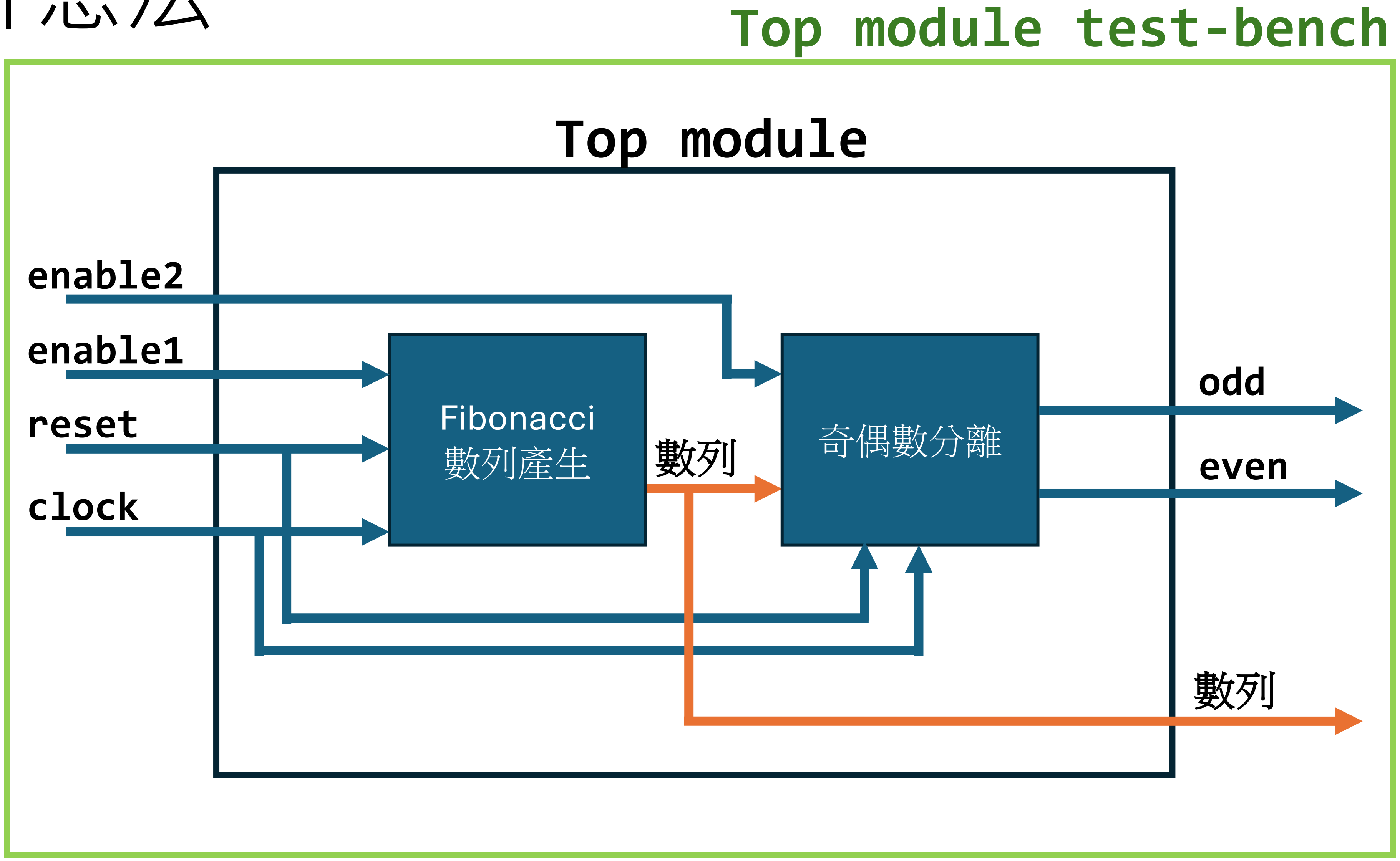
