

# INF5430 V2012

## Register Transfer Methodology II

P.P.Chu

RTL Hardware Design Using VHDL  
Chapter 12.1-5

# Outline

1. Design example: One-shot pulse generator
2. Design example: GCD

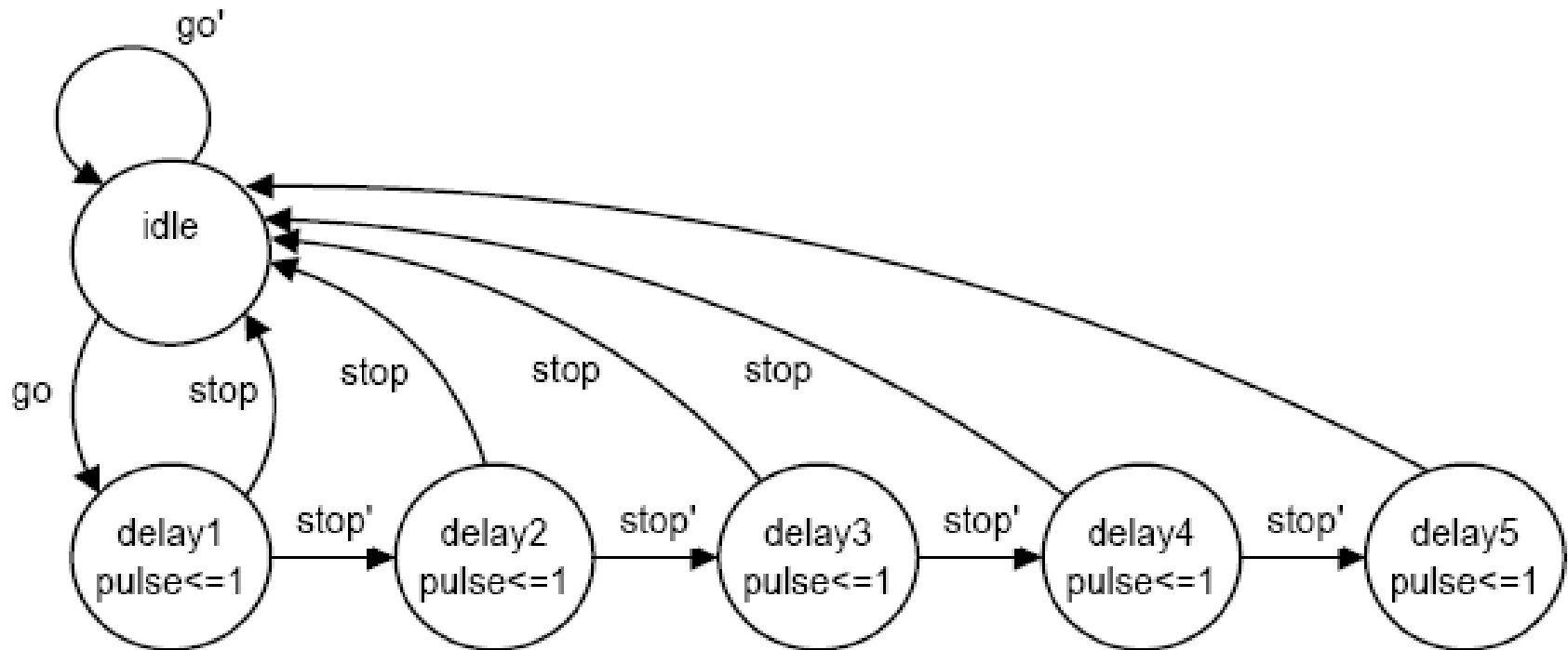
Self study: SRAM Interface Controller  
UART

# 1. One-shot pulse generator

- Sequential circuit divided into
  - Regular sequential circuit: w/ regular next-state logic
  - FSM: w/ random next-state logic
  - FSMD: w/ both
- Some design can be coded in different types
- FSMD is most flexible
- One-shot pulse generator as an example

- One-shot pulse generator
  - I/O: Input: `go`, `stop`; Output: `pulse`
  - `go` is the trigger signal, usually asserted for only one clock cycle
  - During normal operation, assertion of `go` activates `pulse` for 5 clock cycles
  - If `go` is asserted again during this interval, it will be ignored
  - If `stop` is asserted during this interval, `pulse` will be cut short and return to 0

