



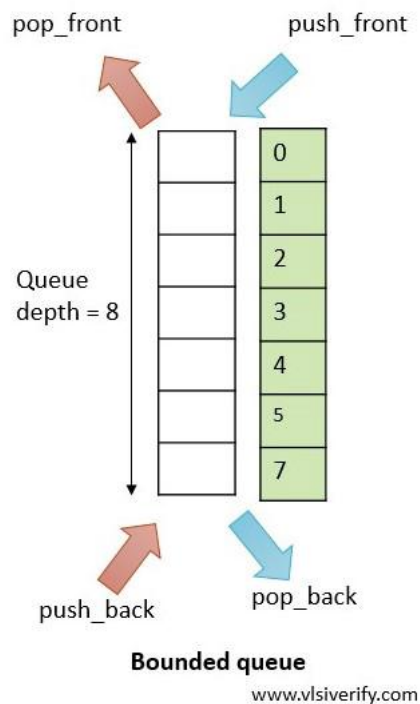
SystemVerilog Queues

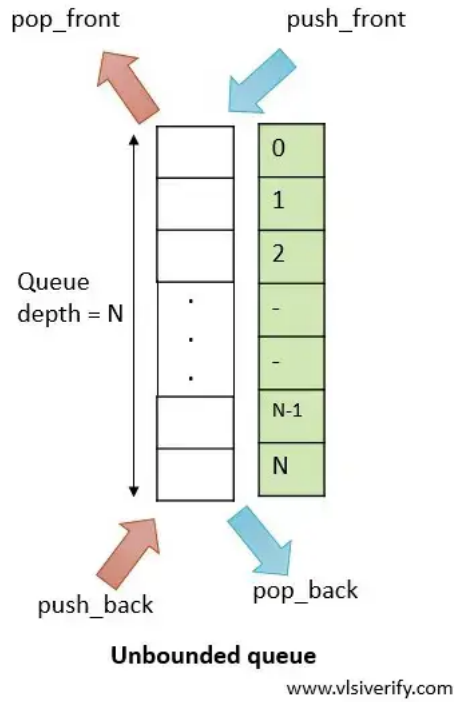
A queue is a variable size and ordered collection of elements (homogeneous element).

To understand it is considered the same as a single-dimensional unpacked array that grows and reduces automatically if it is a bounded queue.

Types of queues in SystemVerilog

1. Bounded queue: Queue having a specific size or a limited number of entries.
2. Unbounded queue: Queue having non-specific queue size or unlimited entries.





Declaration of a queue in SystemVerilog

```
data_type <queue_name> [$];
```

For Example:

```
bit q_1[$]; // Unbounded queue of bit
byte q_2[$]; // Unbounded queue of byte
int q_3 [$:9]; // Bounded queue with qsize = 10

int q_4[$] = {5,6,7};
```

SystemVerilog Queue methods

Methods (functions)	Description
insert (<index>, <item>)	Inserts an item at a specified index.
1. delete(<index>) 2. delete	1. Deletes an item at a specified index 2. Deletes all elements in the queue.
size()	If the queue is not empty, return the number of items in the queue. Otherwise, it returns 0.
push_back(<item>)	Inserts an item at the end of the queue.
pop_back()	Returns and removes the last item of the queue.
push_front(<item>)	Inserts an item at the front of the queue.
pop_front()	Returns and removes the first item of the queue.

shuffle()