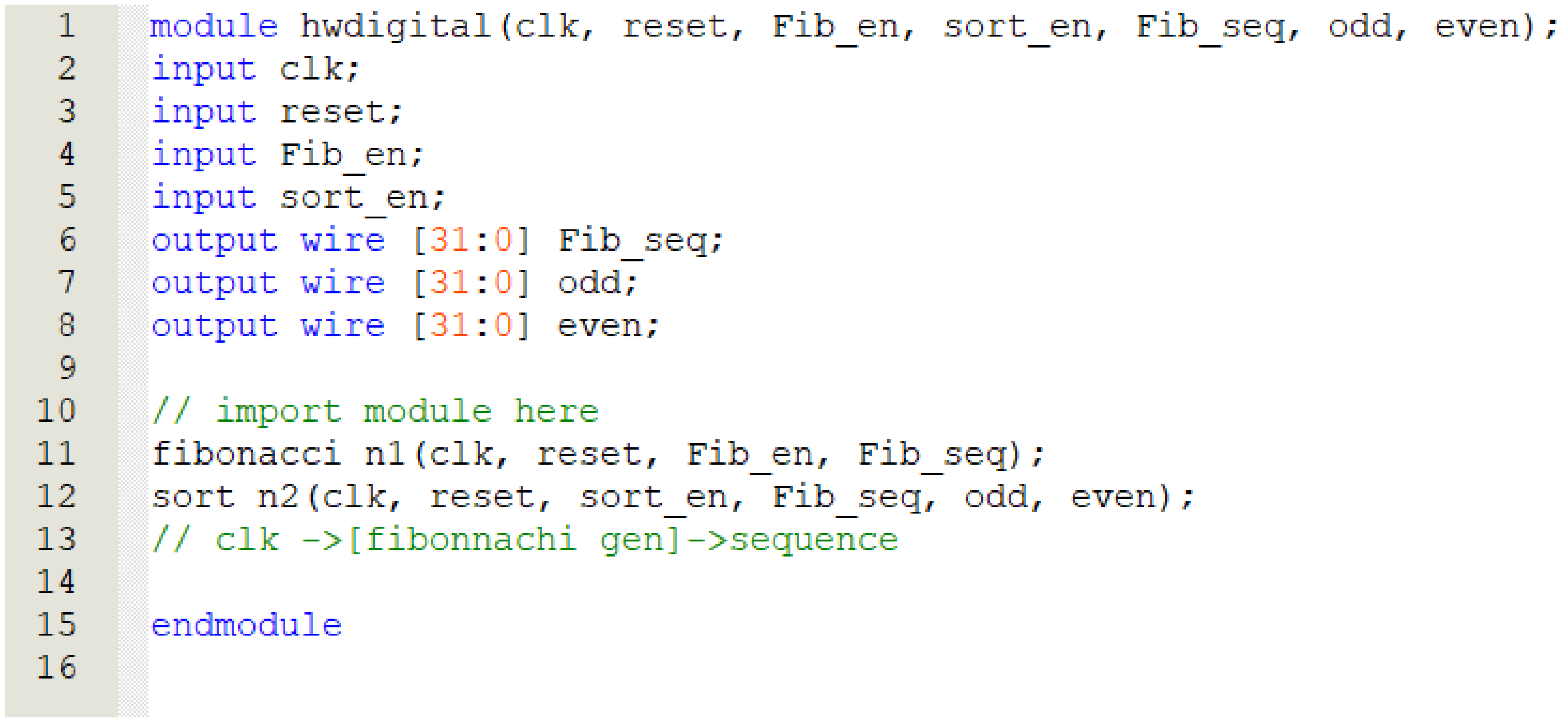
411440117

