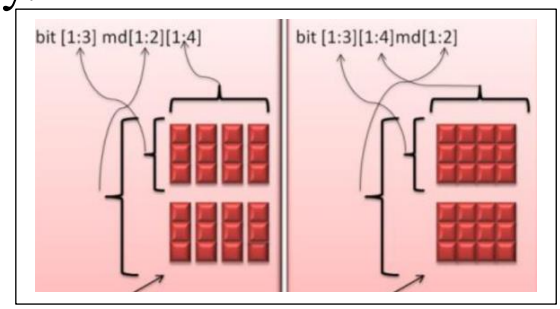
```
module unpacked_arr_ex;
    //  unpacked array declared using size of array
    byte arr1[4];
    //  unpacked array declared using mixed approch
    byte arr2[2][3:0];
    initial begin
        arr1 = '{23, 32, 45, 20};
        $display("Arr1 is %p", arr1);
        //  assignment with only one index
        arr2 [1] = '{8'h34, 12, 'hA, 'hF};
        arr2 [0][0] = 8'h67;
        arr2 [0][3:2] = '{45, 78};
        $display("Arr2 is %p", arr2);
        //  Multiple operator are nested for multi-dimensional
        arr2 = '{'{34, 56, 87, 23}, '{12, 34, 57, 68}};
        $display("Arr2 is %p", arr2);
        $display("Arr2[0] is %p", arr2[0]);
        $display("Arr2[1] is %p", arr2[1]);
        $display("Arr2[0][2] is %p", arr2[0][2]);
        $display("Arr2[1][3:2] is %p", arr2[1][3:2]);
    end
endmodule
```

Result:-

```
Arr1 is '{23, 32, 45, 20}
Arr2 is '{'{45, 78, 0, 103}, '{52, 12, 10, 15}}
Arr2 is '{'{34, 56, 87, 23}, '{12, 34, 57, 68}}
Arr2[0] is '{34, 56, 87, 23}
Arr2[1] is '{12, 34, 57, 68}
Arr2[0][2] is 56
Arr2[1][3:2] is '{12, 34}
```

Mixed Arrays:-

Mixed arrays are the combination of packed and unpacked array. In most of the cases it will not be possible to use the inbuilt data types for array creation. In these cases, we can combine both the types in single array.



Code Practising:-

```
module mixed_array_ex;
    //  declaring mixed array
    bit [7:0][15:0] arr [4];
    initial begin
        //  assignment with only one index
        arr [1] = 128'h234AD342;
        $display("Assignment with only one index");
        $display("Arr[1] = %h", arr[1]);
        arr [0][1] = 16'h23;
        arr [0][2][1] = 1'b1;
        $display("Arr = %h", arr);
    end
endmodule
```

Result:-

```
Assignment with only one index
Arr[1] = 00000000000000000000000234ad342
Arr = '{8592228352, 592106306, 0, 0}
```