SOC Design

From Verilog to HLS – An Example

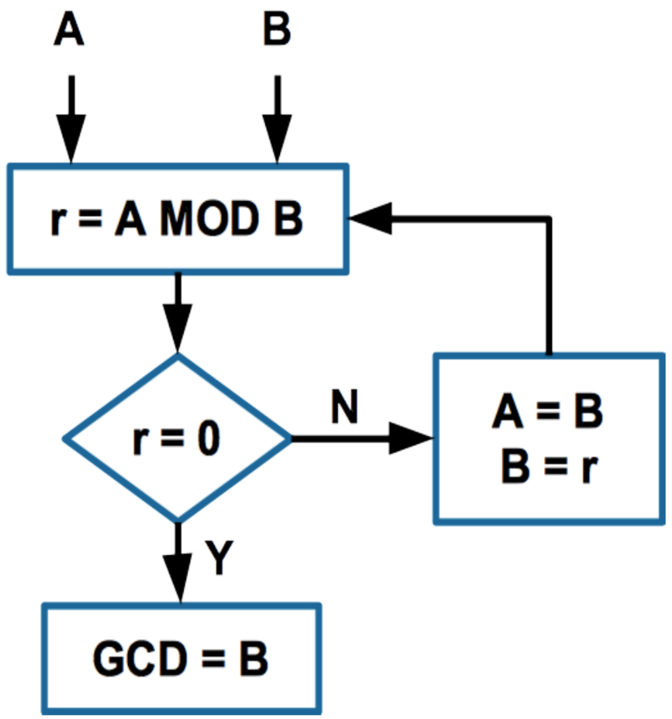
Jiin Lai

An Example - GCD

Euclidean Algorithm

$$\gcd(a,b) = \gcd(b,r)$$

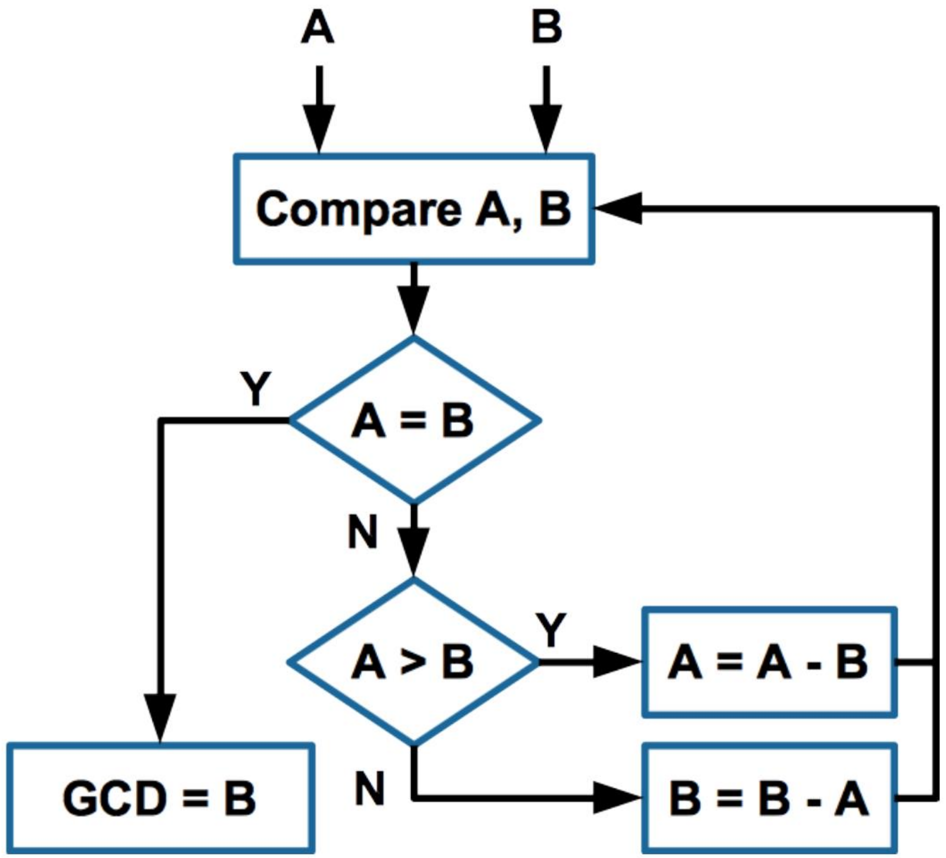
where, $a = qb + r$



An Example - GCD

Simplified Euclidean GCD Algorithm

$$\gcd(a,b) = \gcd(b,(a-b))$$
$$= \gcd(a,(b-a))$$



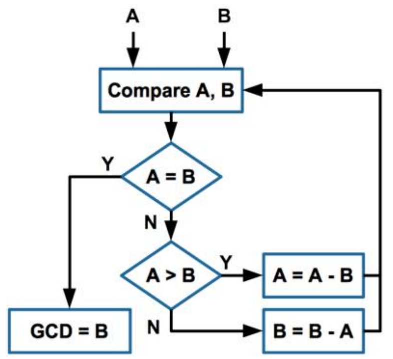
Verilog Behavior Implementation

- RTL synthesis tool only copies the circuit for the while/for loop
