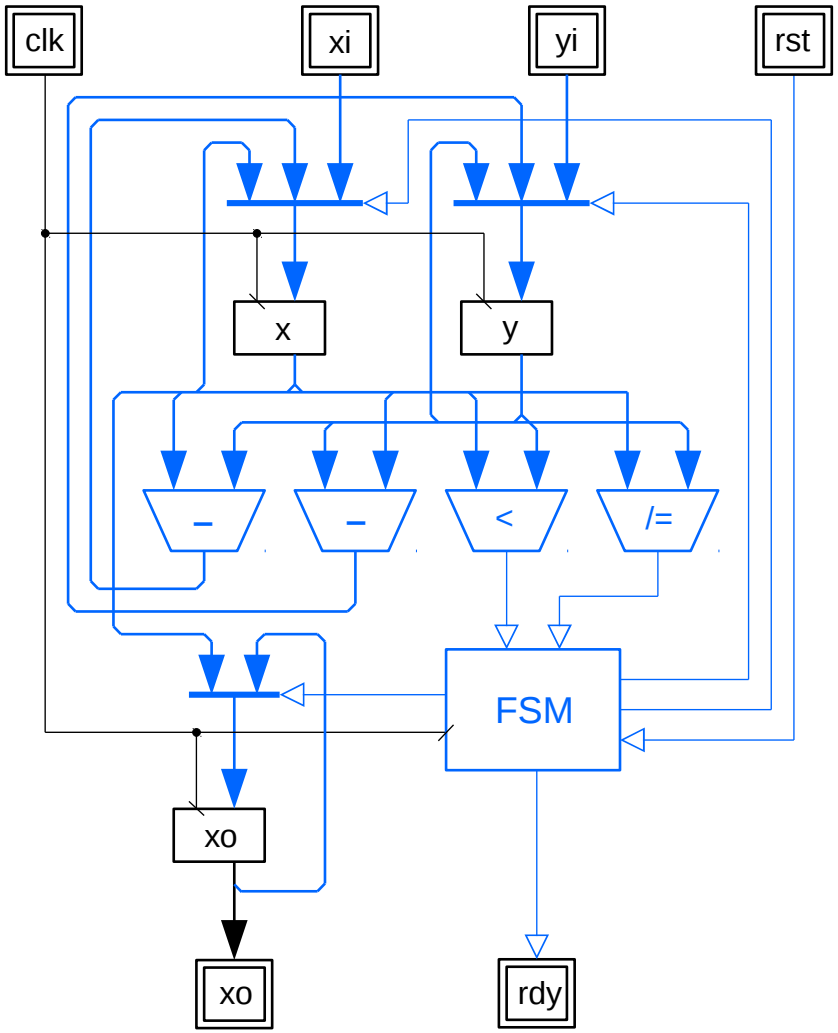
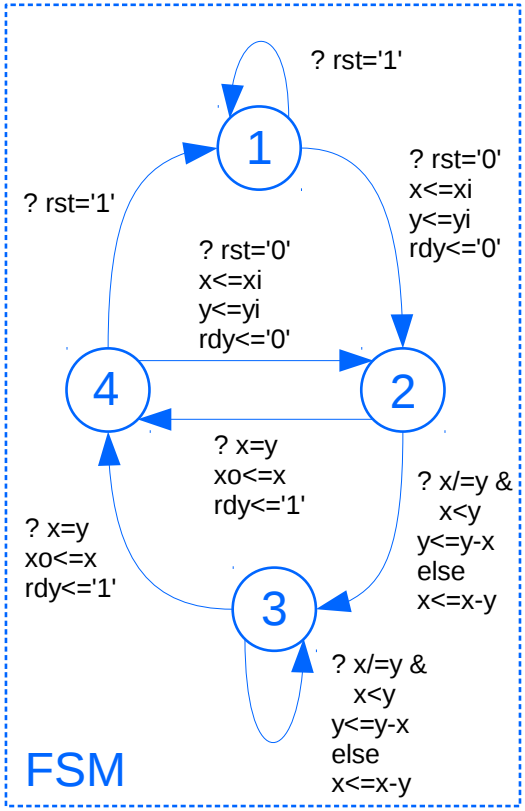
Clocked behavioral (gcd-bhvc.vhdl)

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;

entity gcd is
  port (xi, yi : in unsigned(15 downto 0);
        rst    : in bit;
        xo     : out unsigned(15 downto 0);
        rdy    : out bit;
        clk    : in bit);
end gcd;

architecture experiments of gcd is
  signal x, y: unsigned(15 downto 0);
begin

  process begin
    -- Wait for the new input data
    while  rst = '1'  loop
      wait on clk until clk='1';  -- 1
    end loop;

    x <= xi;    y <= yi;    rdy <= '0';
    wait on clk until clk='1';  -- 2

    -- Calculate
    while  x /= y  loop
      if  x < y  then y <= y - x;
      else            x <= x - y;    end if;
      wait on clk until clk='1';  -- 3
    end loop;

