

Programmable Electronics using FPGAs

This week

- Recap – modelling and execution
- Behavioural model with procedural blocks
- Behavioural modelling using processes
- Using processes: combinatorial, sequential, and mixed logic design
- Lab work: flip-flops and registers

Recap: modelling and execution

- Two different types of execution:
 - Dataflow modelling (concurrent execution)
 - Behavioural modelling (sequential execution)
- In the last lecture, we discussed dataflow modelling and architectures, and their usage in combinatorial logic design.

architecture dataflow of gates is

signal s1, s2: std_logic;

begin

s1 <= a AND b;

s2 <= b AND cin;

cout <= s1 OR s2;

end dataflow;

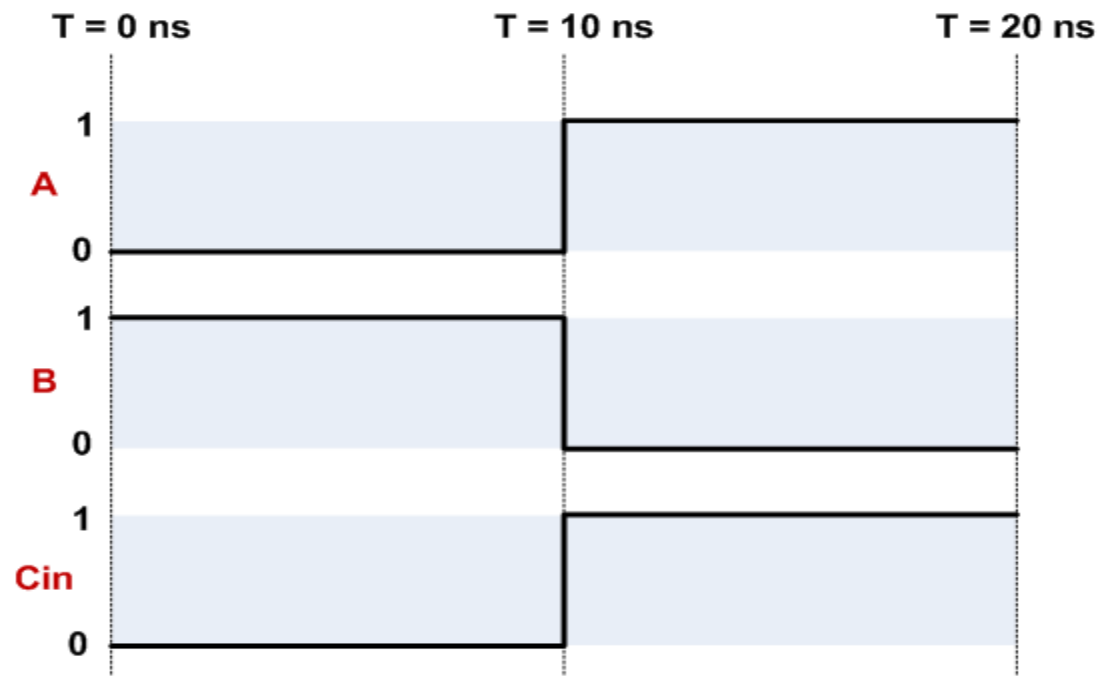
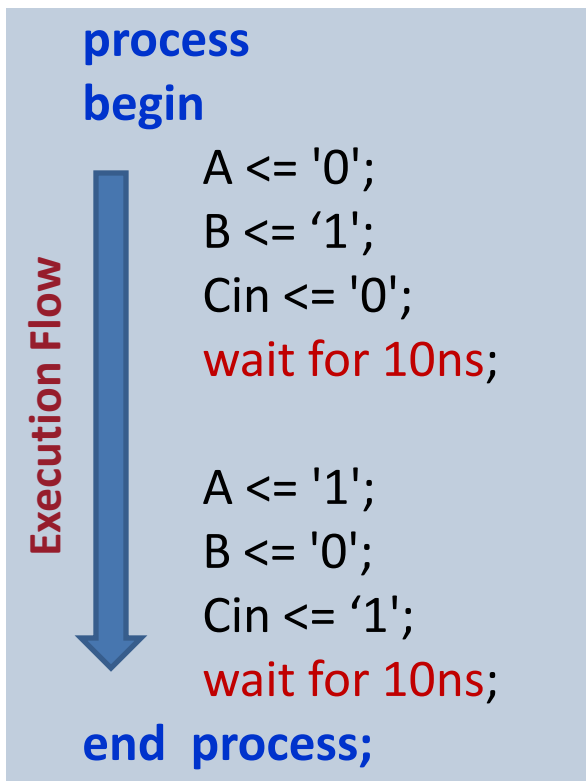
*Concurrent
statements*



Behavioural models: procedural blocks

- This week, we will look in more detail at behavioural modelling.
- We will consider the structure and purpose of blocks of code (known as procedural blocks), and the impact on sequential or concurrent execution (execution flow).