- But the # of loop could not be determined at compiling time
- **The circuit could not be synthesized**
- It needs a structure implementation

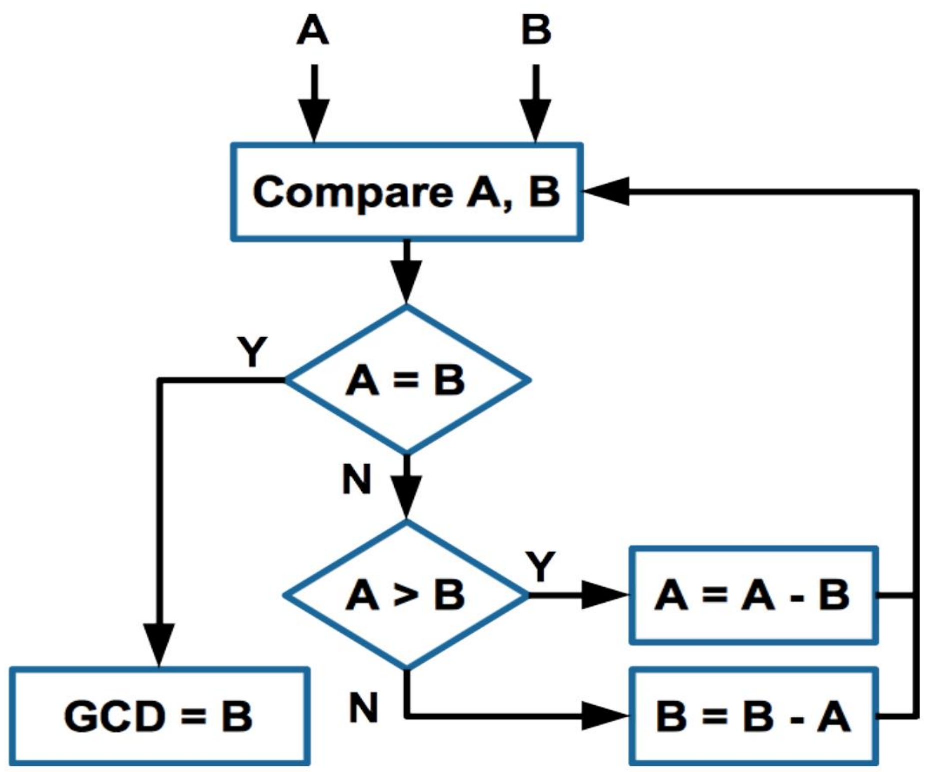
```
module gcd_behavior #(parameter width = 32)
            ( input  [width-1: 0] A_in, B_in,
              output [width-1:0] Y );
reg [width-1:0] A, B, Y, swap
Integer  done;

always @( A_in or B_in ) begin
  while ( A ! = B ) begin
    if( A > B )   A<= A – B;
    else          B<= B – A;
  end
end
Y = B;
endmodule
```