數位組作業

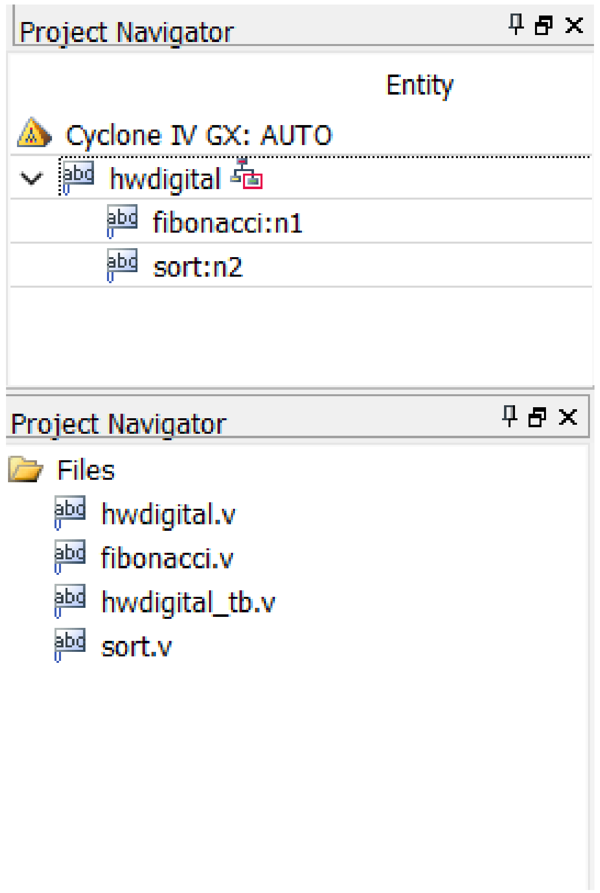
設計想法


Top module test-bench
Top module
enable2
enable1
reset
clock
Fibonacci
數列產生
數列
奇偶數分離
odd
even
數列

主程式碼



```
module hwdigital(clk, reset, Fib_en, sort_en, Fib_seq, odd, even);
input clk;
input reset;
input Fib_en;
input sort_en;
output wire [31:0] Fib_seq;
output wire [31:0] odd;
output wire [31:0] even;

// import module here
fibonacci n1(clk, reset, Fib_en, Fib_seq);
sort n2(clk, reset, sort_en, Fib_seq, odd, even);
// clk ->[fibonnachi gen]->sequence

endmodule