Example: Test vectors generation for a full adder in test bench

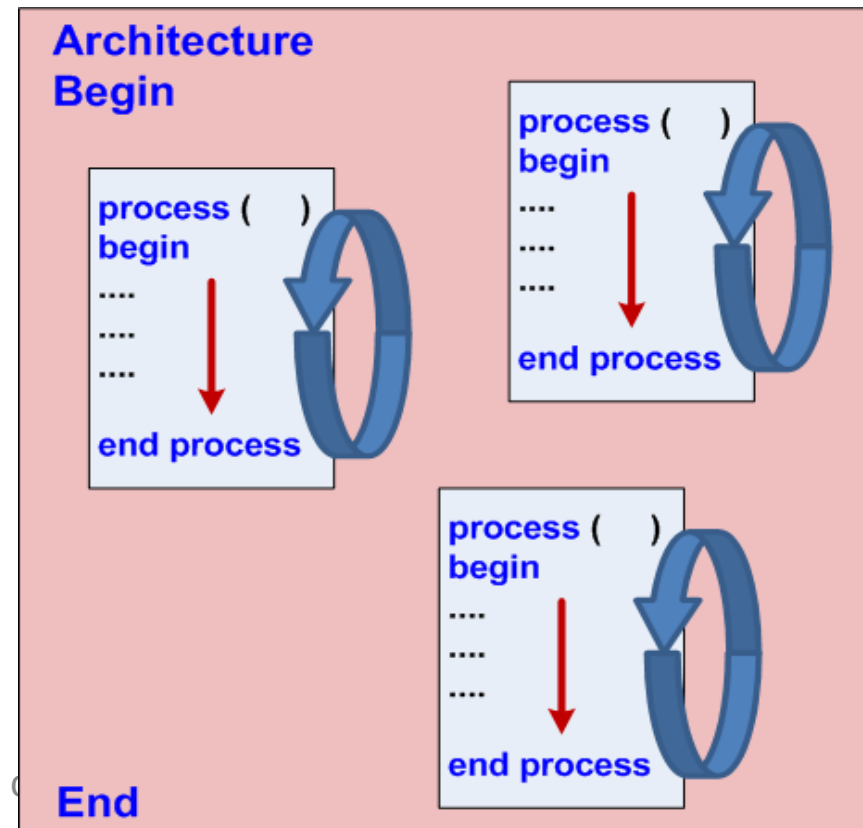


Behavioural modelling using processes

- The keyword process is used to define a single procedural block within a given architecture.
- An architecture can contain any number of processes.
- All processes in an architecture block execute concurrently, and independently.
- In behavioural models, an architecture only contains process blocks – it contains no logic outside of a process block.

Behavioural modelling using processes

- A process is the sequential composition of a number of (related) combinational logic statements.
- Syntactically contained between PROCESS and END PROCESS.
- Processes are normally named.
- Multiple processes execute concurrently.
- They run forever!



Processes and sensitivity lists

- Processes have sensitivity lists.
- Sensitivity lists are lists of input signals that the process is expected to act upon when a signal changes.
- A signal in the sensitivity list changing triggers execution.

Question: How quickly will a process observe a change in a signal in its sensitivity list?

Processes and sensitivity lists

- Processes have sensitivity lists.
- Sensitivity lists are lists of input signals that the process is expected to act upon when the signal changes.
- A signal in the sensitivity list changing triggers execution.

Question: How quickly will a process observe a change in a signal in its sensitivity list?

