SYSTEM VERILOG

Programming related questions using System Verilog

Madhava Rao Srikantam

Armstrong Number

```
module armstrong_number;

reg i;
integer a,b; //input variables for functions
integer j = 0; //variable to count the order of the number
integer ord,n,temp,sum = 0; //variables to calculate the sum
integer number = 153; // variable to check whether a number is an Armstrong number

function integer order(input b); //order tells us how many digits it has
  begin
    while(b)
      begin
        j = j+1;
        b = b/10;
      end
    order = j;
  end
endfunction

function reg armstrong(input a);
  begin
    temp = a;
    ord = order(a); //First we calculate the order

    while(temp)
      begin
        n = temp%10;
        sum = sum + (n**ord); // Then we calculate the sum based on order
        temp = temp/10;
      end
```

Madhava Rao Srikantam

```
 if(sum == a)
 armstrong = 1'b1;
 else
 armstrong = 1'b0;
 end
endfunction


initial
  begin
    i = armstrong(number);
    if(i)
      $display("The given number %0d is an armstrong number",number);
    else
      $display("The given number %0d is not an armstrong number",number);
  end

endmodule
```

Output:

The given number 153 is an armstrong number

Mirror Number

```
module mirror_number;

int number = 1234321;
int a,a_temp,order_count; //variables for order function
int ord,ord_temp,half,count,temp;//variables for mirror number
bit[3:0] store[]; //dynamic array to store each digit
bit[3:0] remainder; //variable to store the remainder

function automatic integer order(input integer a);
  a_temp = a;
  while(a_temp >= 1)
    begin
      order_count = order_count + 1'b1;
      a_temp = a_temp/10;
    end
  order = order_count;
endfunction

initial
  begin
   ord = order(number);
   store = new[ord];
   ord_temp = number;
```

Madhava Rao Srikantam

```
for(integer i = 0 ; i< ord; i++)
 begin
   remainder = ord_temp%10;
   store[i] = remainder;
   ord_temp = ord_temp/10;
 end

if(ord%2 == 0)
 begin
   half = ord/2;
   temp = ord-1;
   for(int i=0; i<half; i++)
     begin
       if(store[i] == store[temp]) begin
         count++; end
       temp--;
     end
 end

else if (ord%2 == 1)
 begin
   half = (ord-1)/2;
   temp = ord-1;
   for(int i=0; i<half; i++)
     begin
       if(store[i] == store[temp]) begin
         count++; end
       temp--;
     end
 end
```

Madhava Rao Srikantam

```
 if(count == half)
 $display("The given number %0d is a mirror number",number);
 else
 $display("The given number %0d is not a mirror number",number);
 end
endmodule
```

Output

```
The given number 123321 is a mirror number
```

Prime Number

```
module prime_number;

int in = 17;
int out;

initial
  begin
    if(in == 2)
      begin
        out = 1;
      end

    else if(in>2)
      begin
        out = 1;
        for (int i=2 ; i<in; i++)
          begin
            if(in%i == 0)
              out = 0;
          end
      end

    if(out == 1)
      $display("The given number %0d is a prime number",in);
    else
      $display("The given number %0d is not a prime number",in);
  end
endmodule
```

Output :

The given number 17 is a prime number

Madhava Rao Srikantam

Perfect Number

```
module perfect_number;

int number = 6;
int sum;

initial
  begin
    if (number == 1)
      $display("The given number %0d is a perfect number",number);

    else if(number>1)
      begin
        for(int i=1;i<number;i++)
          begin
            if(number%i == 0)
              sum = sum+i;
          end

        if(number == sum)
          $display("The given number %0d is a perfect number",number);
        else
          $display("The given number %0d is a not perfect
number",number);
      end
  end

endmodule
```

Output :

The given number 6 is a perfect number

Madhava Rao Srikantam

Palindrome 1

```
module palindrome;

class example;
  rand bit[31:0] a;
  constraint c1 {foreach(a[i])
        a[i] == a[31-i];}
endclass

example e1;

initial
  begin
    e1 = new();
    repeat(5)
      begin
        assert(e1.randomize());
        $display("The value of a is %b",e1.a);
      end
  end

endmodule
```

Output :

```
The value of a is 10100110101110000001110101100101
The value of a is 00101000101011000011010100010100
The value of a is 11111011101111011011110111011111
The value of a is 11101001101000011000010110010111
The value of a is 00000001101001000010010110000000
```

