POLITECNICO DI TORINO

02LQD Specification and Simulation of Digital Systems A.Y. 2024/25

Lab session #7

Activity #1: Edge-detection circuit

A slowly varying input signal, *strobe*, is the input to a synchronous system. The synchronous system is an **edge detection circuit**: it detects the rising edge of the *strobe* signal, generating a "short" pulse when the *strobe* signal changes from '0' to '1'. The width of the output pulse is short because it is about the same or less than a clock period of the FSM.

Design 2 versions of the FSM resorting to the VHDL behavioural description style with 2 processes and a:

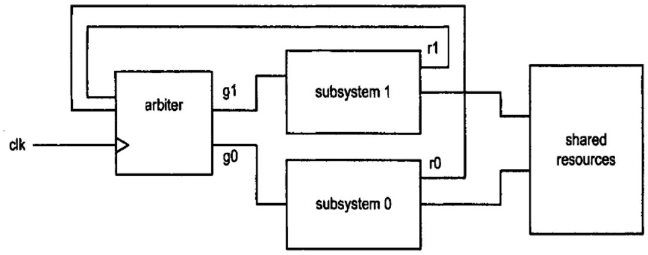
- Mealy output
- Moore output.

Reset is asynchronous. Compare the timing in the 2 cases, in particular when *strobe* stays at '1' for a short time. Create a suitable testbench to test the FSM on relevant values.

Activity #2: 2-request arbiter

Two subsystems may access the same resource. An **arbiter** is a circuit that resolves any conflict and coordinates the access to the shared resource. The arbiter has 2 inputs *req0* and *req1* coming from the 2 subsystems and generates 2 outputs *grant0* and *grant1* that are inputs to the 2 subsystems. When a subsystem needs the resource, it activates its request signal. The arbiter monitors the use of the resource and the requests: if this is the only request, the arbiter grants access to the subsystem by activating the corresponding grant signal. Once its grant signal is activated, a subsystem has permission to access the resource. Any request coming from the other subsystem is ignored. After the task has been completed, the subsystem releases the resource and deactivates the request signal. If both subsystems issue a simultaneous request, 2 strategies are possible:

1. One of the subsystems has priority, e.g., subsystem 1
2. The arbiter keeps track of which subsystem had the resource last time and gives priority to the other subsystem.



Design 2 versions of the Moore FSM resorting to the VHDL behavioural description style with 2 processes and an asynchronous reset:

- With fixed priority to subsystem 1
- Keeping track of which subsystem had the resource last time and giving priority to the other subsystem.

Create a suitable testbench to test the arbiter on relevant values.