**Chapter -4: Arras, Queues**

Q1.In a module, declare a packed array of 8-bit data and array size 64? Declare an unpacked array of 8-bit data and array size 64?

Ans. module tb();

logic [63:0] [7:0] arr1;

logic [7:0] arr2 [63:0];

initial begin

foreach(arr1[i])

begin

arr1[i]=i;

$display("arr1[%d]=%d",i,arr1[i]);

end

foreach(arr2[j])

begin

arr2[j]=j;

$display("arr2[%d]=%d",j,arr2[j]);

end

end

endmodule

========================output==============================

ncsim> run

arr1[ 63]= 63

arr1[ 62]= 62

arr1[ 61]= 61

arr1[ 60]= 60

arr1[ 59]= 59

arr1[ 58]= 58

arr1[ 57]= 57

arr1[ 56]= 56

arr1[ 55]= 55

arr1[ 54]= 54

arr1[ 53]= 53

arr1[ 52]= 52

arr1[ 51]= 51

arr1[ 50]= 50

arr1[ 49]= 49

arr1[ 48]= 48

arr1[ 47]= 47

arr1[ 46]= 46

arr1[ 45]= 45

arr1[ 44]= 44

arr1[ 43]= 43

arr1[ 42]= 42

arr1[ 41]= 41

arr1[ 40]= 40

arr1[ 39]= 39

arr1[ 38]= 38

arr1[ 37]= 37

arr1[ 36]= 36

arr1[ 35]= 35

arr1[ 34]= 34

arr1[ 33]= 33

arr1[ 32]= 32

arr1[ 31]= 31

arr1[ 30]= 30

arr1[ 29]= 29

arr1[ 28]= 28

arr1[ 27]= 27

arr1[ 26]= 26

arr1[ 25]= 25

arr1[ 24]= 24

arr1[ 23]= 23

arr1[ 22]= 22

arr1[ 21]= 21

arr1[ 20]= 20

arr1[ 19]= 19

arr1[ 18]= 18

arr1[ 17]= 17

arr1[ 16]= 16

arr1[ 15]= 15

arr1[ 14]= 14

arr1[ 13]= 13

arr1[ 12]= 12

arr1[ 11]= 11

arr1[ 10]= 10

arr1[ 9]= 9

arr1[ 8]= 8

arr1[ 7]= 7

arr1[ 6]= 6

arr1[ 5]= 5

arr1[ 4]= 4

arr1[ 3]= 3

arr1[ 2]= 2

arr1[ 1]= 1

arr1[ 0]= 0

arr2[ 63]= 63

arr2[ 62]= 62

arr2[ 61]= 61

arr2[ 60]= 60

arr2[ 59]= 59

arr2[ 58]= 58

arr2[ 57]= 57

arr2[ 56]= 56

arr2[ 55]= 55

arr2[ 54]= 54

arr2[ 53]= 53

arr2[ 52]= 52

arr2[ 51]= 51

arr2[ 50]= 50

arr2[ 49]= 49

arr2[ 48]= 48

arr2[ 47]= 47

arr2[ 46]= 46

arr2[ 45]= 45

arr2[ 44]= 44

arr2[ 43]= 43

arr2[ 42]= 42

arr2[ 41]= 41

arr2[ 40]= 40

arr2[ 39]= 39

arr2[ 38]= 38

arr2[ 37]= 37

arr2[ 36]= 36

arr2[ 35]= 35

arr2[ 34]= 34

arr2[ 33]= 33

arr2[ 32]= 32

arr2[ 31]= 31

arr2[ 30]= 30

arr2[ 29]= 29

arr2[ 28]= 28

arr2[ 27]= 27

arr2[ 26]= 26

arr2[ 25]= 25

arr2[ 24]= 24

arr2[ 23]= 23

arr2[ 22]= 22

arr2[ 21]= 21

arr2[ 20]= 20

arr2[ 19]= 19

arr2[ 18]= 18

arr2[ 17]= 17

arr2[ 16]= 16

arr2[ 15]= 15

arr2[ 14]= 14

arr2[ 13]= 13

arr2[ 12]= 12

arr2[ 11]= 11

arr2[ 10]= 10

arr2[ 9]= 9

arr2[ 8]= 8

arr2[ 7]= 7

arr2[ 6]= 6

arr2[ 5]= 5

arr2[ 4]= 4

arr2[ 3]= 3

arr2[ 2]= 2

arr2[ 1]= 1

arr2[ 0]= 0

ncsim: \*W,RNQUIE: Simulation is complete.