    -- Ready
    xo <= x;    rdy <= '1';
    wait on clk until clk='1';  -- 4
  end process;

end experiments;
```



Only registers (signals) are described explicitly.

The rest (datapath & FSM) are described implicitly – subject for interpretations by synthesizers.

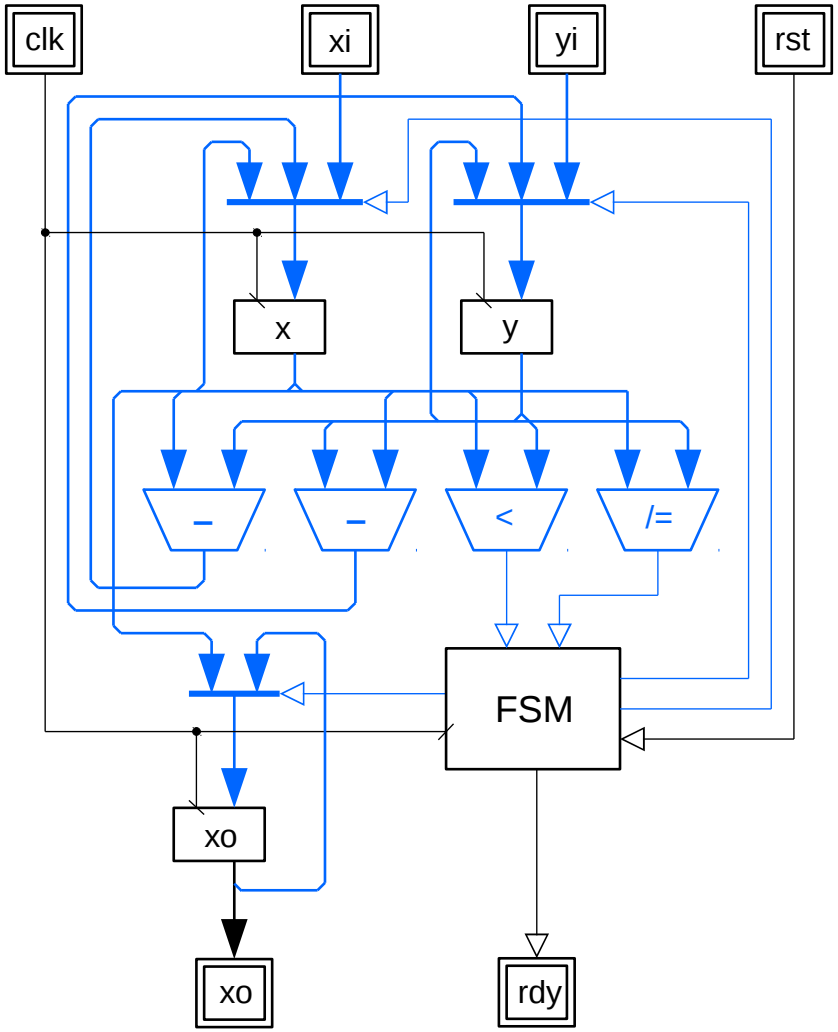


2 subtracters, 2 comparators
[1 clock step per iteration]

ASIC: 961 e.g. / 20.0 ns
FPGA: not synthesizable

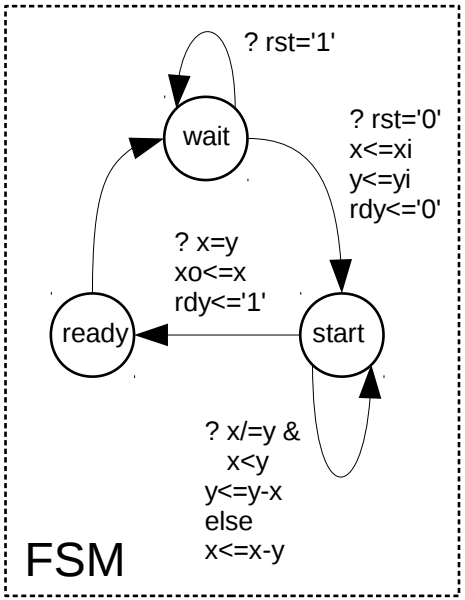
Behavioral state machine (gcd-bfsm.vhdl)

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;

entity gcd is
  port (xi, yi : in unsigned(15 downto 0);
        rst    : in bit;
        xo     : out unsigned(15 downto 0);
        rdy    : out bit;
        clk    : in bit);
end gcd;

architecture experiments of gcd is
  type state_type is (S_wait, S_start, S_ready);
  signal state: state_type;
  signal x, y: unsigned(15 downto 0);
begin

  process begin
    wait on clk until clk='1';
    case state  is
    -- Wait for the new input data
    when S_wait =>
      if  rst='0'  then
        x <= xi;  y <= yi;  rdy <= '0';
        state <= S_start;
      end if;
    -- Calculate
    when S_start =>
      if  x /= y  then
        if  x < y  then  y <= y - x;
        else             x <= x - y;  end if;
        state <= S_start;
      else
        xo <= x;  rdy <= '1';  state <= S_ready;
      end if;
    -- Ready
    when S_ready =>    state <= S_wait;
    end case;
  end process;

end experiments;
```



Registers (signals) and FSM are described explicitly.

The datapath is described implicitly – subject for interpretations by synthesizers.