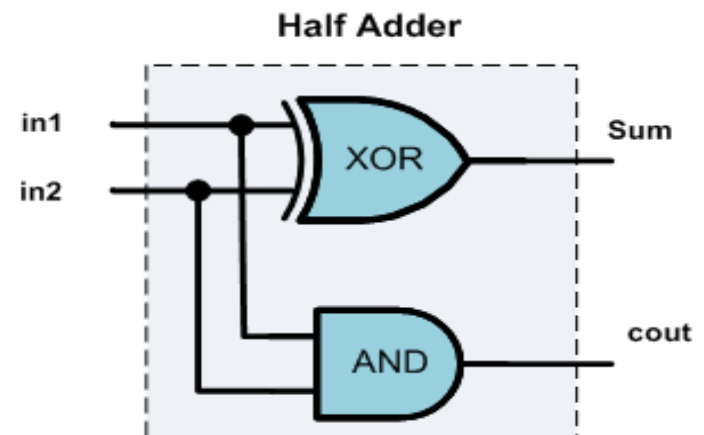
Answer: Depends on sample rates and propagation delays!

Combinatorial logic processes

- A process implements combinatorial logic when there is no edge triggered logic.
 - Typically this means clock edges, although other signal edges may be relevant.
 - Combinational statements are executed sequentially.
- Signals have no intermediate values.

Example: Half-adder

```
PROCESS ( in1, in2 )  
BEGIN  
    Sum <= in1 XOR in2;  
    Cout <= in1 AND in2;  
END PROCESS;
```

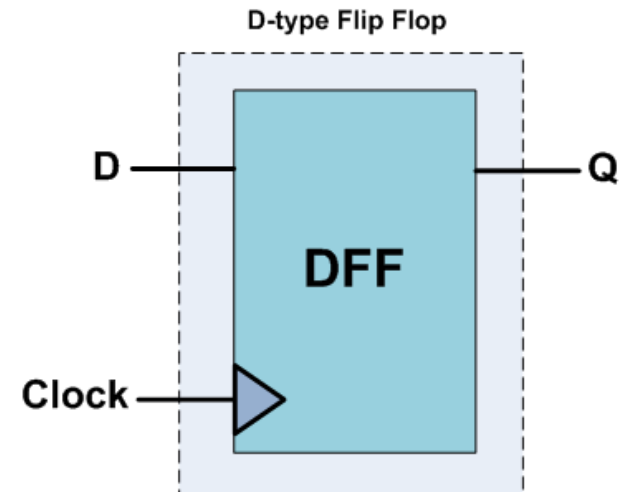


Sequential logic processes

- A process implements sequential logic when there is edge triggered logic.
- Statements are executed on the edge-change given by an IF statement.
 - `if rising_edge(Clock) then` [VHDL 93]
 - `if (Clock'event and Clock = '1') then` [VHDL 87]
- Edge testing is not the same as value testing!

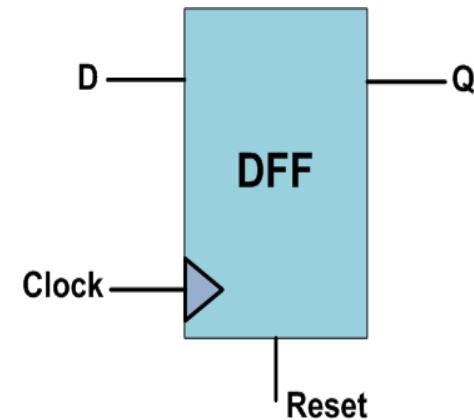
Example: D-type Flip Flop

```
PROCESS ( Clock )  
BEGIN  
    IF rising_edge(Clock) THEN  
        Q <= D;  
    END IF;  
END PROCESS;
```



Sequential logic with reset

- Reset signals are used to put logic into some pre-defined initial state.
- Reset signals can be either active high, or active low.

