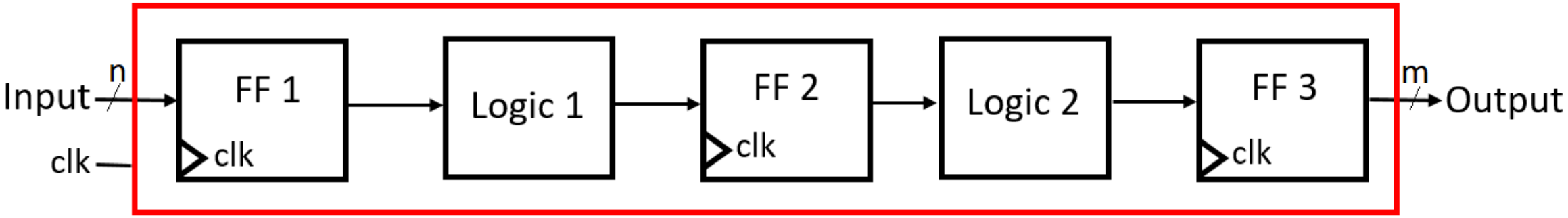
VHDL
System Design Principles

High Level System Description

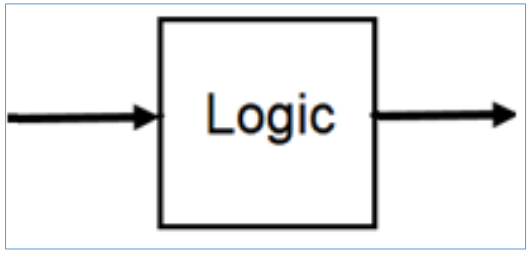
- The most common mistake is to start HDL coding before describing the required digital system in high level of RTL form.
- In RTL system description it becomes clear what are the system's combinational and synchronous subunits.
- Any digital logic system can be disassembled to logic blocks chained to FFs (registers) .

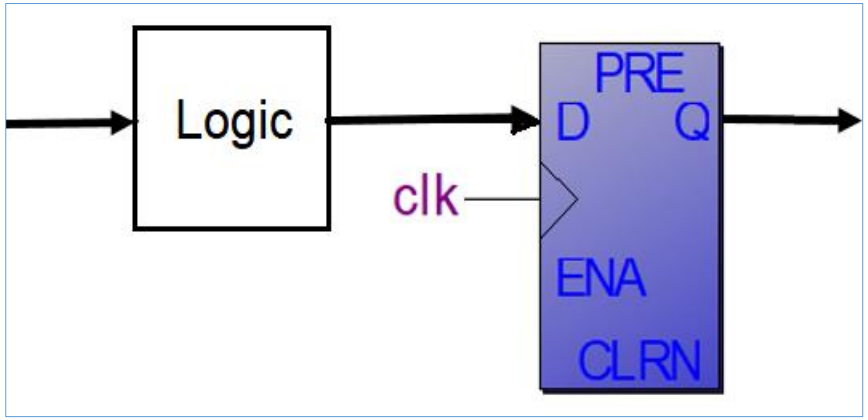


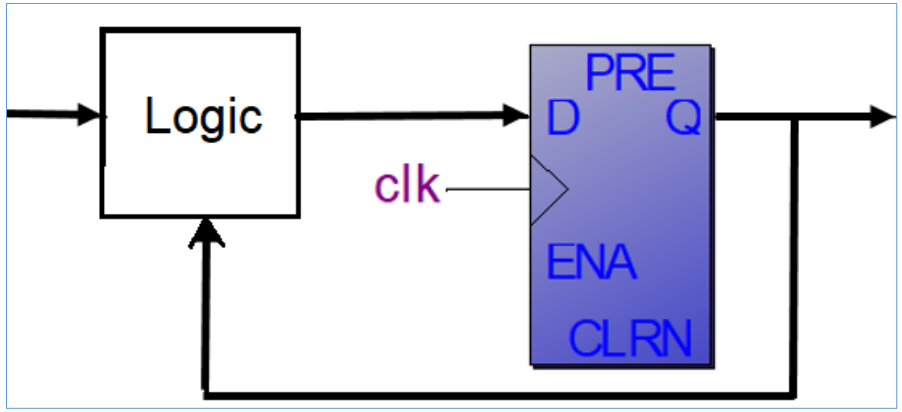
System separation to combinational and synchronous subunits