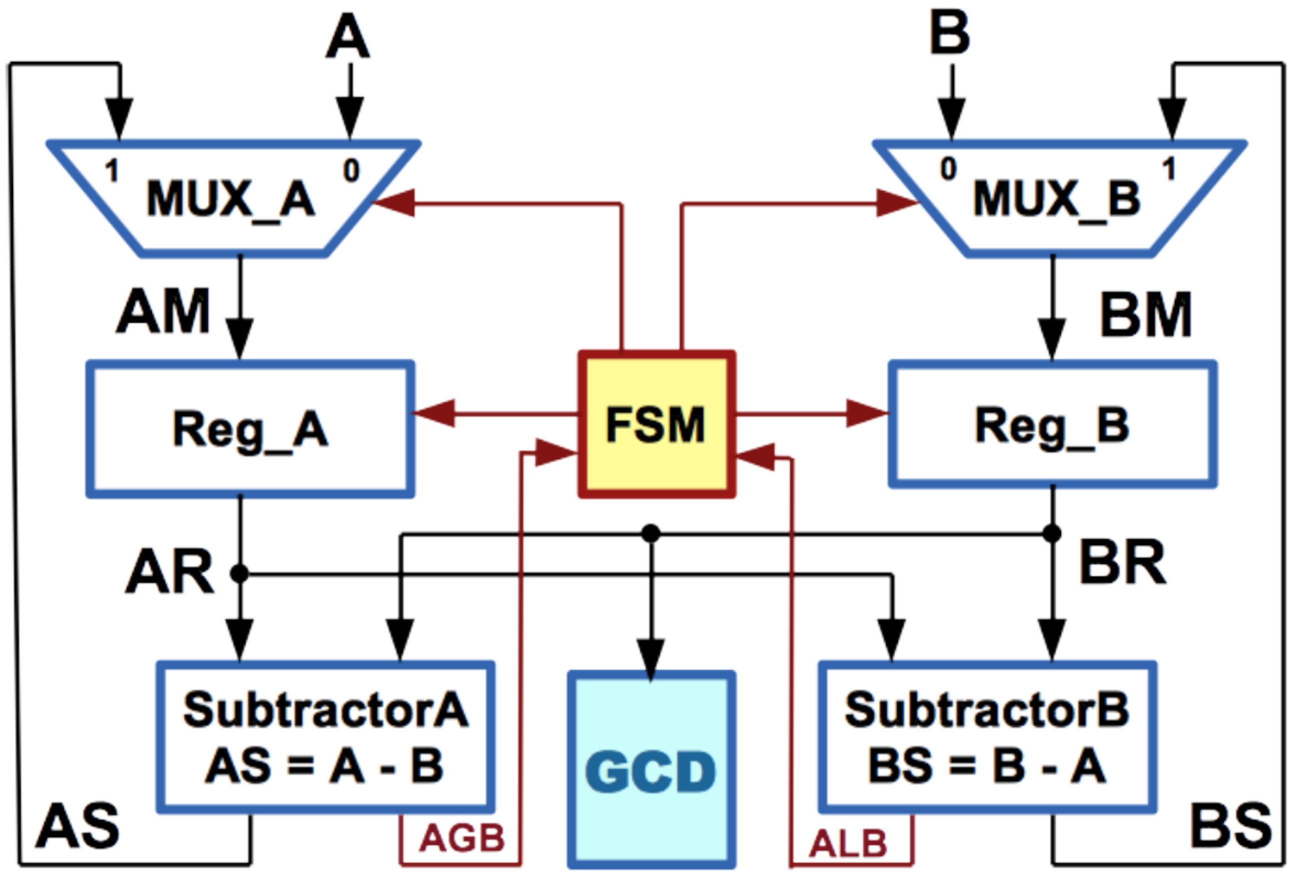
Simplified Euclidean GCD Algorithm

$$\begin{aligned} \gcd(a,b) &= \gcd(b,(a-b)) \\ &= \gcd(a,(b-a)) \end{aligned}$$



GCD Design Structure

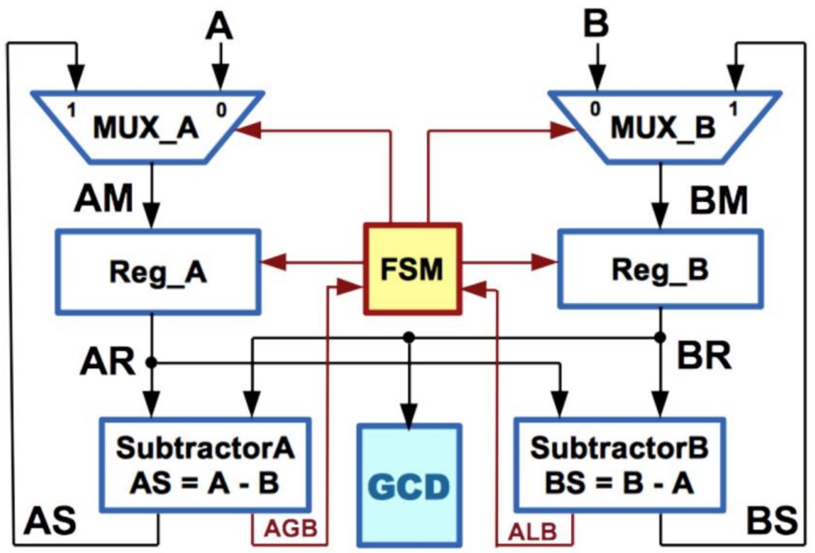




GCD Design Structure

Datapath

- Register/Latch
- Multiplexer
- Operator



```
module gcd_datapath #(parameter width = 16)
( input clock;
  input A_en, B_en, A_mux_sel, B_mux_sel, out_mux_sel;
  input [width-1:0] A_in, B_in;
  output AGB, ALB,
  output [width-1:0] Y; )

reg [width-1:0] A, B;
assign Y = A;

// Datapath Logic
wire [width-1:0] out =  ( out_mux_sel) ? B: A-B:;
wire [width-1:0] A_next = ( A_mux_sel ) ? out : A_in;
wire [width-1:0] B_next = ( B_mux_sel ) ? A : B_in;

// Generate output control signals
wire AGB = ( A > B);
wire ALB =  (A < B);

// edge-triggered flip-flop
always @( posedge clock) begin
   if( A_en )    A <= A_next;
   if (B_en)     B <= B_next;
end
endmodule
```