Q2. In SV, write a program using dynamic array for implementing memory of 64x8?

Ans. module tb();

bit [7:0] mem [];

initial begin

//befor memory allocation

$display("Before Memory Allocation");

$display("\tSize of mem %0d",mem.size());

mem= new[64];

// after memory allocation

$display("After Memory Allocation");

$display("\tSize of mem %0d",mem.size());

end

endmodule

=====================output===============================

ncsim> run

Before Memory Allocation

Size of mem 0

After Memory Allocation

Size of mem 64

ncsim: \*W,RNQUIE: Simulation is complete.

Q3. In SV, write a program using Associative array for implementing memory of 64x8?

Ans. module tb();

int a;

int array [\*];

initial begin

for(a=0;a<64;a++)

begin

array[a]=a;

$display("associative [%d]:%d",a,array[a]);

end

//accessing array

if(array.exists(8))

$display("Index 8 exists in a\_array");

else

$display("Index 8 doesnt exists in a\_array");

//last()-Associative array method

array.last(a);

$display("Last entry is array[%0d] = %0d",a,array[a]);

//prev()-Associative array method

array.prev(a);

$display("entry is array[%0d] = %0d",a,array[a]);

//next()-Associative array method

array.next(a);

$display("entry is array[%0d] = %0d",a,array[a]);

end

endmodule

===============================output=======================

ncsim> run

associative [ 0]: 0

associative [ 1]: 1

associative [ 2]: 2

associative [ 3]: 3

associative [ 4]: 4

associative [ 5]: 5

associative [ 6]: 6

associative [ 7]: 7

associative [ 8]: 8

associative [ 9]: 9

associative [ 10]: 10

associative [ 11]: 11

associative [ 12]: 12

associative [ 13]: 13

associative [ 14]: 14

associative [ 15]: 15

associative [ 16]: 16

associative [ 17]: 17

associative [ 18]: 18

associative [ 19]: 19

associative [ 20]: 20

associative [ 21]: 21

associative [ 22]: 22

associative [ 23]: 23

associative [ 24]: 24

associative [ 25]: 25

associative [ 26]: 26

associative [ 27]: 27

associative [ 28]: 28

associative [ 29]: 29

associative [ 30]: 30

associative [ 31]: 31

associative [ 32]: 32

associative [ 33]: 33

associative [ 34]: 34

associative [ 35]: 35

associative [ 36]: 36

associative [ 37]: 37

associative [ 38]: 38

associative [ 39]: 39

associative [ 40]: 40

associative [ 41]: 41

associative [ 42]: 42

associative [ 43]: 43

associative [ 44]: 44

associative [ 45]: 45

associative [ 46]: 46

associative [ 47]: 47

associative [ 48]: 48

associative [ 49]: 49

associative [ 50]: 50

associative [ 51]: 51

associative [ 52]: 52

associative [ 53]: 53

associative [ 54]: 54

associative [ 55]: 55

associative [ 56]: 56

associative [ 57]: 57

associative [ 58]: 58

associative [ 59]: 59

associative [ 60]: 60

associative [ 61]: 61

associative [ 62]: 62

associative [ 63]: 63

Index 8 exists in a\_array

Last entry is array[63] = 63

entry is array[62] = 62

entry is array[63] = 63

ncsim: \*W,RNQUIE: Simulation is complete

Q4. . In SV, write a program using queues for implementing memory of 64x8 and access?Add and remove memory elements using push\_back() and pop\_front()?

Ans. module tb();

byte mem [$];

initial begin

for(integer i=0; i<64; i++)

begin

mem[i]=i;

end

$display("mem size=%d",mem.size());

$display("mem size=%d",mem.pop\_front());

$display("mem size=%d",mem.size());

mem.push\_back(64);

$display("mem size=%d",mem.size());

end

endmodule

=========================output===================

ncsim> run

mem size= 64

mem size= 0

mem size= 63

mem size= 64

ncsim: \*W,RNQUIE: Simulation is complete.