2 subtracters, 2 comparators
[1 clock step per iteration]

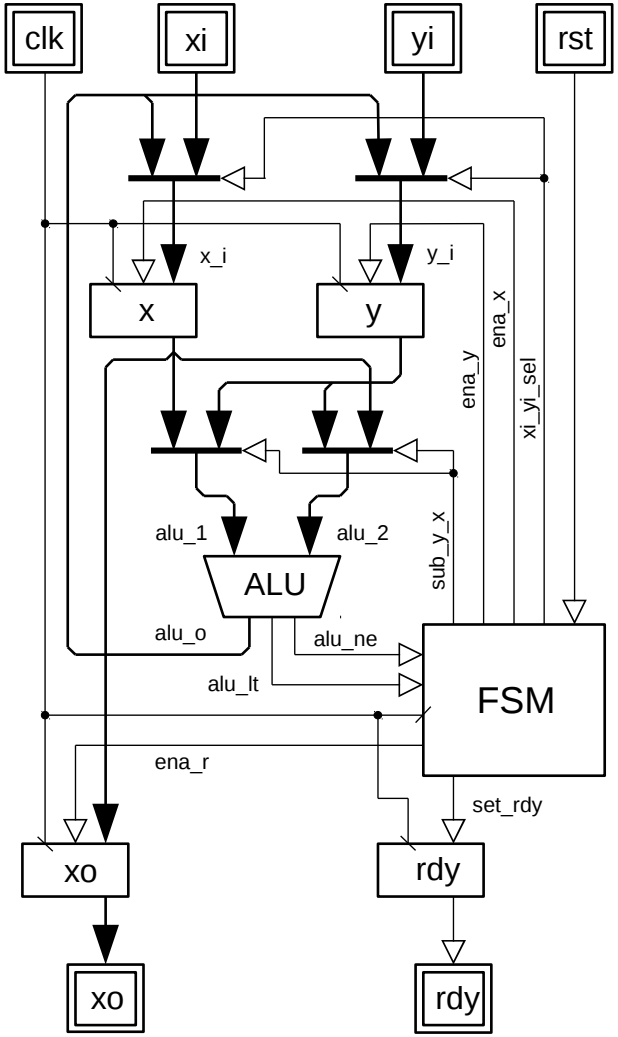
ASIC: 911 e.g. / 19.4 ns
FPGA: 108 SLC / 9.9 ns

RTL #1: (gcd-rtl1.vhdl)

```
architecture experiments of gcd is
  type state_type is (S_wait, S_start, S_comp, S_sub_x_y, S_sub_y_x, S_ready);
  signal state, next_state: state_type;
  signal x, y: unsigned(15 downto 0);
  signal alu_1, alu_2, alu_o, x_i, y_i: unsigned(15 downto 0);
  signal alu_lt, alu_ne, ena_x, ena_y, ena_r, set_rdy: bit;
  signal xi_yi_sel, sub_y_x: bit;
begin
-- Next state function of the FSM
  process (state, rst, alu_ne, alu_lt) begin
    ena_x <= '0';    ena_y <= '0';    ena_r <= '0';
    set_rdy <= '0';    xi_yi_sel <= '0';    sub_y_x <= '0';
    next_state <= state;
    case  state  is
    -- Wait for the new input data
    when S_wait =>
      if  rst='0'  then
        xi_yi_sel <= '1';  ena_x <= '1';  ena_y <= '1';  next_state <= S_start;
      end if;
    -- Loop: ready?
    when S_start =>
      if  alu_ne='1'  then  next_state <= S_comp;
      else                  next_state <= S_ready;    end if;
    -- Loop: compare
    when S_comp =>
      if  alu_lt='1'  then    next_state <= S_sub_y_x;
      else                    next_state <= S_sub_x_y;  end if;
    -- Loop: y-x
    when S_sub_y_x =>  ena_y <= '1';  sub_y_x <= '1';  next_state <= S_start;
    -- Loop: x-y
    when S_sub_x_y =>  ena_x <= '1';  sub_y_x <= '0';  next_state <= S_start;
    -- Ready
    when S_ready =>    ena_r <= '1';  set_rdy <= '1';  next_state <= S_wait;
    end case;
  end process;
-- ALU: subtract / less-than / not-equal
  alu_o <= alu_1 - alu_2;
  alu_lt <= to_bit(alu_o(15));
  process (alu_o)
    variable or_tmp: unsigned(15 downto 0);
  begin
    or_tmp(15) := alu_o(15);
    for  i in 14 downto 0  loop
      or_tmp(i) := or_tmp(i+1) or alu_o(i);
    end loop;
    alu_ne <= to_bit(or_tmp(0));
  end process;
-- Multiplexers
  x_i <= xi when xi_yi_sel='1' else alu_o;
  y_i <= yi when xi_yi_sel='1' else alu_o;
  alu_1 <= y when sub_y_x='1' else x;
  alu_2 <= x when sub_y_x='1' else y;
-- Registers
  process begin
    wait on clk until clk='1';
    state <= next_state;
    if  ena_x='1'  then    x <= x_i;    end if;
    if  ena_y='1'  then    y <= y_i;    end if;
    if  ena_r='1'  then    xo <= x;    end if;
    rdy <= set_rdy;
  end process;
end experiments;
```