- FSM implementation



```

library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
entity pulse_5clk is
    port(
        clk, reset: in std_logic;
        go, stop: in std_logic;
        pulse: out std_logic
    );
end pulse_5clk;

architecture fsm_arch of pulse_5clk is
    type fsm_state_type is
        (idle, delay1, delay2, delay3, delay4, delay5);
    signal state_reg, state_next: fsm_state_type;
begin
    -- state register
    process(clk,reset)
    begin
        if (reset='1') then
            state_reg <= idle;
        elsif (clk'event and clk='1') then
            state_reg <= state_next;
        end if;
    end process;
end fsm_arch;

```

```

-- next-state logic & output logic
process(state_reg,go,stop)
begin
    pulse <= '0';
    case state_reg is
        when idle =>
            if go='1' then
                state_next <= delay1;
            else
                state_next <= idle;
            end if;
        when delay1 =>
            if stop='1' then
                state_next <=idle;
            else
                state_next <=delay2;
            end if;
            pulse <= '1';
        when delay2 =>
            if stop='1' then
                state_next <=idle;
            else

```

```

        state_next <=delay3;
    end if;
    pulse <= '1';
when delay3 =>
    if stop='1' then
        state_next <=idle;
    else
        state_next <=delay4;
    end if;
    pulse <= '1';
when delay4 =>
    if stop='1' then
        state_next <=idle;
    else
        state_next <=delay5;
    end if;
    pulse <= '1';
when delay5 =>
    state_next <=idle;
    pulse <= '1';
end case;
end process;
end fsm_arch;

```



- Regular sequential circuit implementation
  - Based on a mod-5 counter
  - Use a flag FF to indicate whether counter should be active
  - Code difficult to comprehend

```

architecture regular_seq_arch of pulse_5clk is
    constant P_WIDTH: natural := 5;
    signal c_reg, c_next: unsigned(3 downto 0);
    signal flag_reg, flag_next: std_logic;
begin
    — register
    process(clk, reset)
    begin
        if (reset='1') then
            c_reg <= (others=>'0');
            flag_reg <= '0';
        elsif (clk'event and clk='1') then
            c_reg <= c_next;
            flag_reg <= flag_next;
        end if;
    end process;

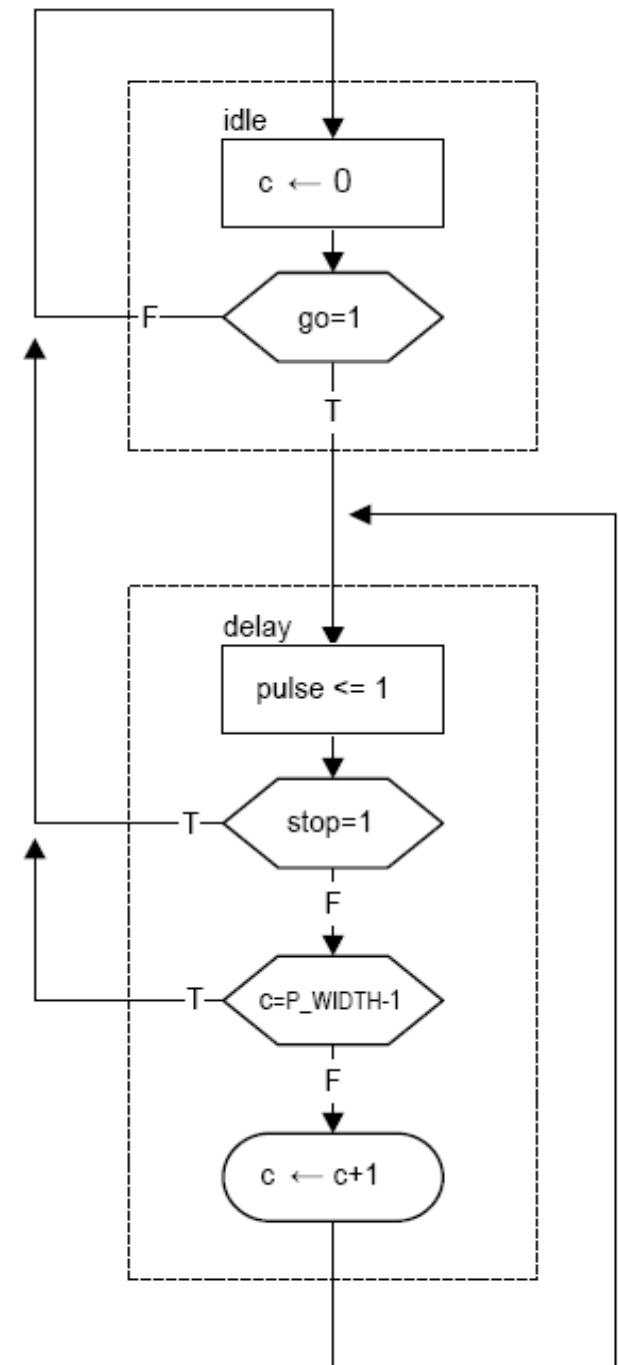
```

```

-- next-state logic
process(c_reg,flag_reg,go,stop)
begin
    c_next <= c_reg;
    flag_next <= flag_reg;
    if (flag_reg='0') and (go='1') then
        flag_next <= '1';
        c_next <= (others=>'0');
    ;
    elsif (flag_reg='1') and
        ((c_reg=P_WIDTH-1) or (stop='1')) then
        flag_next <= '0';
    elsif (flag_reg='1') then
        c_next <= c_reg + 1;
    ,
    end if ;
end process;
-- output logic
pulse <= '1' when flag_reg='1' else '0';
end regular_seq_arch;