Shuffles items in the queue

SystemVerilog Queue Example

```

1  module queue_example;
2      // declaration
3      string animal_q[$];
4
5      initial begin
6          $display("Initial Size: animal_q = %0d", animal_q.size());
7
8          animal_q = {"TIGER", "LION"};
9          $display("Size: animal_q = %0d", animal_q.size());
10         $display("-----");
11
12         animal_q.insert(1, "ELEPHANT");
13         animal_q.insert(3, "FOX");
14         animal_q.insert(4, "ZEBRA");
15         $display("Size: animal_q = %0d", animal_q.size());
16
17         foreach(animal_q[i]) $display("animal_q[%0d] = %s", i, animal_q[i]);
18         $display("-----");
19
20         $display("--- Access queue item ---");
21         $display("The second element of animal_q = %s", animal_q[2]);
22         $display("The fourth element of animal_q = %s", animal_q[4]);
23         $display("-----");
24
25         $display("--- Delete queue item ---");
26         animal_q.delete(2);
27         foreach(animal_q[i]) $display("animal_q[%0d] = %s", i, animal_q[i]);
28         $display("-----");
29
30         $display("--- Delete complete queue ---");
31         animal_q.delete();
32         $display("Size after queue deletion: animal_q size = %0d", animal_q.size());
33         $display("-----");
34
35
36         animal_q = {"TIGER", "LION"};
37
38         $display("--- push_back methods ---");
39         animal_q.push_back("ELEPHANT");
40         foreach(animal_q[i]) $display("animal_q[%0d] = %s", i, animal_q[i]);
41         $display("-----");
42
43         $display("--- push_front methods ---");
44         animal_q.push_front("FOX");
45         foreach(animal_q[i]) $display("animal_q[%0d] = %s", i, animal_q[i]);
46         $display("-----");
47
48         $display("--- pop_back methods ---");
49         animal_q.pop_back();
50         foreach(animal_q[i]) $display("animal_q[%0d] = %s", i, animal_q[i]);
51         $display("-----");
52
53         $display("--- pop_front methods ---");
54         animal_q.pop_front();
55         foreach(animal_q[i]) $display("animal_q[%0d] = %s", i, animal_q[i]);
56         $display("-----");
57     end
58 endmodule

```

Output:

```

Initial Size: animal_q = 0
Size: animal_q = 2
-----
Size: animal_q = 5
animal_q[0] = TIGER
animal_q[1] = ELEPHANT
animal_q[2] = LION
animal_q[3] = FOX
animal_q[4] = ZEBRA
-----
--- Access queue item ---
The second element of animal_q = LION
The fourth element of animal_q = ZEBRA
-----
--- Delete queue item ---
animal_q[0] = TIGER
animal_q[1] = ELEPHANT
animal_q[2] = FOX
animal_q[3] = ZEBRA
-----
--- Delete complete queue ---
Size after queue deletion: animal_q size = 0
-----
--- push_back methods ---
animal_q[0] = TIGER
animal_q[1] = LION
animal_q[2] = ELEPHANT
-----
--- push_front methods ---
animal_q[0] = FOX
animal_q[1] = TIGER
animal_q[2] = LION
animal_q[3] = ELEPHANT
-----
--- pop_back methods ---
animal_q[0] = FOX
animal_q[1] = TIGER
animal_q[2] = LION
-----
--- pop_front methods ---
animal_q[0] = TIGER
animal_q[1] = LION
-----

```

Example for shuffle method

Let's see how the shuffle method shuffles queue's items.

```

1  module queue_example;
2      // declaration
3      int num_q[$];
4
5      initial begin
6          for(int i = 0; i < 10; i++) num_q.push_back(i);
7          $display("---- Before shuffle ----");
8          foreach(num_q[i]) $display("num_q[%0d] = %0d", i, num_q[i]);
9          num_q.shuffle();
10         $display("-----");
11         $display("---- After shuffle ----");
12         foreach(num_q[i]) $display("num_q[%0d] = %0d", i, num_q[i]);
13     end
14 endmodule

```

Output:

```

--- Before shuffle ---
num_q[0] = 0
num_q[1] = 1
num_q[2] = 2
num_q[3] = 3
num_q[4] = 4
num_q[5] = 5
num_q[6] = 6
num_q[7] = 7
num_q[8] = 8
num_q[9] = 9
-----
--- After shuffle ---
num_q[0] = 1
num_q[1] = 3
num_q[2] = 9
num_q[3] = 6
num_q[4] = 8
num_q[5] = 5
num_q[6] = 2
num_q[7] = 4
num_q[8] = 0
num_q[9] = 7

```

Array of queues

An array can store queues. In the below example,

array[0] stores a queue of even numbers.

array[1] stores a queue of odd numbers.

array[2] stores a queue of multiple hundreds.