control

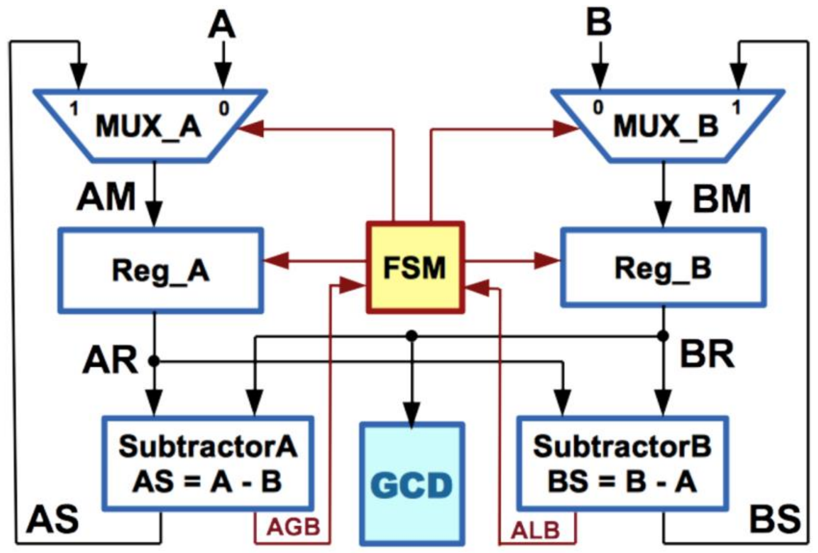
multiplexer

operator

Registers/latch

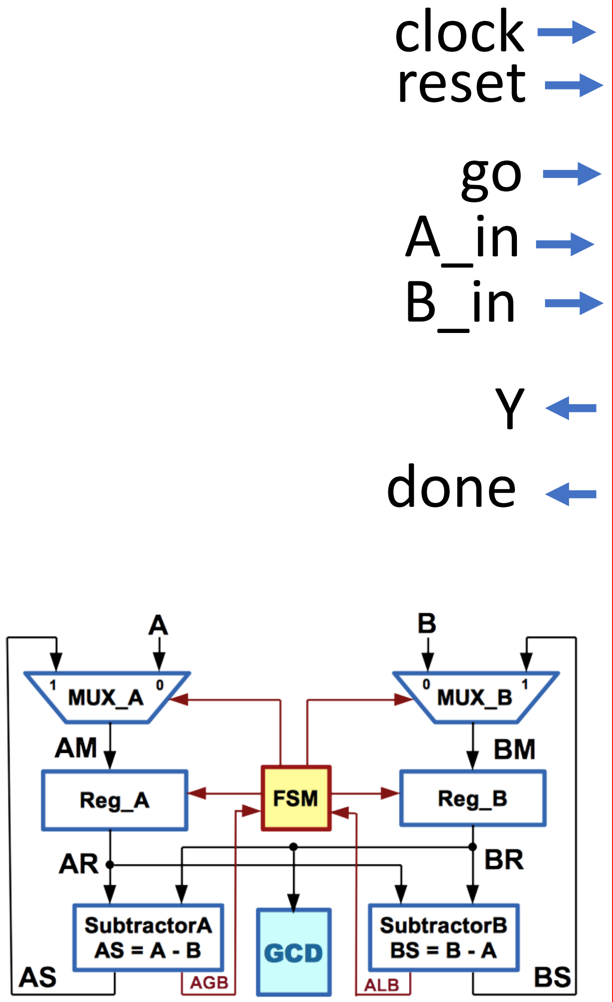
©BOLEDU

GCD Design Structure - Control



```
module gcd_fsm(
            input  clock, reset, go,
            input  AGB, ALB,
            output  A_en, B_en,
            A_mux_sel, B_mux_sel,  out_mux_sel, ouput done );
reg running = 0;
always @( posedge clock) begin
   if( go )    running <= 1;
   else if (done)  running <= 0;
end
reg [5:0] ctrl_sig;
assign { A_en, B_en,  A_mux_sel, B_mux_sel,  done } = ctrl_sig;
  always @(*)  begin
     if( !running )    ctrl_sig =            5’b11_00_0;
     else if( AGB )    ctrl_sig =            5’b10_1x_0;
     else if( ALB  )   ctrl_sig =            5’b11_11_0;
     else              ctrl_sig =            5’b00_xx_1;
  end
endmodule
```

GCD Design Verilog – Put Together



```
module gcd_fsm(
                    input  clock, reset, go,
                    input  AGB, ALB,
                    output  A_en, B_en,
                    A_mux_sel, B_mux_sel,
                    out_mux_sel, ouput done );
reg running = 0;
always @( posedge clock) begin
   if( go )    running <= 1;
   else if (done)  running <= 0;
end
reg [5:0] ctrl_sig;
assign { A_en, B_en, A_mux_sel, B_mux_sel, done }
= ctrl_sig;
  always @(*)  begin
     if( !running )   ctrl_sig = 5'b11_00_0;
     else if( AGB )   ctrl_sig = 5'b10_1x_0;
     else if( ALB  )  ctrl_sig = 5'b11_11_0;
     else             ctrl_sig = 5'b00_xx_1;
  end
endmodule