```



Project Navigator

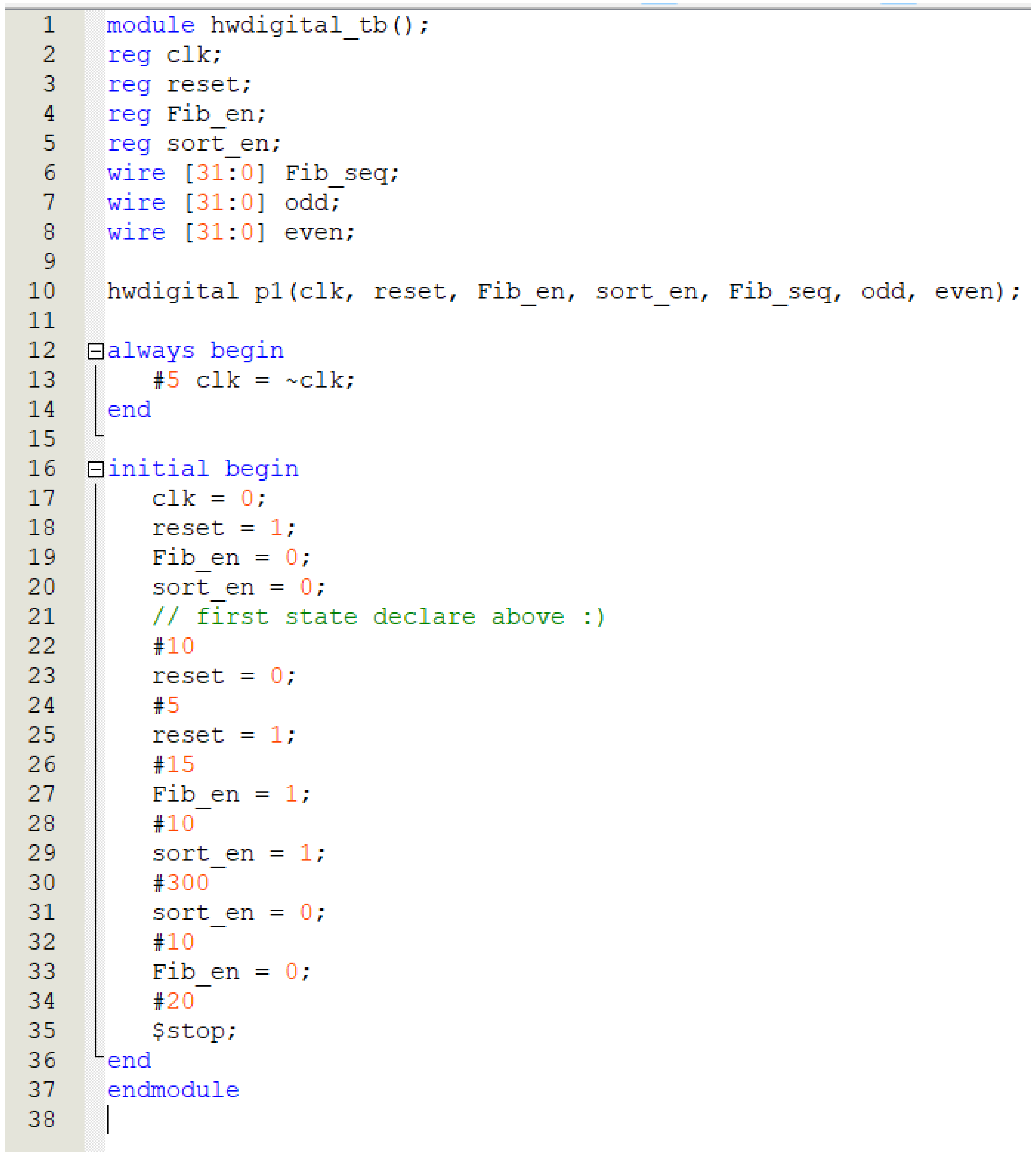
Entity

Cyclone IV GX: AUTO
- hwdigital
  - fibonacci:n1