```

- FSMD Implementation



```

architecture fsmd_arch of pulse_5clk is
    constant P_WIDTH: natural:= 5;
    type fsmd_state_type is (idle, delay);
    signal state_reg, state_next: fsmd_state_type;
    signal c_reg, c_next: unsigned(3 downto 0);
begin
    — state and data registers
    process(clk,reset)
    begin
        if (reset='1') then
            state_reg <= idle;
            c_reg <= (others=>'0');
        elsif (clk'event and clk='1') then
            state_reg <= state_next;
            c_reg <= c_next;
        end if;
    end process;
end architecture fsmd_arch;

```

```

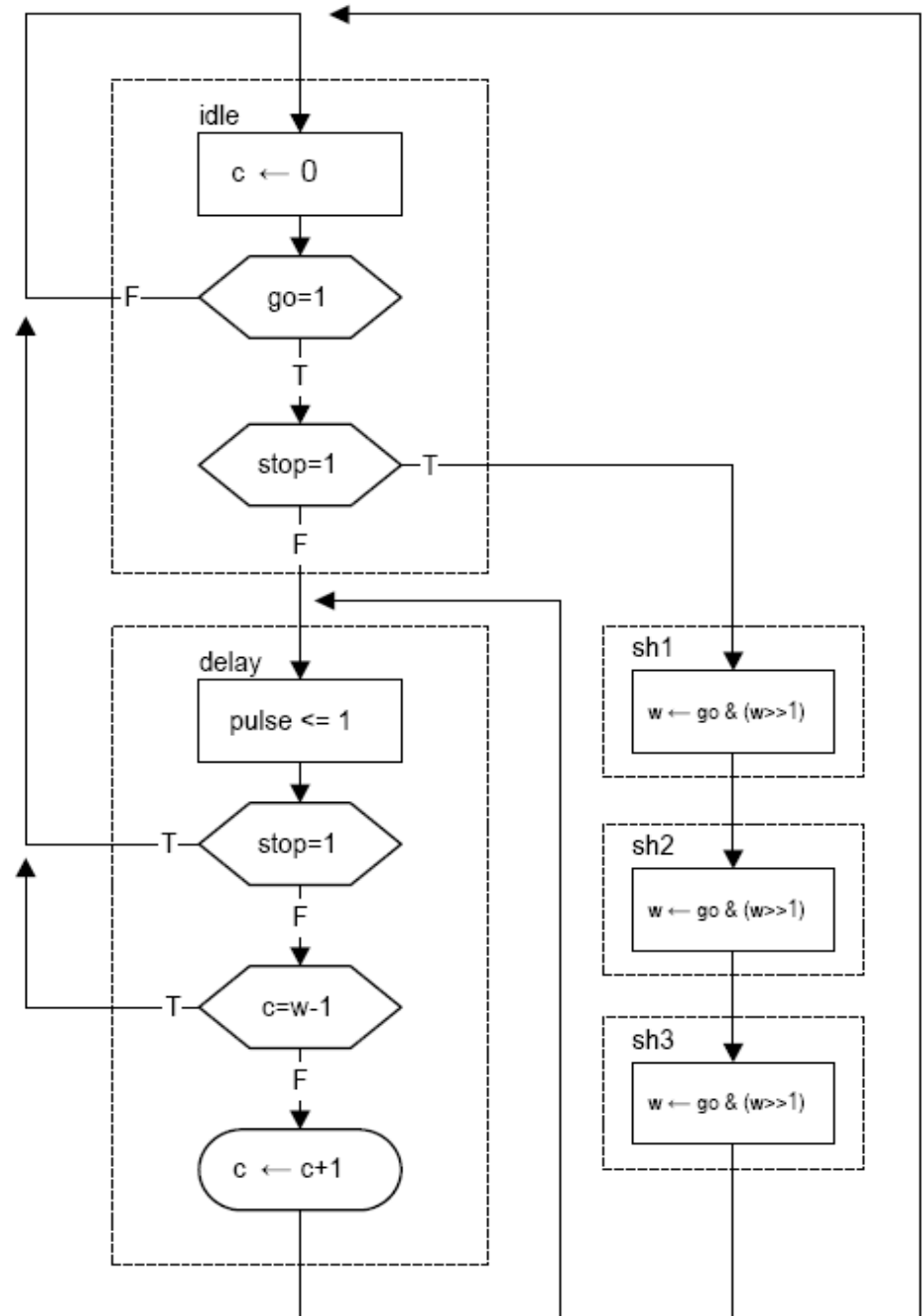
-- next-state logic & data path functional units/routing
process(state_reg,go,stop,c_reg)
begin
    pulse <= '0';
    c_next <= c_reg;
    case state_reg is
        when idle =>
            if go='1' then
                state_next <= delay;
            else
                state_next <= idle;
            end if;
            c_next <= (others=>'0');
        when delay =>
            if stop='1' then
                state_next <=idle;
            else
                if (c_reg=P_WIDTH-1) then
                    state_next <=idle;
                else
                    state_next <=delay;
                    c_next <= c_reg + 1;
                end if;
            end if;
            pulse <= '1';
        end case;
    end process;
end fsmd_arch;

