1) Consider the non-recursive signal processing structure shown below. Find an equivalent implementation of this algorithm to improve the speed of the system. Your proposed architecture should have fewer flip-flops than the current implementation. Calculate the throughput or sample speed of the broadcast architecture.

2) Consider the following recursive filter:

\[ X(n) = aX(n-2) + U(n) \]

Assuming that the multiplication and addition take 3 ns and 1 ns, respectively

a. What is the maximum clock frequency that this filter can work at?
b. Pipeline this filter so that it can operate up to 333 MHz.
c. Write the Verilog code implementing the above filter.
d. How can you improve the above architecture to speed up to work at 1GHz? (hint: you may want to use pipelining at different levels)

3) In this problem you are to write a memory model that initially reads hexadecimal data from an external file, "mem_file.v", and holds it in its internal buffer. Data being read from the memory will be read from the buffer and data being written into the memory will be written into the buffer and into the external memory file, "mem_file.v", at the same time. Assume the data in "mem_file.v" is for the first 256 words of the memory. You are to implement this memory for addresses 0 to 255 only, and reading from addresses outside of this range will return Zs. Use the module declaration shown below.

```verilog
module memory (mem_wr, mem_rd, databus, adbus);
  input mem_wr, mem_rd;
  inout [47:0] databus ;
  input [15:0] adbus;
```
4) Consider a subsequence counter with following input/outputs:
   - **input**: a synchronous signal \( a[31 : 0] \)
   - **input**: a synchronous signal \( s \) which activates the module
   - **input**: the clock signal \( clk \)
   - **output**: a signal \( \text{max}[5 : 0] \) that provides the size of the largest subsequence of ones found in \( a \).

   a) Provide an algorithmic state machine for this module.
   b) Write an RTL code describing this counter
   c) Write a testbench for this module. Test the module on inputs FFFFF, H0000, and HAAAA and display the results on screen. If the result is incorrect it should display an error message on screen. (Hint: assignments to \( a \) and \( s \) should be scheduled at the positive edge of \( clk \)).

5) Consider the following Verilog module that uses Euclid’s algorithm to iteratively compute the greatest common divisor of two 16-bit unsigned integer values \( A_in \) and \( B_in \) where \( A_in \geq B_in \).

   ```verilog
   module gcd(clk,start,A_in,B_in,answer,done);
   input clk,start;
   input [15:0] A_in,B_in;
   output reg [15:0] answer;
   output reg done;
   reg [15:0] a,b;
   always @ (posedge clk) begin
     if (start) begin a <= A_in; b <= B_in; done <= 0; end
     else if (b == 0) begin answer <= a; done <= 1; end
     else if (a > b) a <= a - b;
     else b <= b - a;
   end
   endmodule
   ```

   Please complete the timing diagram below as the module computes the gcd of 21 and 15.