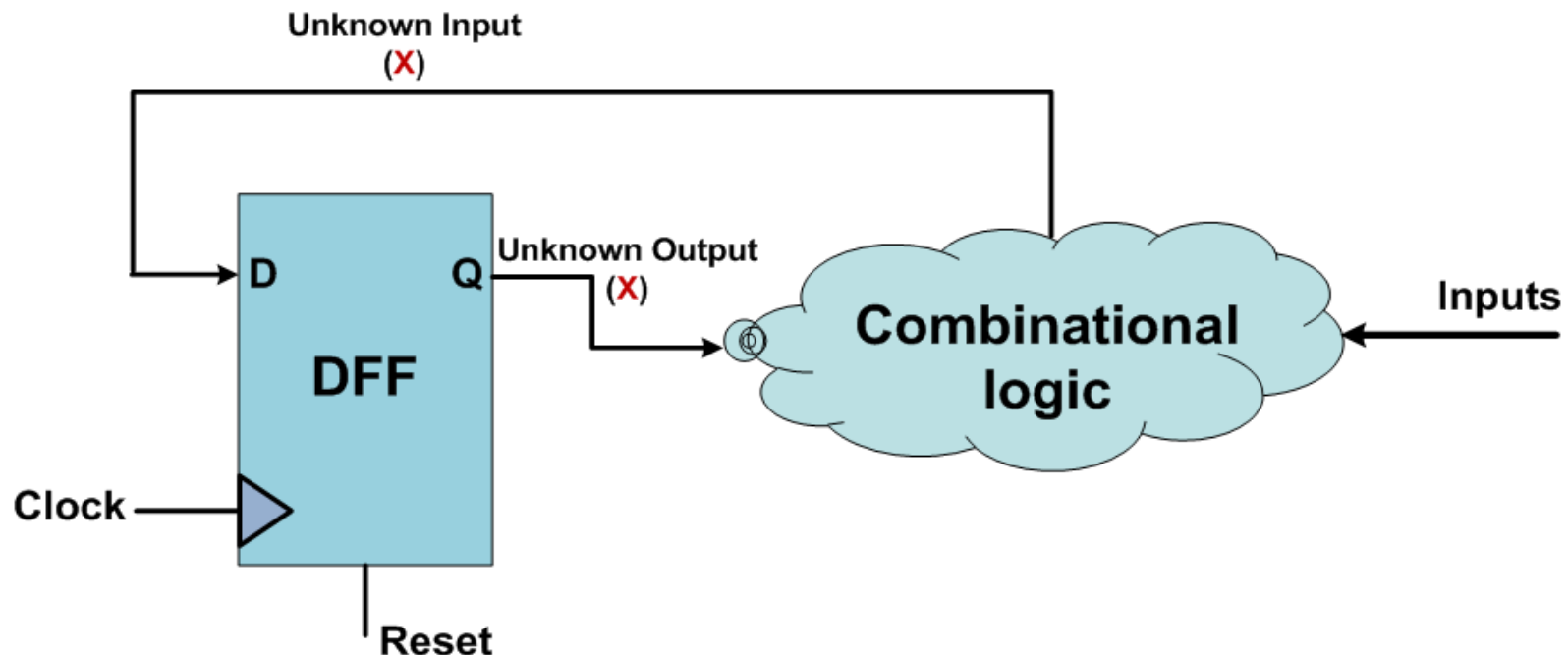


- Synchronous reset signals are clock dependent – in this case the system can reset only if the clock is alive.
- Asynchronous reset signals are not clock dependent – the system can reset anytime irrespective of clock signal.

When is reset necessary?

- Reset is necessary in any circuit that has feedback.
- In case of feedback registers, if we don't reset them, we always get an unknown X at the outputs.

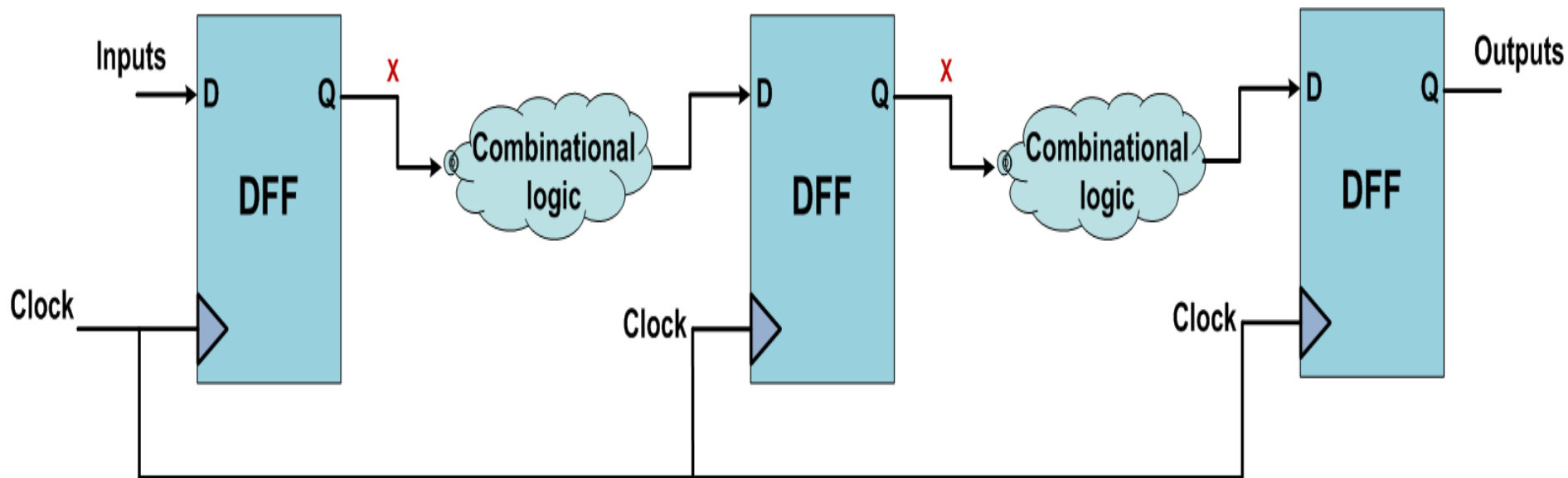
Example: Feedback register logic with reset.