  - sort:n2

Project Navigator

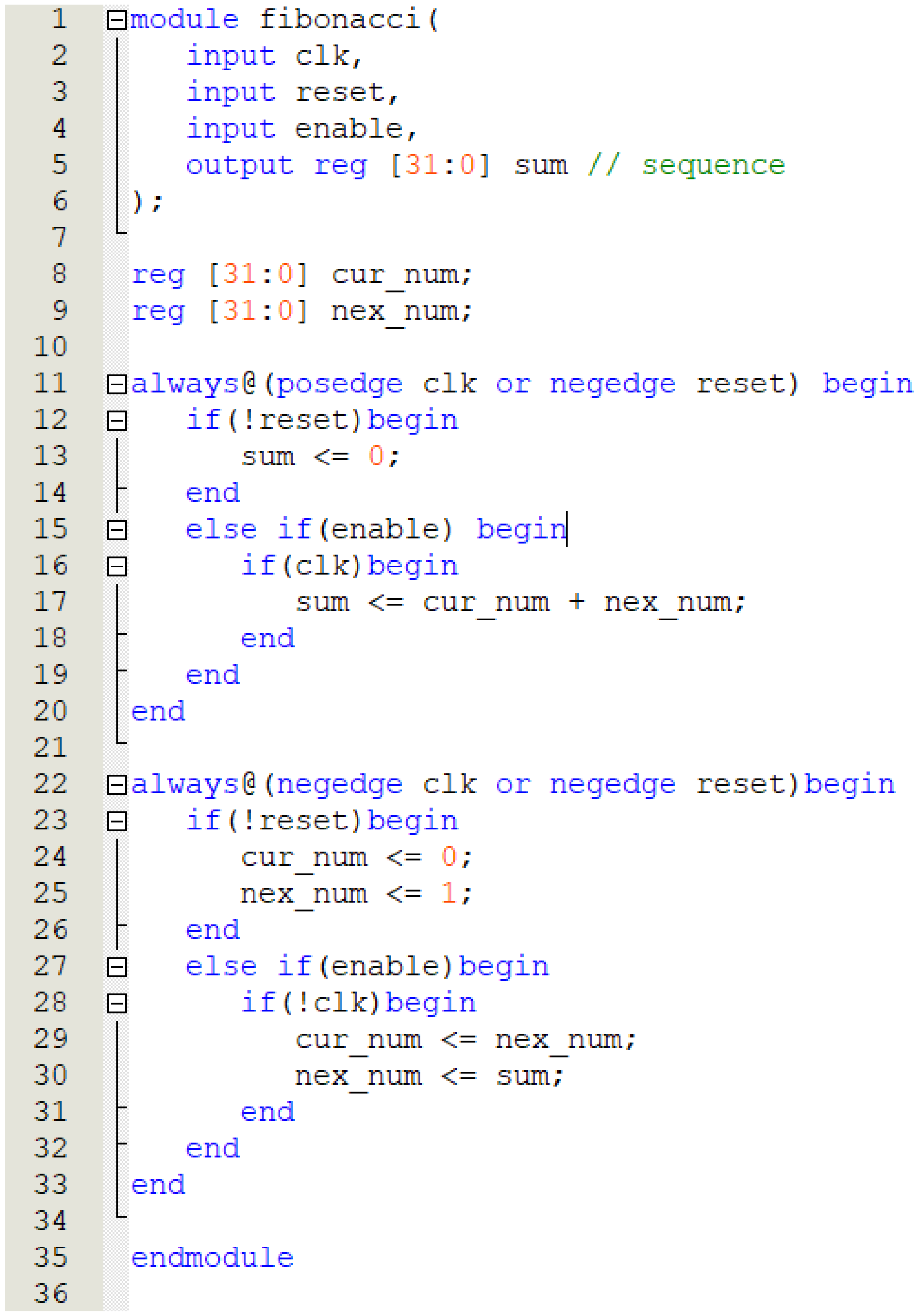
Files
- hwdigital.v
- fibonacci.v
- hwdigital_tb.v
- sort.v