Madhava Rao Srikantam

Palindrome 2

```
module palindrome;

class example;
  rand bit[31:0] a;
  function void post_randomize;
    for(int i=0; i<16; i++)
      begin
        a[31-i] = a[i];
      end
  endfunction
endclass

example e1;

initial
  begin
    e1 = new();
    repeat(5)
      begin
        assert(e1.randomize());
        $display("The value of a is %b",e1.a);
      end
  end

endmodule
```

Output :

```
The value of a is 00011001011001111110011010011000
The value of a is 01000000100100100100100100000010
The value of a is 11000100110110111101101100100011
The value of a is 01000101010001011010001010100010
The value of a is 11000001111011111111011110000011
```

Madhava Rao Srikantam

Palindrome 3

```
module palindrome;

bit[63:0] name = "abcddcba";
bit[0:31] s ;
int j = 63;

initial
  begin
   for (int i = 0; i<32; i++)
    begin
     s[i] = name[j];
     j = j-1;
    end

    if(s[0:31] == name[63:32])
      $display("The given name %s is a palindrome",name);
    else
      $display("The given name %s is not a palindrome",name);
  end

endmodule
```

Output :

The given name abcddcba is a palindrome

Madhava Rao Srikantam

Sudoku 9x9

```
module top;

class sudoku;

 int M = 3;
 int N = M * M;
 rand bit[3:0] a[9][9];

  // The value of each a must be between 1 and N.
  constraint value {foreach ( a[row, col] )
            a[row][col] inside {[ 1:N ]};}

  // The places on the same row must have unique values.
  constraint row_con { foreach (a[row, colA] )
          { foreach (a[ row , colB] )
          { if ( colA != colB )
          { a[row][colA] != a[row][colB]; } } } }

  // The places on the same column must have unique values.
  constraint column_con { foreach ( a[rowA, col] )
            { foreach ( a[rowB, col ] )
            { if ( rowA != rowB )
            { a[rowA][col] != a[rowB][col]; } } } }

  // The places in the same MxM block must have unique values.
  constraint block_con { foreach(a[rowA,colA])
            {foreach ( a[rowB, colB] )
            { if ( rowA / M == rowB / M && colA / M == colB / M && ! (
rowA == rowB && colA == colB ) )
            { a[rowA][colA] != a[rowB][colB]; } } } }
endclass
```

Madhava Rao Srikantam

```
sudoku s1;

initial
 begin
 s1 = new();
 repeat(1)
 begin
 assert(s1.randomize);
 for(int i=0;i<9;i++)
 begin
 $display("%p",s1.a[i]);
 end
 $display();
 end
 end

endmodule
```

Output :

```
'{2, 4, 9, 6, 8, 3, 7, 5, 1}
'{8, 7, 3, 1, 5, 4, 9, 6, 2}
'{6, 1, 5, 2, 7, 9, 3, 4, 8}
'{4, 6, 7, 5, 2, 8, 1, 3, 9}
'{3, 9, 8, 4, 6, 1, 2, 7, 5}
'{1, 5, 2, 9, 3, 7, 4, 8, 6}
'{7, 3, 6, 8, 1, 2, 5, 9, 4}
'{5, 2, 4, 7, 9, 6, 8, 1, 3}
'{9, 8, 1, 3, 4, 5, 6, 2, 7}
```

Matrix 3x3

```
//sum of each row and column = 4
module top;

class matrix;

rand bit [2:0] m[3][3];

constraint row_sum { foreach(m[i])
        m[i][0]+m[i][1]+m[i][2] == 4;}

constraint column_sum { foreach(m[j])
        m[0][j]+m[1][j]+m[2][j] == 4;}

endclass

matrix m1;

initial

begin
  m1 = new;
  repeat(5)
   begin
    assert(m1.randomize);

  for(int i=0; i<3; i++)
    begin
      $display("%p",m1.m[i]);
    end
    $display();
   end
end

endmodule
```

Output :

```
'{0, 2, 2}    '{1, 1, 2}    '{2, 2, 0}    '{3, 0, 1}    '{1, 2, 1}
'{1, 2, 1}    '{2, 0, 2}    '{2, 1, 1}    '{0, 3, 1}    '{2, 2, 0}
'{3, 0, 1}    '{1, 3, 0}    '{0, 1, 3}    '{1, 1, 2}    '{1, 0, 3}
```