```

- Comparison:
  - FSMD is most flexible and easy to comprehend
- What happens to the following modifications
  - The delay extend from 5 cycles to 100 ccyles
  - The `stop` signal is only effective for the first 2 delay cycles and will be ignored otherwise

- “Programmable” one-shot generator
  - The desired width can be programmed.
  - The circuit enters the programming mode when **both go and stop** are asserted
  - The desired width shifted in via `go` in the next three clock cycles

- Can be easily extended in ASMD chart
- How about FSM and regular sequential circuit?





## 2. GCD circuit

- GCD: Greatest Common Divisor
  - E.g,  $\text{gcd}(1, 10)=1$ ,  $\text{gcd}(12,9)=3$
- GCD without division:

$$\text{gcd}(a, b) = \begin{cases} a & \text{if } a = b \\ \text{gcd}(a - b, b) & \text{if } a > b \\ \text{gcd}(a, b - a) & \text{if } a < b \end{cases}$$

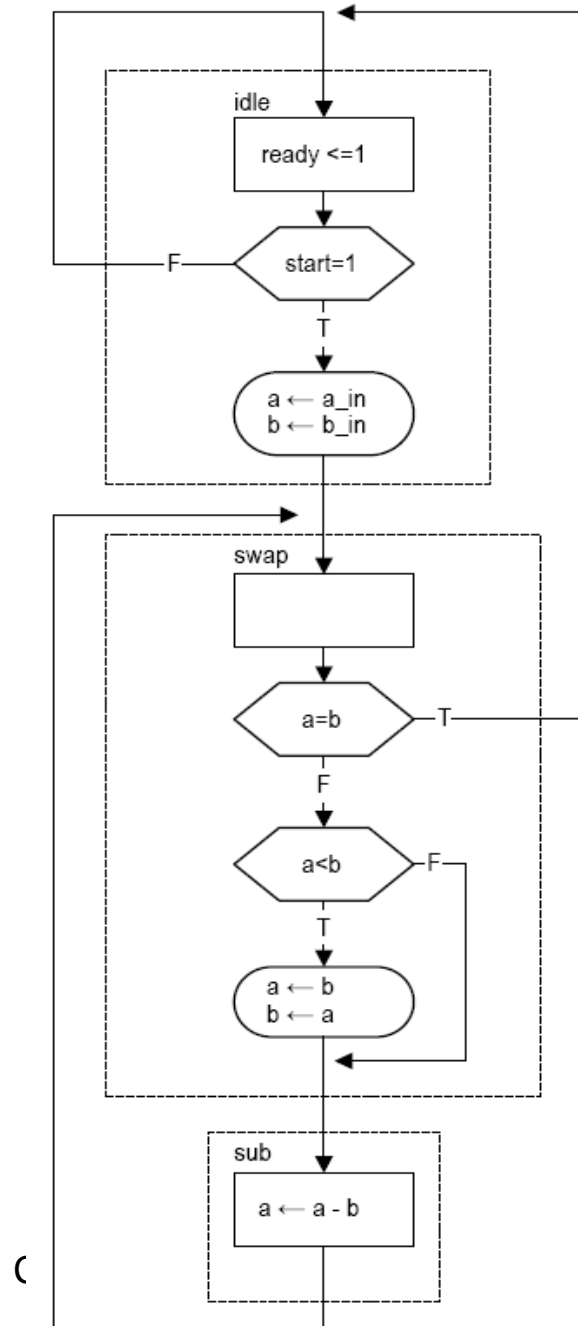
- Pseudo algorithm

```
a = a_in;  
b = b_in;  
while (a /= b) {  
    if (a > b) then  
        a = a - b;  
    else  
        b = b - a;  
}  
r = a;
```

- Modified pseudo algorithm w/o while loop

```
    a = a_in;  
    b = b_in;  
swap: if (a = b)  
    goto stop;  
else  
    if (a < b) then -- swap a and b  
        a = b;  -- assume the two operations  
        b = a;  -- can be done in parallel  
    end if;  
    a = a - b;  
    goto swap;  
end if;  
stop: r = a;
```

- ASMD chart



- VHDL code

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
entity gcd is
    port(
        clk, reset: in std_logic;
        start: in std_logic;
        a_in, b_in: in std_logic_vector(7 downto 0);
        ready: out std_logic;
        r: out std_logic_vector(7 downto 0)
    );
end gcd ;

architecture slow_arch of gcd is
    type state_type is (idle, swap, sub);