Initialization of array of queues

- Based on array index

```

array[0] = {2, 4, 6, 8};
array[1] = {1, 3, 5, 7};
array[2] = {100, 200, 300};

```

- Without using an array index

```

array = '{ {2, 4, 6, 8},
           {1, 3, 5, 7},
           {100, 200, 300}
        }';

```

Array of queues Example

```

1 module array_example;
2   int array [3][$];
3
4   initial begin
5     //array[0] = {2, 4, 6, 8};
6     //array[1] = {1, 3, 5, 7};
7     //array[2] = {100, 200, 300};
8     //or

```

```

9      array = '{ {2, 4, 6, 8},
10                {1, 3, 5, 7},
11                {100, 200, 300}
12            };
13
14      // Print array of queues
15      foreach (array[i,j]) $display("array[%0d][%0d] = %0d", i, j, array[i][j]);
16      $display("-----");
17
18      array[0].push_back(10);
19      array[1].push_back(9);
20      array[2].push_back(400);
21
22      $display("After push_back operation");
23      // Print array of queues
24      foreach (array[i,j]) $display("array[%0d][%0d] = %0d", i, j, array[i][j]);
25
26      end
27  endmodule

```

Output:

```

array[0][0] = 2
array[0][1] = 4
array[0][2] = 6
array[0][3] = 8
array[1][0] = 1
array[1][1] = 3
array[1][2] = 5
array[1][3] = 7
array[2][0] = 100
array[2][1] = 200
array[2][2] = 300
-----
After push_back operation
array[0][0] = 2
array[0][1] = 4
array[0][2] = 6
array[0][3] = 8
array[0][4] = 10
array[1][0] = 1
array[1][1] = 3
array[1][2] = 5
array[1][3] = 7
array[1][4] = 9
array[2][0] = 100
array[2][1] = 200
array[2][2] = 300
array[2][3] = 400

```

System Verilog Tutorials

- [Data Types in SV](#)
- [SystemVerilog Arrays](#)
- [Dynamic Array in SV](#)
- [Associative array in SV](#)
- [Array manipulation methods](#)
 - [Array locator methods](#)
 - [Array ordering methods](#)
 - [Array reduction methods](#)
 - [Iterator index querying](#)

- SystemVerilog Queues
- Structures and Unions in SV
- If statement in SV
- Loops in SV
 - while loop
 - do while loop
 - forever loop
 - For Loop
 - Foreach loop
 - Repeat loop
 - Break and Continue
- Functions
- Tasks
- SystemVerilog Processes
- SystemVerilog Classes
 - Accessing unallocated memory
 - Class assignment in SV
 - An array of objects
 - Static properties and methods in SV classes
 - this keyword in classes
 - Shallow copy
 - Deep copy
 - Inheritance
 - Super keyword in classes
 - virtual keyword in classes
 - Abstract class
 - Polymorphism in SV
 - Scope Resolution Operator in SV
 - Extern method in Classes
 - Parameterized classes
 - Data Encapsulation and Hiding
 - Local access Qualifier
 - Protected access Qualifier
 - Constant class properties
 - typedef Class in SV
- Randomization in SV
 - If else in constraints
 - Implication operator in constraint
 - foreach loop in constraint
 - Dist keyword in constraints
 - Inheritance in constraint
 - Function in constraint
 - Disable randomization
 - Static constraints
 - Unique constraint
 - Bidirectional constraint
 - Solve Before in constraint
 - Inline constraints
 - Soft constraint
 - Randomization methods
 - randcase in SV
- InterProcessCommunication
 - SV Events
 - Nonblocking Events
 - wait_order in SV events
 - Merging events in SV
 - Passing an Event

- [SV Semaphores](#)
- [SV Mailbox](#)
- [System Verilog interface](#)
 - [SystemVerilog Modport](#)
 - [SystemVerilog Clocking Block](#)
 - [Virtual Interface](#)
- [Scheduling Semantics](#)
- [Program Block](#)
- [SystemVerilog Casting](#)
- [SystemVerilog packages](#)
- [SystemVerilog compiler directives](#)
- [SystemVerilog parameters](#)
- [SystemVerilog final block](#)
- [Named Blocks, Statement Labels](#)
- [SystemVerilog callback](#)
- [SystemVerilog DPI](#)
- [Verification process and Testbench](#)
- [SV Adder TB Example](#)

[PREVIOUS](#)

[NEXT](#)

Interview Questions

[Verilog Interview Questions with Answers](#)
[SystemVerilog Interview Questions with Answers](#)
[UVM Interview Questions with Answers](#)

Learn More

[ASIC Flows](#)
[Verilog Codes](#)
[Resources](#)
[Blogs](#)

Tutorials

[Verilog Tutorials](#)
[SystemVerilog Tutorials](#)
[Functional Coverage Tutorials](#)
[Assertions Tutorials](#)
[UVM Tutorials](#)
[TLM Tutorials](#)
[RAL Tutorials](#)

[Contact Us](#)
[Privacy Policy](#)

Follow Us

Support Us On