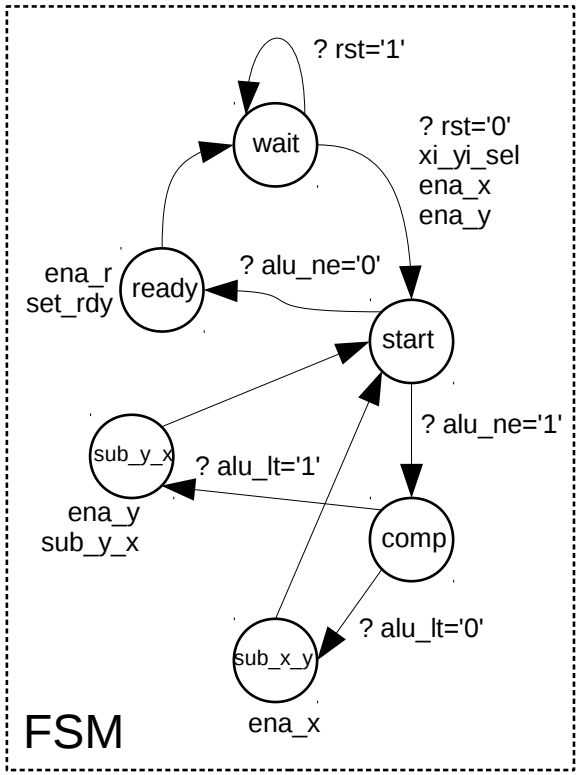
Single ALU (“–”, “<”, “/=”)
[3 clock steps per iteration]

ASIC: 986 e.g. / 19.8 ns
FPGA: 50 SLC / 10.8 ns



Register-transfer level description with universal ALU (operation reuse).

Default value for the control signals is '0'

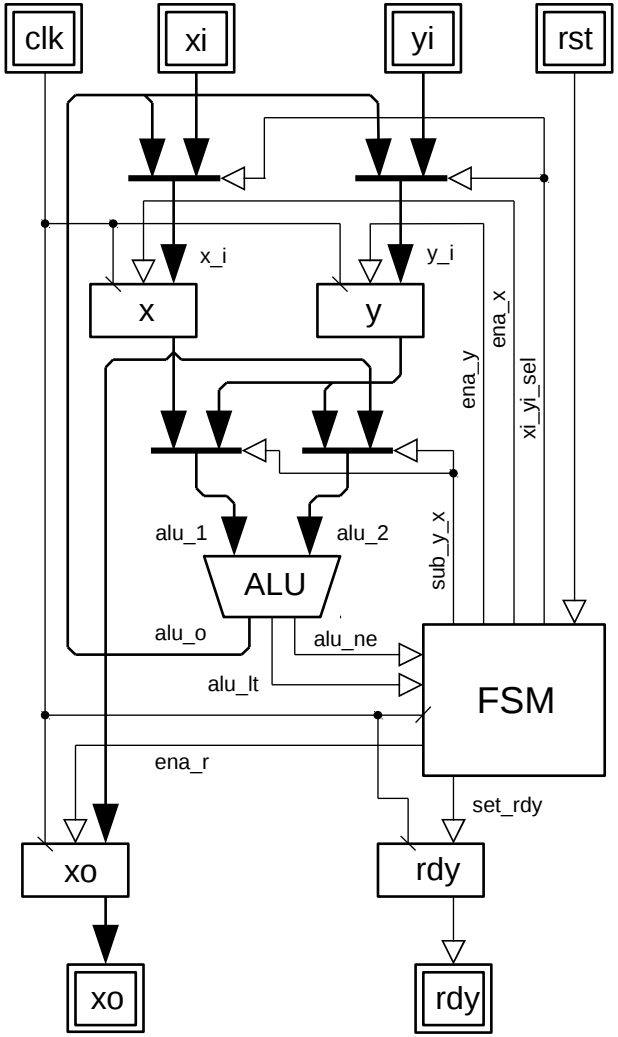


RTL #2: (gcd-rtl2.vhdl)

```
architecture experiments of gcd is
  type state_type is (S_wait, S_start, S_sub_x_y, S_sub_y_x, S_ready);
  signal state, next_state: state_type;
  signal x, y: unsigned(15 downto 0);
  signal alu_1, alu_2, alu_o, x_i, y_i: unsigned(15 downto 0);
  signal alu_lt, alu_ne, ena_x, ena_y, ena_r, set_rdy: bit;
  signal xi_yi_sel, sub_y_x: bit;
begin

-- Next state function of the FSM
  process (state, rst, alu_ne, alu_lt) begin
    ena_x <= '0';    ena_y <= '0';    ena_r <= '0';
    set_rdy <= '0';    xi_yi_sel <= '0';    sub_y_x <= '0';
    next_state <= state;
    case state  is
    -- Wait for the new input data
    when S_wait =>
      if  rst='0'  then
        xi_yi_sel <= '1';  ena_x <= '1';  ena_y <= '1';  next_state <= S_start;
      end if;
    -- Loop: ready?
    when S_start =>
      if  alu_ne='1'  then
        if  alu_lt='1'  then    next_state <= S_sub_y_x;
        else                    next_state <= S_sub_x_y;    end if;
      else                      next_state <= S_ready;    end if;
    -- Loop: y-x
    when S_sub_y_x =>  ena_y <= '1';  sub_y_x <= '1';  next_state <= S_start;
    -- Loop: x-y
    when S_sub_x_y =>  ena_x <= '1';  sub_y_x <= '0';  next_state <= S_start;
    -- Ready
    when S_ready =>    ena_r <= '1';  set_rdy <= '1';  next_state <= S_wait;
    end case;
  end process;

-- ALU: subtract / less-than / not-equal
  alu_o <= alu_1 - alu_2;
  alu_lt <= to_bit(alu_o(15));
  process (alu_o)
    variable or_tmp: unsigned(15 downto 0);
  begin
    or_tmp(15) := alu_o(15);
    for  i in 14 downto 0  loop
      or_tmp(i) := or_tmp(i+1) or alu_o(i);
    end loop;
    alu_ne <= to_bit(or_tmp(0));
  end process;
-- Multiplexers
  x_i <= xi when xi_yi_sel='1' else alu_o;
  y_i <= yi when xi_yi_sel='1' else alu_o;
  alu_1 <= y when sub_y_x='1' else x;
  alu_2 <= x when sub_y_x='1' else y;
-- Registers
  process begin
    wait on clk until clk='1';
    state <= next_state;
    if  ena_x='1'  then    x <= x_i;    end if;
    if  ena_y='1'  then    y <= y_i;    end if;
    if  ena_r='1'  then    xo <= x;    end if;
    rdy <= set_rdy;
  end process;
end experiments;
```