    signal state_reg, state_next: state_type;
    signal a_reg, a_next, b_reg, b_next: unsigned(7 downto 0);
```

```

-- state & data registers
process (clk, reset)
begin
    if reset='1' then
        state_reg <= idle;
        a_reg <= (others=>'0');
        b_reg <= (others=>'0');
    elsif (clk'event and clk='1') then
        state_reg <= state_next;
        a_reg <= a_next;
        b_reg <= b_next;
    end if;
end process;

```

```

process (state_reg, a_reg, b_reg, start, a_in, b_in)
begin
    a_next <= a_reg;
    b_next <= b_reg;
    case state_reg is
        when idle =>
            if start='1' then
                a_next <= unsigned(a_in);
                b_next <= unsigned(b_in);
                state_next <= swap;
            else
                state_next <= idle;
            end if;
        when swap =>
            if (a_reg=b_reg) then
                state_next <= idle;
            else
                if (a_reg < b_reg) then
                    a_next <= b_reg;
                    b_next <= a_reg;
                end if;
                state_next <= sub;
            end if;
        when sub =>
            a_next <= a_reg - b_reg;
            state_next <= swap;
        end case;
    end process;

```

- What is the problem of this code?
- Another observation

$$\text{gcd}(a, b) = \begin{cases} a & \text{if } a = b \\ 2 \text{gcd}(\frac{a}{2}, \frac{b}{2}) & \text{if } a \neq b \text{ and } a, b \text{ even} \\ \text{gcd}(a, \frac{b}{2}) & \text{if } a \neq b \text{ and } a \text{ odd, } b \text{ even} \\ \text{gcd}(\frac{a}{2}, b) & \text{if } a \neq b \text{ and } a \text{ even, } b \text{ odd} \\ \text{gcd}(a - b, b) & \text{if } a > b \text{ and } a, b \text{ odd} \\ \text{gcd}(a, b - a) & \text{if } a < b \text{ and } a, b \text{ odd} \end{cases}$$