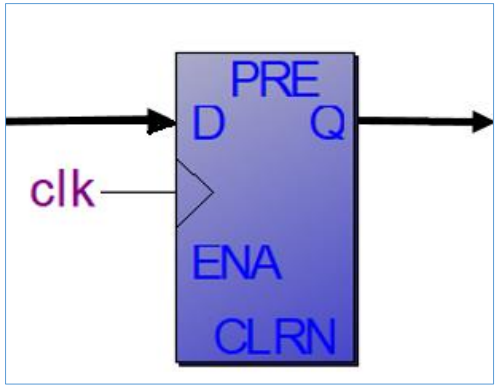
There are four building blocks of combinational and synchronous subunits:

1. Pure logic
2. Pure synchronous (single or multiple chained)
3. Logic through Synchronous
4. Logic through Synchronous feedback

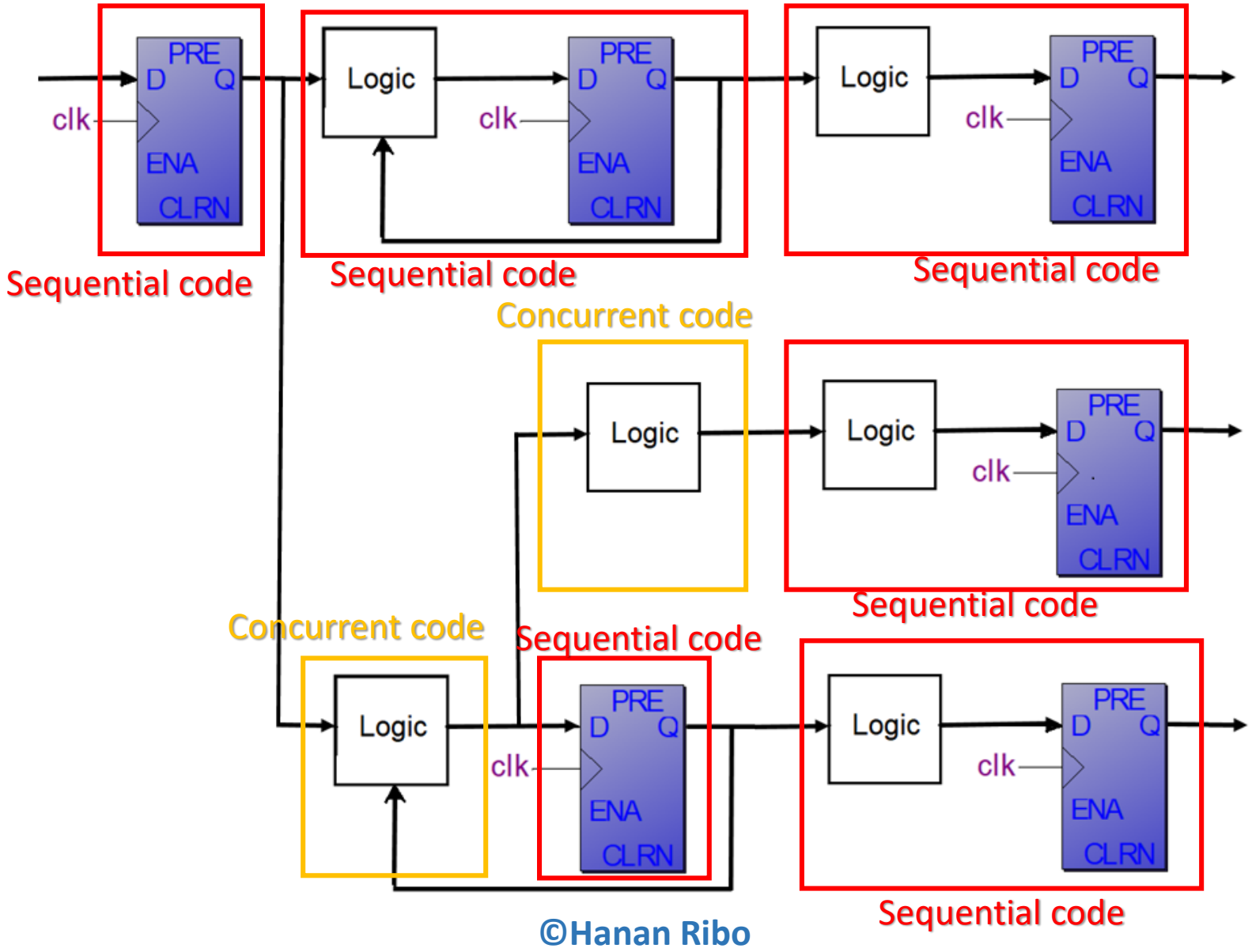








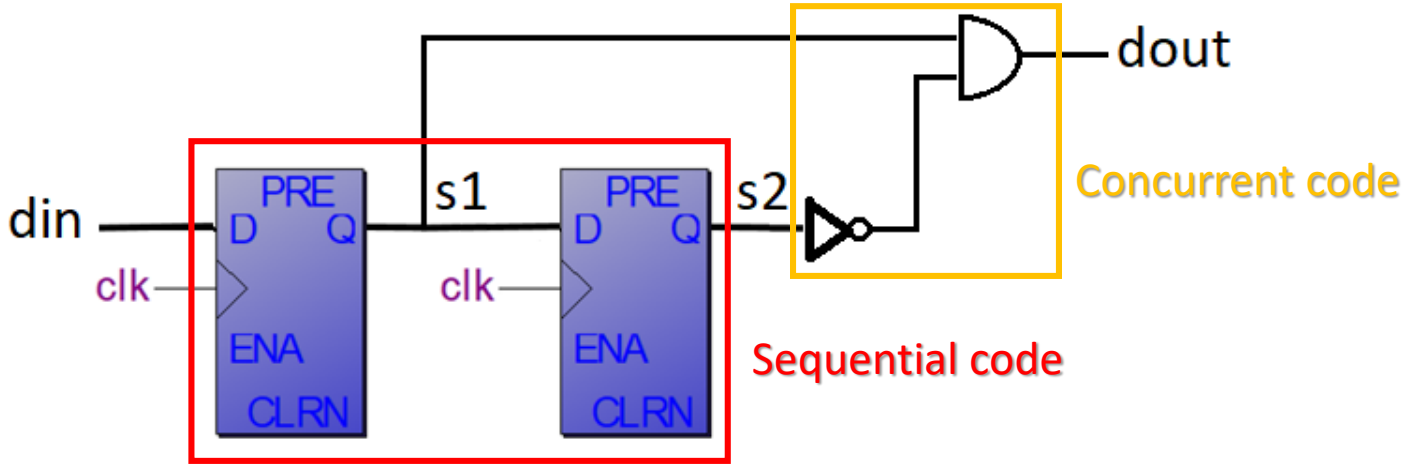
Example of Digital System Disassembly



Mixed Process

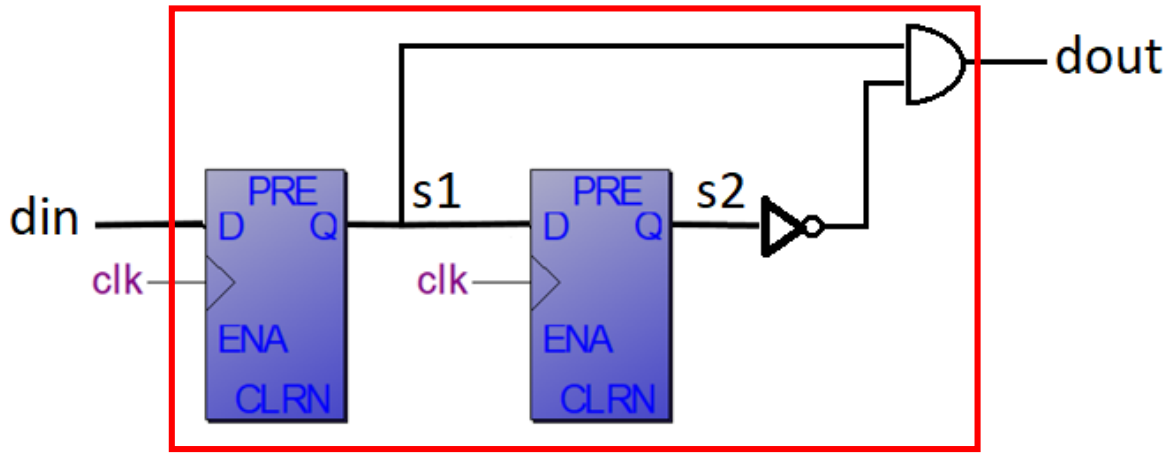
- Mixed PROCESS is a PROCESS which combines combinatorial and synchronous elements inside it (even Latch elements).
- Don’t write Mixed PROCESS, is non synthesizable (ieee-1076.6 standard).
- When you write a PROCESS, ask yourself if it describes combinatorial logic **or** synchronous circuit. If you're not sure, you are writing a Mixed PROCESS.
