Problem Statement

- Create and test a module that computes Fibonacci numbers
- Definition of Fibonacci numbers:
  \[ \text{fib}(0) = 0 \]
  \[ \text{fib}(1) = 1 \]
  \[ \text{fib}(i) = \text{fib}(i-1) + \text{fib}(i-2) \] for \( i \geq 2 \)

The Verilog Code for fibNumberGen

```verilog
always @((posedge start))
begin
  if (n == 0) fibNth = 0;
  else if (n == 1) fibNth = 1;
  else begin
    count = n;
    prevNum = 0;
    for (fibNth = 1; count > 1; count = count - 1)
      //Module to generate nth Fibonacci
      //Number
      module fibNumberGen(n, start, done, v, fibNth);
      input [15:0] n;
      input start;
      output done, v;
      begin
        temp = fibNth;
        fibNth = fibNth + prevNum;
        if (fibNth < prevNum) v = 1;
        prevNum = temp;
      end
      reg done, v;
      reg [15:0] fibNth;
      reg [15:0] count, prevNum, temp;
    initial
      begin
        done = 0;
        v = 0;
      end
    #5 done = 1;
    #10 done = 0; v = 0;
    end
endmodule
```

The fibNumberGen module

When a 0 to 1 transition occurs on `start`, the module computes \( \text{fibNth} = \text{fib}(n) \). When the result is stable on `fibNth`, `done` is held high for 10 nanoseconds. If the computation results in an overflow (i.e. if \( \text{fib}(n) \) is too large to represent in 16 bits) \( v \) (overflow flag) is set and remains set until the next 1 to 0 transition of the `done` flag.

The entire computation, from the leading edge of the `start` signal to the leading edge of the `done` signal takes 100 nsec.
A Module to “Exercise” fibNumberGen

As long as run is high, this module generates successive values 0, 1, 2, 3… on output num. A new output value is generated every 150 nsec. and the ready flag held high for 10 nsec when a new value is stable on the num output.

Verilog Code for Module numberGen

```
module numberGen(num, ready, run);
output [15:0] num;
output ready;
input run;
always
begin
  wait(run == 1);
  #145 num = num + 1;
  #5  ready = 1;
  #10 ready = 0;
end
reg [15:0] num;
reg ready;
initial
begin
  num = -1;
  ready = 0;
end
endmodule
```

Putting the Modules Together

Verilog Code for Module fibTop

```
module fibTop();
reg runIt;
wire [15:0] numOut, fibOut;
wire numRdy, fibDone, fibV;
numberGen M1(numOut, numRdy, runIt);
fibNumberGen M2(numOut numRdy, fibDone, fibV);
always @posedge fibDone
begin
  if (fibV == 1)
  begin
    $display($time, "Overflow at Fib(%d). Simulation terminated", numRdy);
    #1001
  fibNumberGen M2(numOut, numRdy, fibDone, fibV, fibOut);
  initial
  begin
    #1;
    runIt = 1;
  end
  else
    $display($time, "Fib(%d) is: %d", numOut, fibOut);
  wait(fibDone == 0);
end
endmodule
```
Compiling and Running the Verilog Model

Model Technology ModelSim SE vlog 5.5b Compiler 2001.05 May 23 2001

-- Compiling module fibNumberGen

Top level modules:
   fibNumberGen

Model Technology ModelSim SE vlog 5.5b Compiler 2001.05 May 23 2001

-- Compiling module numberGen

Top level modules:
   numberGen

Model Technology ModelSim SE vlog 5.5b Compiler 2001.05 May 23 2001

-- Compiling module fibTop

Top level modules:
   fibTop

Model Technology ModelSim SE vlog 5.5b Compiler 2001.05 May 23 2001

-- Compiling module fibNumberGen

Simulation Run of Module fibTop

VSIM 1> run 10000
# 251 F(0) is: 0
# 411 F(1) is: 1
# 731 F(2) is: 2
# 891 F(3) is: 3
# 1061 F(4) is: 5
# 1211 F(5) is: 8
# 1371 F(7) is: 13
# 1531 F(8) is: 21
# 1691 F(9) is: 34
# 1851 F(10) is: 55
# 2011 F(11) is: 89
# 2171 F(12) is: 144
# 2331 F(13) is: 233
# 2491 F(14) is: 377
# 2651 F(15) is: 610
# 2811 F(16) is: 897
# 2971 F(17) is: 1597
# 3131 F(18) is: 2584
# 3291 F(19) is: 4181
# 3451 F(20) is: 6765
# 3611 F(21) is: 10945
# 3771 F(22) is: 17111
# 3931 F(23) is: 28657
# 4091 F(24) is: 46368
# 4251 Overflow at F(25).
Simulation terminated
# ** Note: $finish : fibTop.v(23)
# Time: 4251 ns Iteration: 1 Instance: fibTop