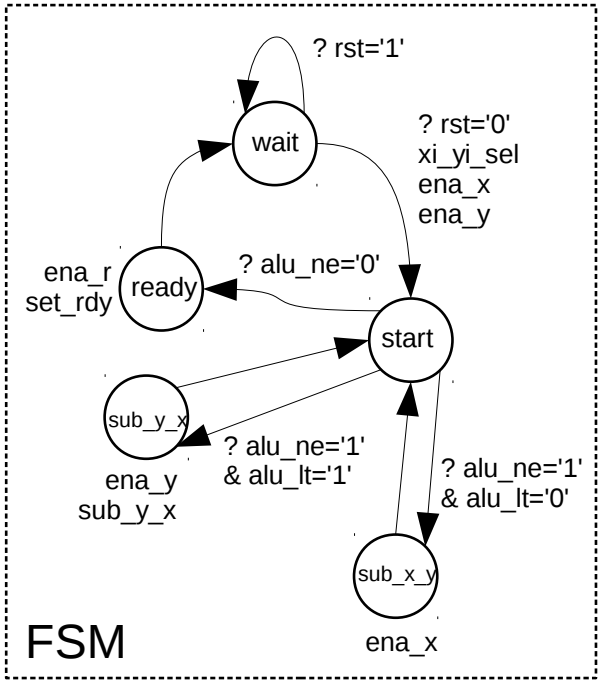
Single ALU (“–”, “<”, “/=”)
[2 clock steps per iteration]

ASIC: 931 e.g. / 19.9 ns
FPGA: 48 SLC / 10.8 ns



Register-transfer level description with universal ALU (operation reuse).

Default value for the control signals is '0'.