- Synchronous derivator example – circuit implementation:

Note: without dismantle it into two parts, you write a Mixed PROCESS



Mixed Process – Synchronous derivator

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
-------------------------------------
entity deriv is
    port(
        clk : in std_logic;
        din : in std_logic;
        dout : out std_logic
    );
end deriv;
-------------------------------------
architecture rtl of deriv is
    signal s1,s2 : std_logic;
begin
    process(clk,s1,s2)
    begin
        if (clk'event and clk='1') then
            s1 <= din;
            s2 <= s1;
        end if;
        dout <= s1 and not s2;
    end process;
end rtl;
```



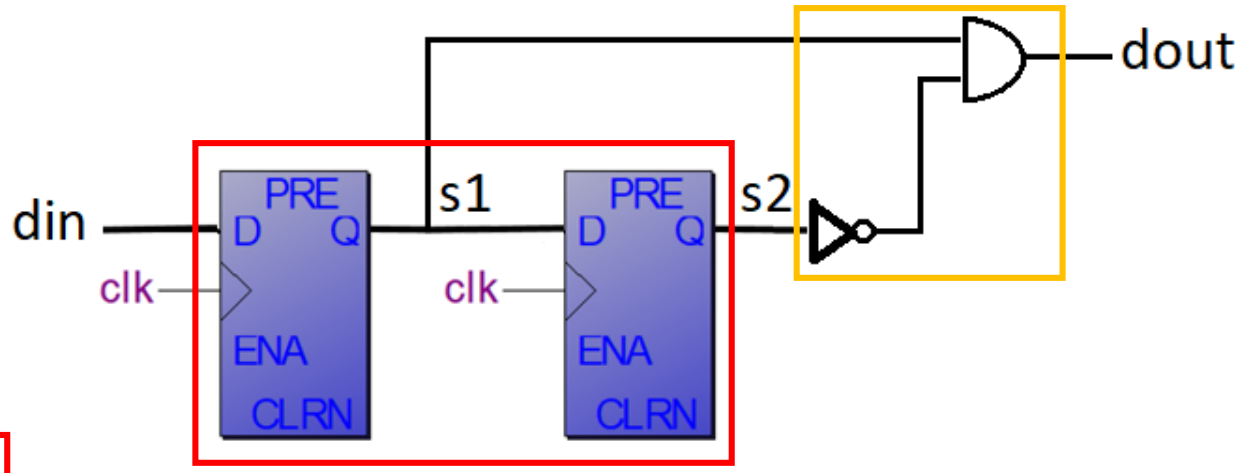
The PROCESS combines combinatorial and synchronous elements inside it

Synchronous derivator – correct approach

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
-------------------------------------
entity deriv is
    port(
        clk : in std_logic;
        din : in std_logic;
        dout : out std_logic
    );
end deriv;
-------------------------------------
architecture rtl of deriv is
    signal s1,s2 : std_logic;
begin
    process(clk)
    begin
        if (clk'event and clk='1') then
            s1 <= din;
            s2 <= s1;
        end if;
    end process;
    dout <= s1 and not s2;
end rtl;
```