```

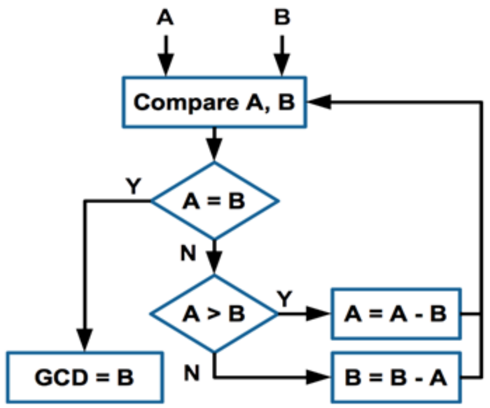
A_en,
B_en,
A_mux_sel
B_mux_sel
out_mux_sel

AGB
ALB

```
module gcd_datapath #(parameter width = 16)
                    ( input clock,
                     input A_en, B_en, A_mux_sel, B_mux_sel,
                     out_mux_sel,
                     input [width-1:0] A_in, B_in;
                     output AGB, ALB,
                     output [width-1:0] Y; )
reg [width-1:0] A, B;
assign Y = A;

// Datapath Logic
wire [width-1:0] out =  ( out_mux_sel) ? B: A-B:;
wire [width-1:0] A_next = ( A_mux_sel ) ? out : A_in;
wire [width-1:0] B_next = ( B_mux_sel ) ? A : B_in;

// Generate output control signals
wire AGB = ( A > B);
wire ALB =  (A < B);

// edge-triggered flip-flop
always @( posedge clock) begin
   if( A_en )   A <= A_next;
   if (B_en)    B <= B_next;
end
endmodule
```

A Glimpse of High-Level-Synthesis

- HLS build synchronous design
  - No timing -> no clock, reset
  - No port width – imply by data type
  - Port direction – lhs, rhs
    - Input: only read, "pass by value"
    - Ouptut: function return, a reference, or a pointer
    - Inout: a reference or a pointer
- Loop:
  - Automatic control/datapath synthesis



```
ap_uint<32> gcd(
  ap_uint<32> opA,
  ap_uint<32> opB ) {

#pragma HLS INLINE

  while ( opA != opB ) {
       #pragma HLS PIPELINE
       if ( opA > opB )
           opA = opA - opB;
       else
           opB = opB - opA;
  }
       return opA;
}
```

Matches its original algorithmic description