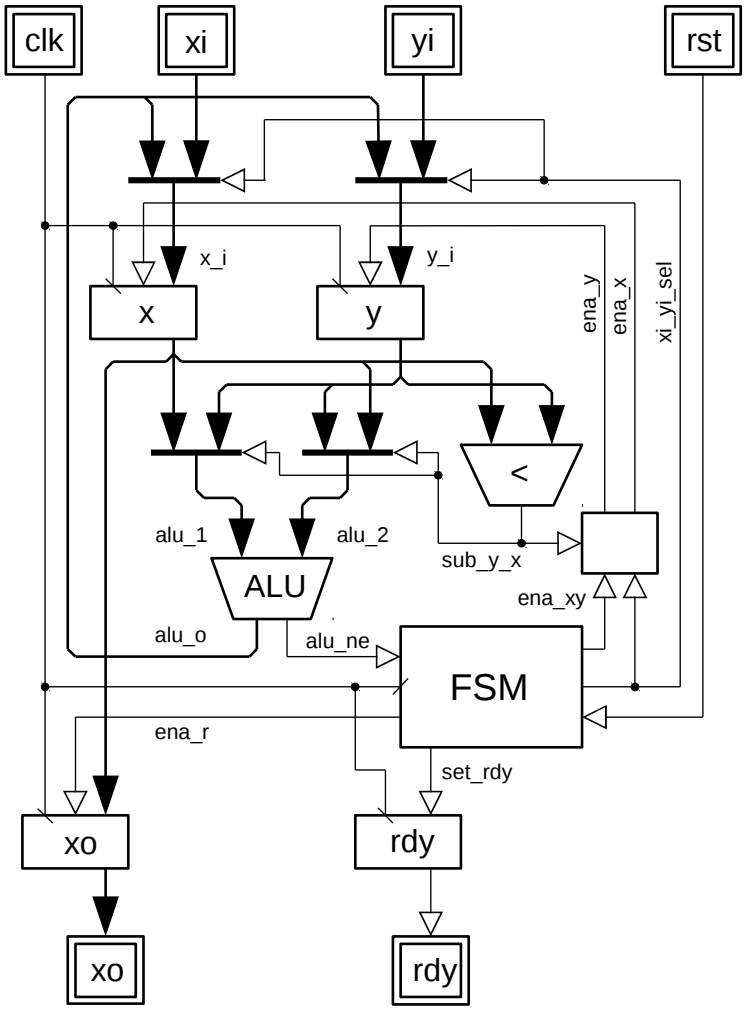


RTL #3: (gcd-rtl3.vhdl)

```
architecture experiments of gcd is
  type state_type is (S_wait, S_start, S_ready);
  signal state, next_state: state_type;
  signal x, y: unsigned(15 downto 0);
  signal alu_1, alu_2, alu_o, x_i, y_i: unsigned(15 downto 0);
  signal alu_ne, ena_xy, ena_x, ena_y, ena_r, set_rdy: bit;
  signal xi_yi_sel, sub_y_x: bit;
begin
-- Next state function of the FSM
  process (state, rst, alu_ne) begin
    ena_xy <= '0';  ena_r <= '0';  set_rdy <= '0';  xi_yi_sel <= '0';
    next_state <= state;
    case state  is
    -- Wait for the new input data
    when S_wait =>
      if  rst='0'  then
        xi_yi_sel <= '1';  ena_xy <= '1';  next_state <= S_start;
      end if;
    -- Calculate
    when S_start =>
      if  alu_ne='1'  then   ena_xy <= '1';   next_state <= S_start;
      else    ena_r <= '1';  set_rdy <= '1';  next_state <= S_ready;
      end if;
    -- Ready
    when S_ready =>
      ena_r <= '1';    set_rdy <= '1';    next_state <= S_wait;
    end case;
  end process;

-- Comparator (less-than)
  sub_y_x <= '1' when x < y else '0';
-- Subtracter (+not-equal)
  alu_o <= alu_1 - alu_2;
  process (alu_o)
    variable or_tmp: unsigned(15 downto 0);
  begin
    or_tmp(15) := alu_o(15);
    for  i in 14 downto 0  loop
      or_tmp(i) := or_tmp(i+1) or alu_o(i);
    end loop;
    alu_ne <= to_bit(or_tmp(0));
  end process;
-- Multiplexers
  x_i <= xi when xi_yi_sel='1' else alu_o;
  y_i <= yi when xi_yi_sel='1' else alu_o;
  alu_1 <= y when sub_y_x='1' else x;
  alu_2 <= x when sub_y_x='1' else y;
  ena_x <= '1' when (sub_y_x='0' and ena_xy='1') or xi_yi_sel='1' else '0';
  ena_y <= '1' when (sub_y_x='1' and ena_xy='1') or xi_yi_sel='1' else '0';
-- Registers
  process begin
    wait on clk until clk='1';
    state <= next_state;
    if  ena_x='1'  then    x <= x_i;    end if;
    if  ena_y='1'  then    y <= y_i;    end if;
    if  ena_r='1'  then    xo <= x;    end if;
    rdy <= set_rdy;
  end process;
end experiments;
```

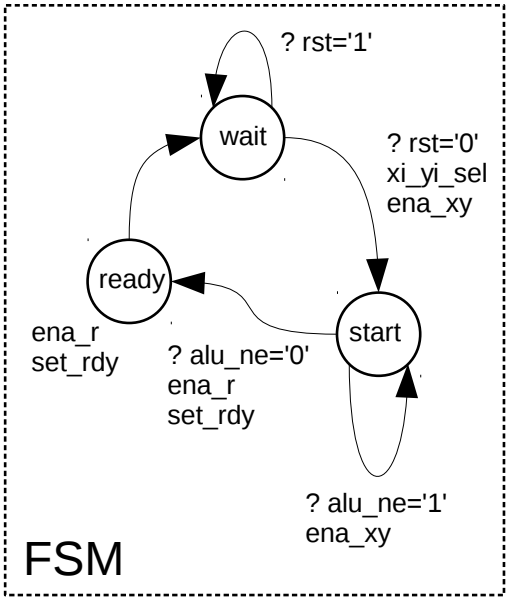
1 ALU (“–”, “/=”), 1 comparator
[1 clock step per iteration]

ASIC: 1134 e.g. / 20.0 ns
FPGA: 58 SLC / 17.0 ns



Register-transfer level description with comparator controlling subtractions.

Default value for the control signals is '0'.