Sequential code

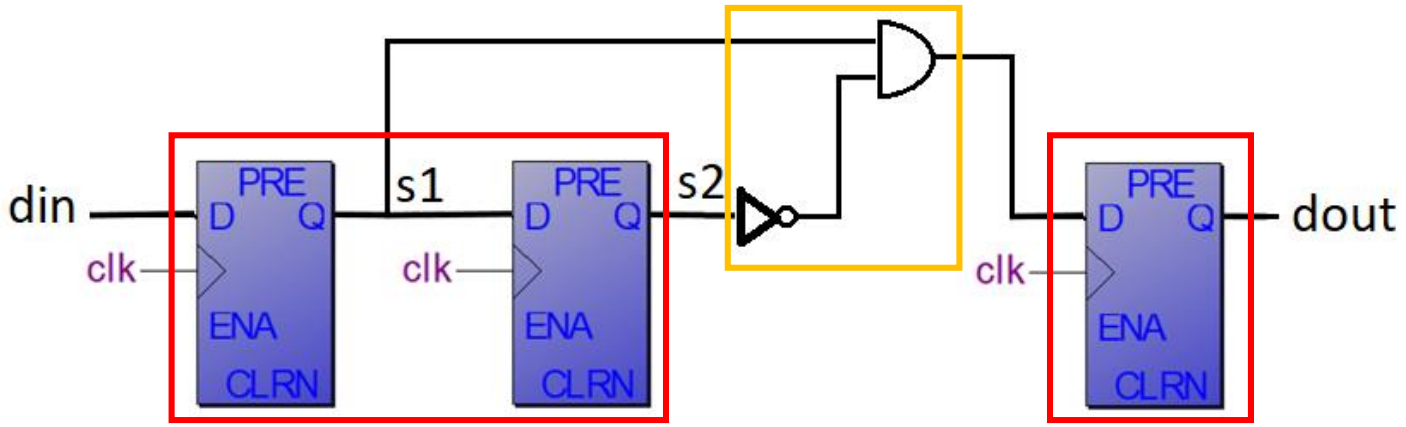
Concurrent code



Question:

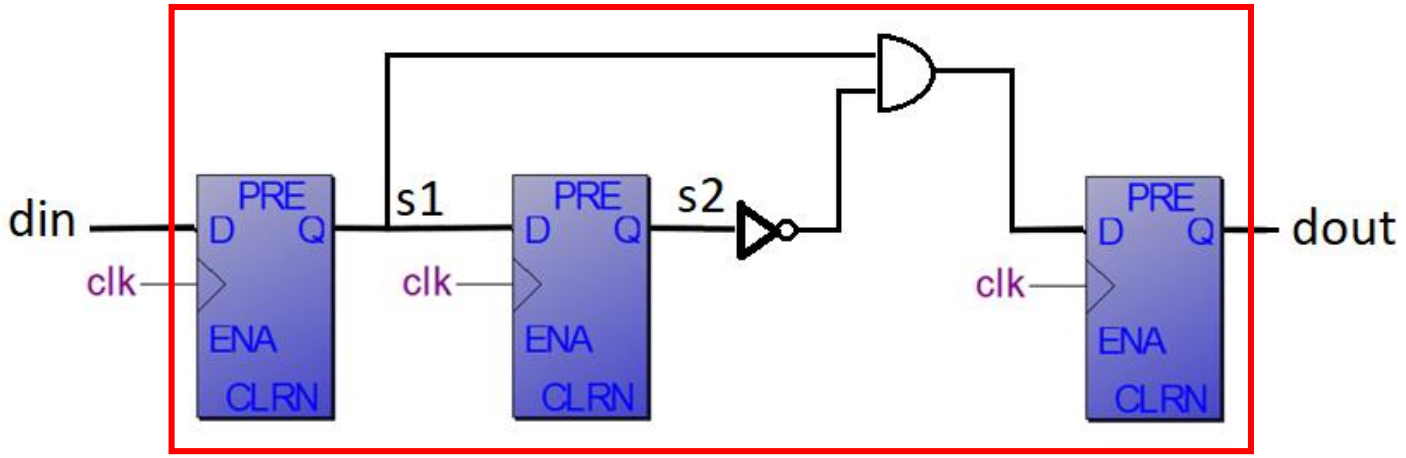
How to implement the given circuit ?

Option No.1



Correct but contains redundancy

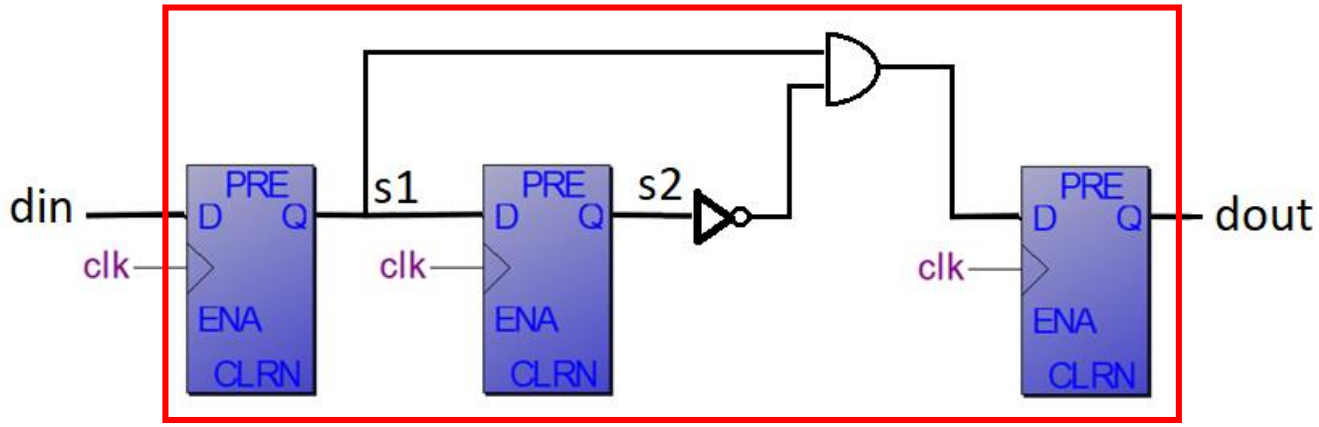
Option No.2



Correct: Pure synchronous (multiple chained)

Pure synchronous (multiple chained) PROCESS

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
---------------------------------------
entity deriv is
    port(
        clk : in std_logic;
        din : in std_logic;
        dout : out std_logic
    );
end deriv;
---------------------------------------
architecture rtl of deriv is
    signal s1,s2 : std_logic;
begin
    process(clk)
    begin
        if (clk'event and clk='1') then
            s1 <= din;
            s2 <= s1;
            dout <= s1 and not s2;
        end if;
    end process;
end rtl;
```



Sequential code