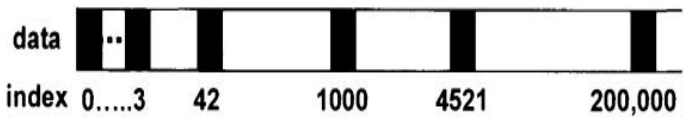
Associative Arrays in SystemVerilog

Dynamic arrays are good if you want to occasionally create a big array, but what if you want something really large? Perhaps you are modeling a processor that has a multi-gigabyte address range. During a typical test, the processor may only touch a few hundred or thousand memory locations containing executable code and data, so allocating and initializing gigabytes of storage is wasteful.

SystemVerilog offers associative arrays that store entries in sparse matrix. This means that while you can address a very large address space SystemVerilog only allocates memory for an element when you write to it.

Figure 2-4 Associative array



Associative Array Declaration:

Syntax: **datatype array_name [index type];**

Where , data_type – data type of the array elements.
array_name – name of the associative array.
index_type – data-type to be used as an index, or * .
we can use any data type

* indicates the array is indexed by any integral expression of arbitrary size.

CODE-1 Declaring. Initializing and using associative arrays .

```
module a_array;
int a_array[int]; //integer array with integer index

initial begin
        a_array = '{1:10, 2:20, 3:30, 4:50, 5:60};
        $display("associative array size=%0d",a_array.size());

        foreach(a_array[i])
    $display("\t a_array[%0d]=%0d",i,a_array[i]);
end
endmodule
```

```
# QuestaSim-64 vlog 2021.1 Compiler 2021.01 Jan 19 2021
# Start time: 17:47:25 on Sep 04,2023
# vlog -reportprogress 300 AArray.sv
# -- Compiling module a_array
#
# Top level modules:
#       a_array
# End time: 17:47:26 on Sep 04,2023, Elapsed time: 0:00:01
# Errors: 0, Warnings: 0
# vsim a_array
# Start time: 17:45:45 on Sep 04,2023
# ** Note: (vsim-3812) Design is being optimized...
# Loading sv_std.std
# Loading work.a_array(fast)
# associative array size=5
#        a_array[1]=10
#        a_array[2]=20
#        a_array[3]=30
#        a_array[4]=50
#        a_array[5]=60

VSIM(paused)>
```

CODE-2: Using an associative array with a string index.

```
module a_array;
int a_array1[string];     //integer array with string index
string a_array2[string];     //string array with string index

initial begin
          a_array1 = '{"maths":10, "chem":18, "phy":21,"eng":15};
            $display("\n Size of a_array1 is %0d ",a_array1.size());

           foreach(a_array1[i])
              $display(" a_array1[%0d]=%0d",i,a_array1[i]);

          a_array2 = '{"Name":"Arti", "sub":"SV", "year":"2023"};
          $display("\n Size of a_array2 is %0d ",a_array2.size());
          foreach(a_array2[i])
          $display(" a_array2[%0d]=%0d",i,a_array2[i]);

end
endmodule
```

```
# QuestaSim-64 vlog 2021.1 Compiler 2021.01 Jan 19 2021
# Start time: 18:15:18 on Sep 04,2023
# vlog -reportprogress 300 string.sv
# -- Compiling module a_array
#
# Top level modules:
#       a_array
# End time: 18:15:18 on Sep 04,2023, Elapsed time: 0:00:00
# Errors: 0, Warnings: 0
# End time: 18:15:20 on Sep 04,2023, Elapsed time: 0:00:39
# Errors: 0, Warnings: 0
# vsim a_array
# Start time: 18:15:20 on Sep 04,2023
# ** Note: (vsim-3813) Design is being optimized due to module recompilation...
# Loading sv_std.std
# Loading work.a_array(fast)
#
#  Size of a_array1 is 4
#  a_array1[chem]=18
#  a_array1[eng]=15
#  a_array1[maths]=10
#  a_array1[phy]=21
#
#  Size of a_array2 is 3
#  a_array2[Name]=Arti
#  a_array2[sub]=SV
#  a_array2[year]=2023
```

Associative array methods

Array Methods:

num:

Returns the number of entries of an associative array.

delete:

delete all the array elements.

And also possible to delete only one elements(which is not possible in Dynamic array.)

exists:

Returns 1 if an element exists at a specified index else returns 0.

First:

Returns first index value in the array.

last:

Returns last index value in the array.

next:

Returns next index value in the array for a given index value.

prev:

Returns previous index value in the array for a given index value.

CODE-3 Example of associative array Methods

```
module a_array;
  bit [7:0] a_array [int];
  int b;
  initial
    begin
      a_array[5]=20;
      a_array[17]=2;
      a_array[1]=10;
      a_array[2]=21;
      a_array[4]=22;
      a_array[6]=23;

     $display("\nSize of associative array  is %0d",a_array.size());
      $display("Number of entries is %0d ",a_array.num());

      if(a_array.exists(4))
        $display("\nElement is exists at index = 7");
      else
        $display("Invalid Index");

      a_array.last(b);
      $display("\nLast index of array a_array is %0d ",b);

      a_array.first(b);
      $display("\nFirst index of array is %0d",b);

      b=2;
      a_array.next(b);
      $display("\nNext index of 2 is %0d",b);

      b=2;
      a_array.prev(b);
      $display("\nPrev index of 2 is %0d",b);

      a_array.delete(17);
      $display("\nArray a_array after deleting index 17");
      foreach(a_array[i])
        $display("\t a_array=[%0d]=%0d",i,a_array[i]);

    end
endmodule
```

```
#
# Size of associative array  is 6
# Number of entries is 6
#
# Element is exists at index = 7
#
# Last index of array a_array is 17
#
# First index of array is 1
#
# Next index of 2 is 4
#
# Prev index of 2 is 1
#
# Array a_array after deleting index 17
#        a_array=[1]=10
#        a_array=[2]=21
#        a_array=[4]=22
#        a_array=[5]=20
#        a_array=[6]=23
```