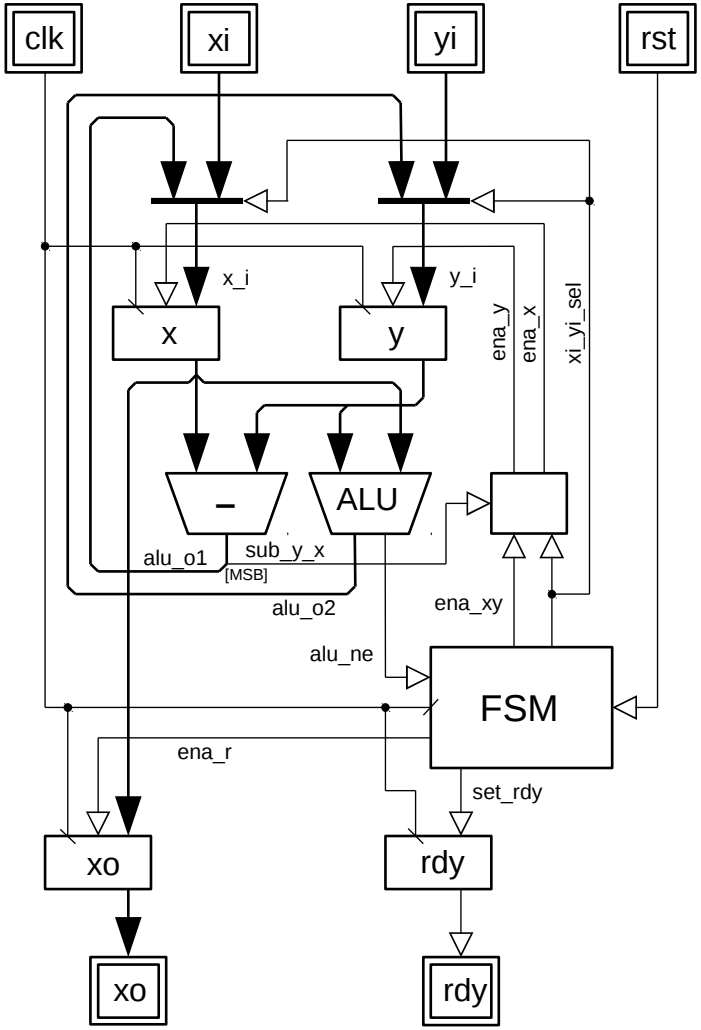


RTL #4: (gcd-rtl4.vhdl)

```
architecture experiments of gcd is
  type state_type is (S_wait, S_start, S_ready);
  signal state, next_state: state_type;
  signal x, y: unsigned(15 downto 0);
  signal alu_o1, alu_o2, x_i, y_i: unsigned(15 downto 0);
  signal alu_ne, ena_xy, ena_x, ena_y, ena_r, set_rdy: bit;
  signal xi_yi_sel, sub_y_x: bit;
begin
-- Next state function of the FSM
  process (state, rst, alu_ne) begin
    ena_xy <= '0';  ena_r <= '0';  set_rdy <= '0';  xi_yi_sel <= '0';
    next_state <= state;
    case state  is
    -- Wait for the new input data
    when S_wait =>
      if  rst='0'  then
        xi_yi_sel <= '1';  ena_xy <= '1';  next_state <= S_start;
      end if;
    -- Calculate
    when S_start =>
      if  alu_ne='1'  then   ena_xy <= '1';   next_state <= S_start;
      else    ena_r <= '1';  set_rdy <= '1';  next_state <= S_ready;
      end if;
    -- Ready
    when S_ready =>
      ena_r <= '1';    set_rdy <= '1';    next_state <= S_wait;
    end case;
  end process;

-- Subtracter (x-y) / comparator (x<y)
  alu_o1 <= x - y;
  sub_y_x <= '1' when alu_o1(alu_o1'high)='1' else '0';
-- Subtracter (y-x) / comparator (y/=x)
  alu_o2 <= y - x;
  process (alu_o2)
    variable or_tmp: unsigned(15 downto 0);
  begin
    or_tmp(15) := alu_o2(15);
    for  i in 14 downto 0  loop
      or_tmp(i) := or_tmp(i+1) or alu_o2(i);
    end loop;
    alu_ne <= to_bit(or_tmp(0));
  end process;

-- Multiplexers
  x_i <= xi when xi_yi_sel='1' else alu_o1;
  y_i <= yi when xi_yi_sel='1' else alu_o2;
  ena_x <= '1' when (sub_y_x='0' and ena_xy='1') or xi_yi_sel='1' else '0';
  ena_y <= '1' when (sub_y_x='1' and ena_xy='1') or xi_yi_sel='1' else '0';
-- Registers
  process begin
    wait on clk until clk='1';
    state <= next_state;
    if  ena_x='1'  then    x <= x_i;    end if;
    if  ena_y='1'  then    y <= y_i;    end if;
    if  ena_r='1'  then    xo <= x;    end if;
    rdy <= set_rdy;
  end process;
end experiments;
```

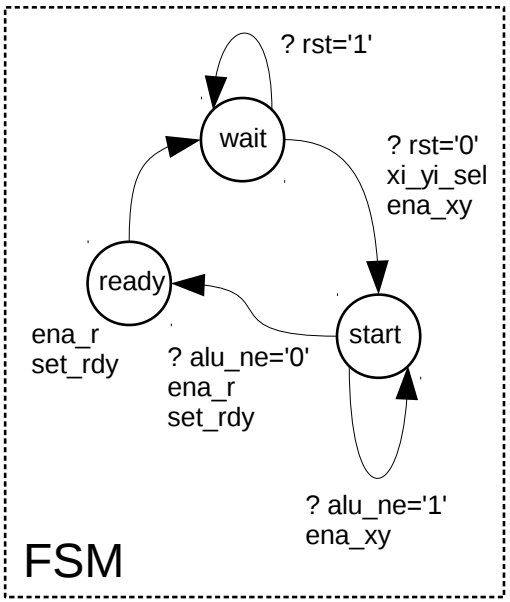
1 subtracter, 1 ALU (“–”, “/=”)
[1 clock step per iteration]

ASIC: 976 e.g. / 19.9 ns
FPGA: 78 SLC / 12.6 ns



Register-transfer level description with out-of-order subtractions. Only data-path differs from RTL #3.