Test Bench



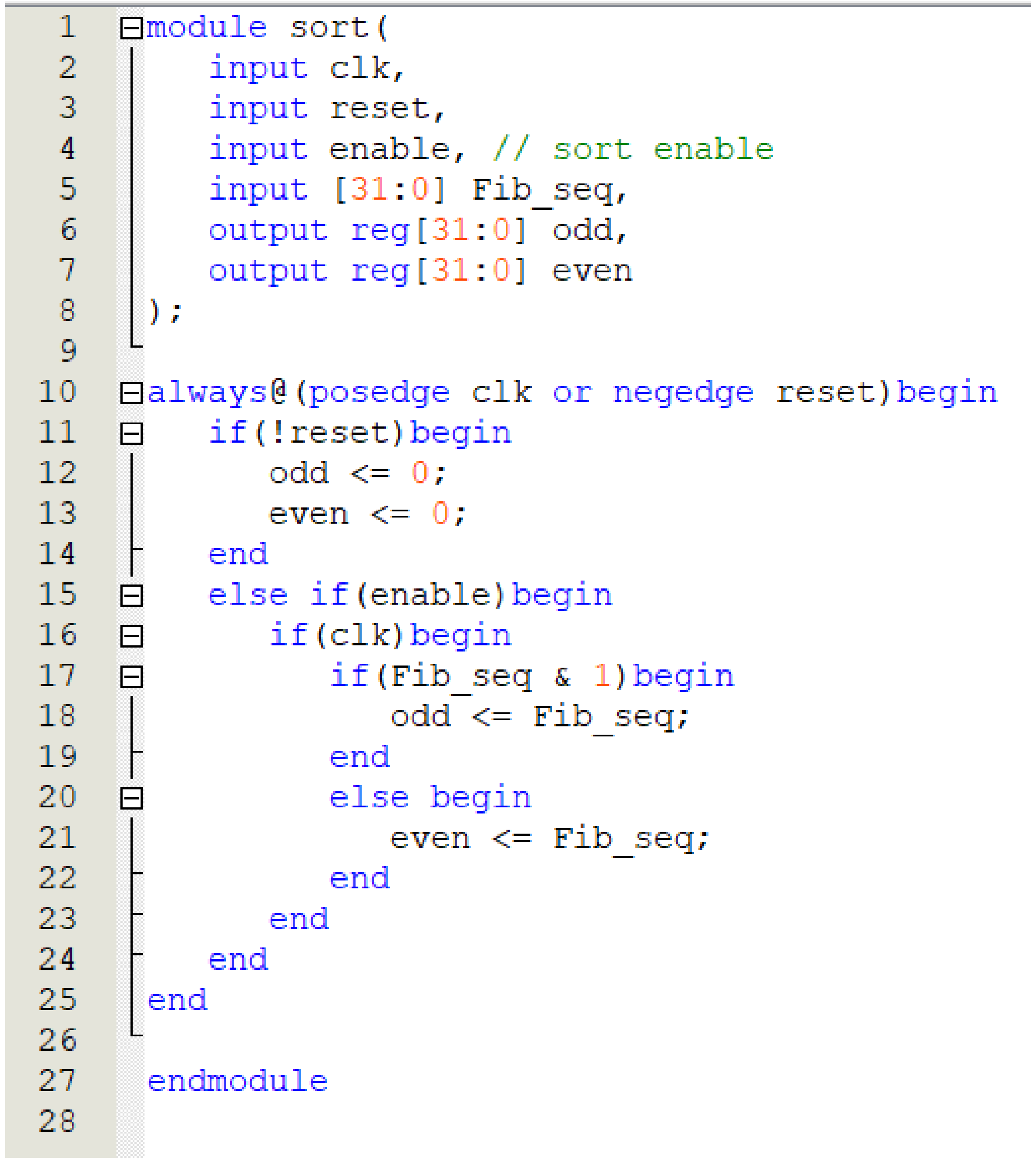
```
module hwdigital_tb();
reg clk;
reg reset;
reg Fib_en;
reg sort_en;
wire [31:0] Fib_seq;
wire [31:0] odd;
wire [31:0] even;

hwdigital p1(clk, reset, Fib_en, sort_en, Fib_seq, odd, even);

always begin
   #5 clk = ~clk;
end

initial begin
   clk = 0;
   reset = 1;
   Fib_en = 0;
   sort_en = 0;
   // first state declare above :)
   #10
   reset = 0;
   #5
   reset = 1;
   #15
   Fib_en = 1;
   #10
   sort_en = 1;
   #300
   sort_en = 0;
   #10
   Fib_en = 0;
   #20
   $stop;
end
endmodule
```

