Problem Statement

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

The Verilog Code for \text{fibNumberGen}

```verilog
always @(posedge start)
begin
  #5 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)
      begin
        temp = fibNth;
        fibNth = fibNth + prevNum;
        if (fibNth < prevNum) v = 1;
        prevNum = temp;
      end
    end
  end
  #5 done = 1;
  #10 done = 0; v = 0;
end

endmodule
```

The fibNumberGen module

When a 0 to 1 transition occurs on \text{start}, the module computes \text{fibNth} = \text{fib}(n). When the result is stable on \text{fibNth}, \text{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) \text{v} (overflow flag) is set 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

```verilog
// module to exercise Fib.Num Generator
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

delaysendmodule
```

Putting the Modules Together

Verilog Code for Module fibTop

```verilog
module fibTop();
reg runIt;
wire [15:0] numOut; fibOut; wire numRdy, fibDone, fibV;
numberGen M1(numOut, numRdy, runIt);
fibNumberGen M2(numOut, numRdy, fibDone, fibV, fibOut);

initial begin
    #1;
    runIt = 1;
end

always @(posedge fibDone)
begin
    if (fibV == 1)
    begin
        $display($time,, "Overflow at Fib(%d). Simulation terminated", numRdy);
    end
    else
    $display($time,, "Fib(%d) is: %d", numOut, fibOut);
    wait(fibDone == 0);
end

delaysendmodule
```
Compiling and Running the Verilog Model

```
$ echo $PS1
$ vlog fibNumberGen.v
$ vlog numberGen.v
$ vlog fibTop.v
```

Simulation Run of Module fibTop

```
VSIM 1> run 10000
# 251 Fib(0) is:  0
# 411 Fib(1) is:  1
# 571 Fib(2) is:  1
# 731 Fib(3) is:  2
# 891 Fib(4) is:  3
# 1161 Fib(5) is:  5
# 1371 Fib(6) is:  8
# 1597 Fib(7) is:  13
# 1851 Fib(8) is:  21
# 2011 Fib(9) is:  34
# 2331 Fib(10) is:  55
# 2584 Fib(11) is:  89
# 2711 Fib(12) is:  144
# 2971 Fib(13) is:  233
# 3291 Fib(14) is:  377
# 3611 Fib(15) is:  610
# 3986 Fib(16) is:  987
# 4378 Fib(17) is: 1597
# 4877 Fib(18) is: 2584
# 5478 Fib(19) is: 4181
# 6175 Fib(20) is: 6765
# 6953 Fib(21) is:10946
# 7832 Fib(22) is:17711
# 8804 Fib(23) is:28657
# 9875 Fib(24) is:46368
# 10946 Fib(25) Overflow at Fib(25).
Simulation terminated
```

Note: The Fibonacci sequence is generated by the Verilog model fibTop. The run command runs the simulation for 10,000 iterations, printing the value of Fibonacci numbers up to Fib(25). The sequence is stored in the variables Fib(0), Fib(1), ..., Fib(25). The simulation terminates due to an overflow at Fib(25).