Default value for the control signals is '0'.

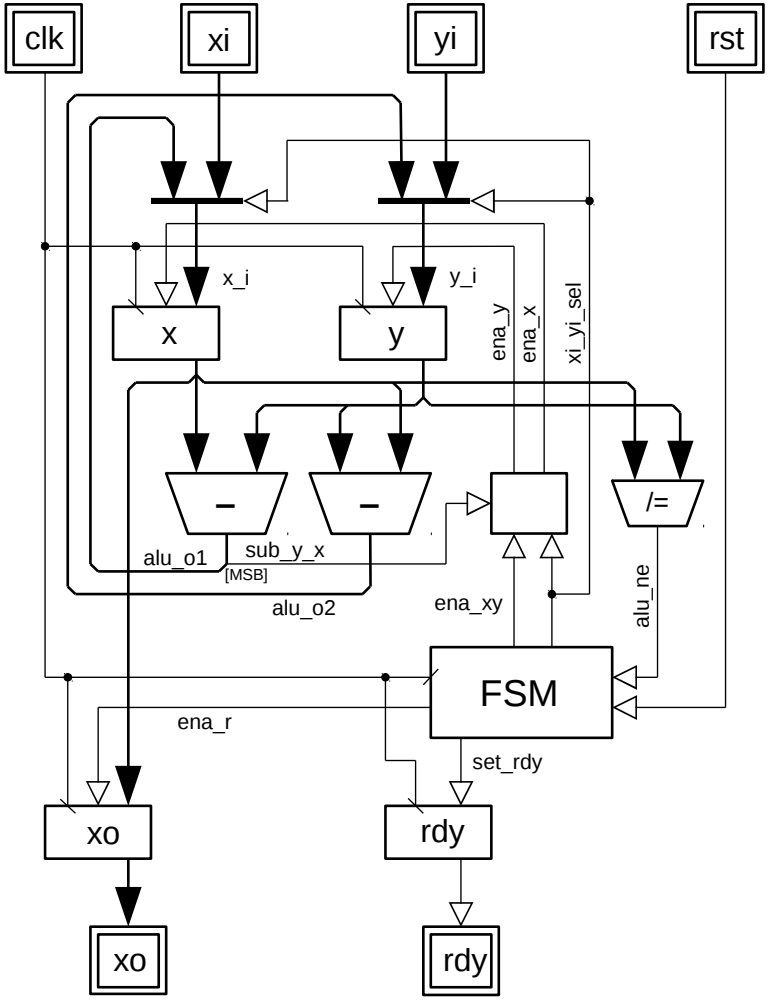


RTL #5: (gcd-rtl5.vhdl)

```
architecture experiments of gcd is
  type state_type is (S_wait, S_start, S_ready);
  signal state, next_state: state_type;
  signal x, y: unsigned(15 downto 0);
  signal alu_o1, alu_o2, x_i, y_i: unsigned(15 downto 0);
  signal alu_ne, ena_xy, ena_x, ena_y, ena_r, set_rdy: bit;
  signal xi_yi_sel, sub_y_x: bit;
begin
-- Next state function of the FSM
  process (state, rst, alu_ne) begin
    ena_xy <= '0';  ena_r <= '0';  set_rdy <= '0';  xi_yi_sel <= '0';
    next_state <= state;
    case state  is
    -- Wait for the new input data
    when S_wait =>
      if  rst='0'  then
        xi_yi_sel <= '1';  ena_xy <= '1';  next_state <= S_start;
      end if;
    -- Calculate
    when S_start =>
      if  alu_ne='1'  then   ena_xy <= '1';   next_state <= S_start;
      else    ena_r <= '1';  set_rdy <= '1';  next_state <= S_ready;
      end if;
    -- Ready
    when S_ready =>
      ena_r <= '1';    set_rdy <= '1';    next_state <= S_wait;
    end case;
  end process;

-- Subtracter (x-y) / comparator (x<y)
  alu_o1 <= x - y;
  sub_y_x <= '1' when alu_o1(alu_o1'high)='1' else '0';

-- Subtracter (y-x)
  alu_o2 <= y - x;



--  Comparator (y/=x)
  alu_ne <= '1' when x /= y else '0';



-- Multiplexers
  x_i <= xi when xi_yi_sel='1' else alu_o1;
  y_i <= yi when xi_yi_sel='1' else alu_o2;
  ena_x <= '1' when (sub_y_x='0' and ena_xy='1') or xi_yi_sel='1' else '0';
  ena_y <= '1' when (sub_y_x='1' and ena_xy='1') or xi_yi_sel='1' else '0';
-- Registers
  process begin
    wait on clk until clk='1';
    state <= next_state;
    if  ena_x='1'  then    x <= x_i;    end if;
    if  ena_y='1'  then    y <= y_i;    end if;
    if  ena_r='1'  then    xo <= x;    end if;
    rdy <= set_rdy;
  end process;
end experiments;
```

2 subtracters, 1 comparator
[1 clock step per iteration]

ASIC: 915 e.g. / 20.0 ns
FPGA: 58 SLC / 8.0 ns



Register-transfer level description with out-of-order subtractions. Only data-path differs from RTL #3 & #4.

Default value for the control signals is '0'.