When is reset not necessary?

- With feed forward registers, reset may not be necessary because the unknown X will eventually be flushed.
- But it's good practice to use it!


Example: A chain of feed forward registers without reset.



Sequential process with synchronous reset

```
PROCESS ( Clock, Reset )  
BEGIN  
    IF rising_edge(Clock) THEN  
        IF Reset = '0' THEN  
            Q <= '0';  
        ELSE  
            Q <= D;  
        END IF;  
    END IF;  
END PROCESS;
```


Active low reset



Sequential process with asynchronous reset

```
PROCESS ( Clock, Reset )  
BEGIN  
    IF Reset = '1' THEN  
        Q <= '0';  
    ELSIF rising_edge(Clock) THEN  
        Q <= D;  
    END IF;  
END PROCESS;
```

Active high reset

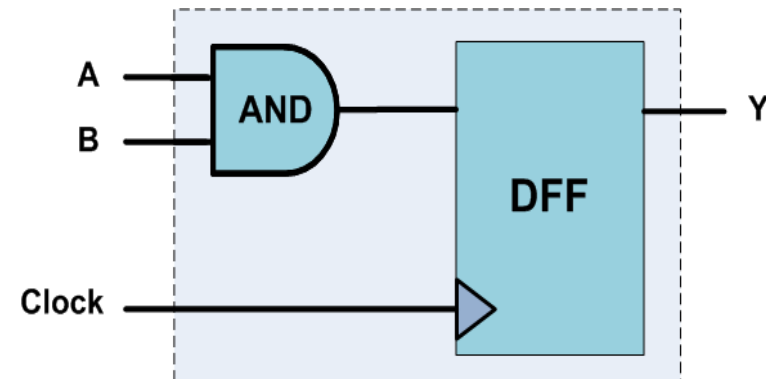


Mixed logic processes

- In behavioural modelling a process implements sequential logic if there is edge triggered logic involved.
- Within the edge triggered logic, assignment statements are used for the combinatorial part of the mixed logic design.

Example: Registered AND gate logic design




```
PROCESS ( Clock )  
BEGIN  
    IF rising_edge(Clock) THEN  
        Y <= A AND B;  
    END IF;  
END PROCESS;
```

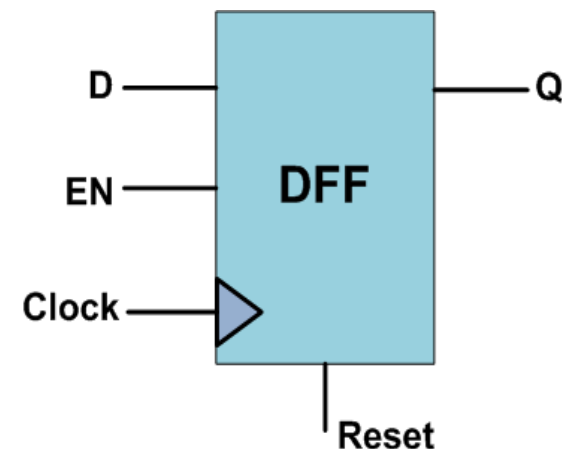


Lab Task 1 : Design an enabled flip-flop (DFF) with asynchronous reset logic

- Input data (D) is stored in DFF (Q) if the input pin (EN) goes high.
- Use an asynchronous reset with active low logic.

Truth table:

Clock	Reset	EN	D	Q	
-	1	-	-	0	Output will be zero
	0	1	1	1	Q=D
	0	1	0	0	Q=D
	0	0	-	Q	No change



Lab Task 2:

Build a 4-bit register using a DFF

- Use your DFF design as a component.
- Structural approach!