Madhava Rao Srikantam

Fibonacci 1

```
module fibonacci;

typedef struct{int array[];} a;

int array1[];
int number = 8;

function automatic a f(input int number);

 int array2[];

 if(number == 0)
  begin
   array2 = new[number];
   array2[0] = 0;
  end
 else if(number == 1)
  begin
   array2 = new[number];
   array2[0] = 0;
   array2[1] = 1;
  end
 else
  begin
   array2 = new[number];
   array2[0] = 0;
   array2[1] = 1;
   for(int i = 2; i<number; i++)
    begin
     array2[i] = array2[i-1] + array2[i-2];
    end
  end
f.array = array2;
endfunction
```

Madhava Rao Srikantam

```
initial
 begin
 array1 = new[number];
 array1 = f(number).array;

 for(int j=0; j<number; j++)
 begin
 $display("%0d",array1[j]);
 end
 end

endmodule
```

Output :

0 1 1 2 3 5 8 13

Fibonacci 2

```
module top;

class Fibonacci;

rand bit [8:0] a[]; //dynamic array

constraint c1 {a.size inside {[2:11]};
            foreach (a[j])
               if (j == 0)
                 a[j] == 0;
               else if (j == 1)
                 a[j] == 1;
               else
               a[j] == a[j-1] + a[j-2]; }

endclass

Fibonacci f;

initial
  begin
    f = new();
    repeat (5)
      begin
      assert(f.randomize);
       $display("%p",f.a);
      end
  end
endmodule
```

Output :

```
'{0, 1, 1, 2}
'{0, 1, 1, 2}
'{0, 1, 1, 2, 3, 5, 8, 13}
'{0, 1}
'{0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55}
```

Madhava Rao Srikantam

Fibonacci 3

```
module top;

class fibonacci;;

rand bit[8:0] a[];

constraint c1 {a.size inside {[5:15]};}

function void post_randomize();
  a[0] = 0;
  a[1] = 1;

  for(int i=2; i<a.size;i++)
    a[i] = a[i-1] + a[i-2];
endfunction

endclass

fibonacci f;

initial
  begin
    f = new();
    repeat(3)
      begin
        assert(f.randomize);
        $display("%p",f.a);
      end
  end
endmodule
```

Output :

```
'{0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233}
'{0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89}
'{0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233}
```

Madhava Rao Srikantam

Factorial

```
module factorial_of_a_number;

int number = 5;
int result;

function automatic int factorial(input int a);
  if (a == 1)
    factorial = 1;
  else if(a>1)
    factorial = a*factorial(a-1);
endfunction

initial
  begin
    result = factorial(number);
    $display("The factorial of the given number %0d is
%0d",number,result);
  end

endmodule
```

Output :

The factorial of the given number 5 is 120

Madhava Rao Srikantam

Sum of squares of first n natural numbers

```
module sum_of_squares;

int number = 5;
int temp,result;

function int sum(input int a);
  for (int i=1; i<=number; i++)
    temp = temp + (i**2);

  sum = temp;
endfunction

initial
  begin
    result = sum(number);
    $display("The sum of squares of first %0d natural numbers is %0d",
number,result);
  end
endmodule
```

Output :

```
The sum of squares of first 5 natural numbers is 55
```

Madhava Rao Srikantam

Square Root

```
//This is only for perfect squares
//I have used the repeated subtraction method

module square_root;

int number = 25;
int odd=1,count,temp;

initial
  begin
    temp = number;
    while(temp>0)
      begin
        count++;
        temp = temp-odd;
        odd = odd+2;
      end
    $display("The square root of the given number %0d is
%0d",number,count);
  end

endmodule
```

Output :

The square root of the given number 25 is 5

Madhava Rao Srikantam

Nearest multiple of 5

```
module nearest;

int number = 73;
int rem,result;

initial

  begin
    rem = number%5;
    case(rem)
      0: result = number;
      1: result = number - rem;
      2: result = number - rem;
      3: result = number + rem-1;
      4: result = number + rem-2;
    endcase
    $display("The nearest multiple of 5 of the given number
%0d is %0d",number,result);
  end

endmodule
```

Output :

The nearest multiple of 5 of the given number 73 is 75

Madhava Rao Srikantam