Fibonacci 數列



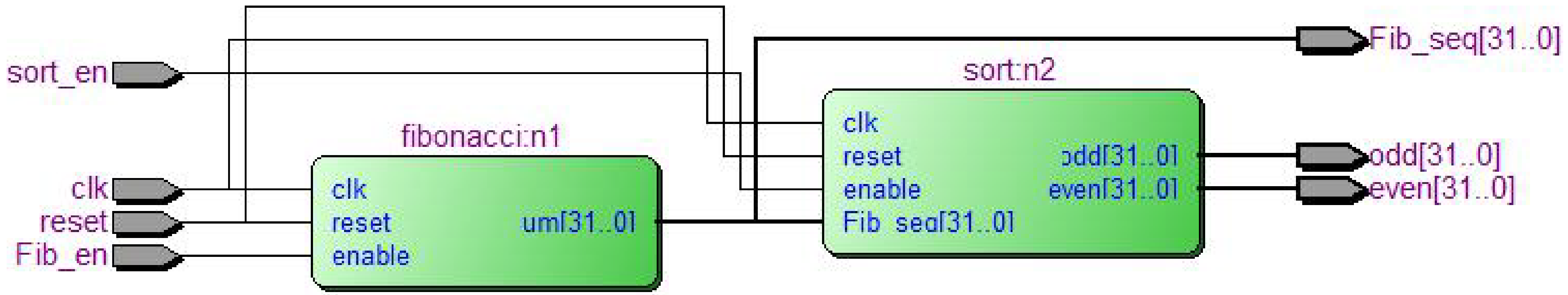
```verilog
module fibonacci(
    input clk,
    input reset,
    input enable,
    output reg [31:0] sum // sequence
);

reg [31:0] cur_num;
reg [31:0] nex_num;

always@(posedge clk or negedge reset) begin
    if(!reset)begin
        sum <= 0;
    end
    else if(enable) begin
        if(clk)begin
            sum <= cur_num + nex_num;
        end
    end
end

always@(negedge clk or negedge reset)begin
    if(!reset)begin
        cur_num <= 0;
        nex_num <= 1;
    end
    else if(enable)begin
        if(!clk)begin
            cur_num <= nex_num;
            nex_num <= sum;
        end
    end
end

endmodule

```

Sort



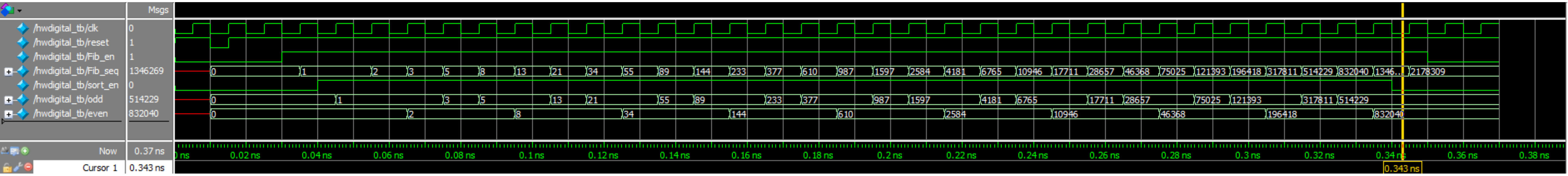
```
module sort(
    input clk,
    input reset,
    input enable, // sort enable
    input [31:0] Fib_seq,
    output reg[31:0] odd,
    output reg[31:0] even
);

always@(posedge clk or negedge reset)begin
    if(!reset)begin
        odd <= 0;
        even <= 0;
    end
    else if(enable)begin
        if(clk)begin
            if(Fib_seq & 1)begin
                odd <= Fib_seq;
            end
            else begin
                even <= Fib_seq;
            end
        end
    end
end

endmodule
```

RTL 圖示



sort_en
clk
reset
Fib_en
fibonacci:n1
clk
reset
enable
um[31..0]
sort:n2
clk
reset
enable
Fib_seq[31..0]
odd[31..0]
even[31..0]
Fib_seq[31..0]
odd[31..0]
even[31..0]

波形模擬圖



```
                      Msgs
/hwdigital_tb/clk      0
/hwdigital_tb/reset    1
/hwdigital_tb/Fib_en   1
/hwdigital_tb/Fib_seq  1346269
/hwdigital_tb/sort_en  0
/hwdigital_tb/odd      514229
/hwdigital_tb/even     832040

Now       0.37 ns
Cursor 1  0.343 ns
```