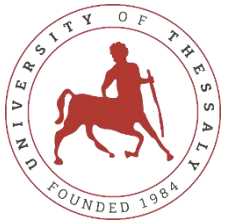


Basic Principles of HDL Verilog Design

ECE333 | Digital Systems Laboratory – Professor: Christos Sotiriou

Lab Instructor(s): Dimitris Tsalapatas, Nikos Zazatis, Nikos Chatzivangelis, Katerina Tsilingiri

Outline

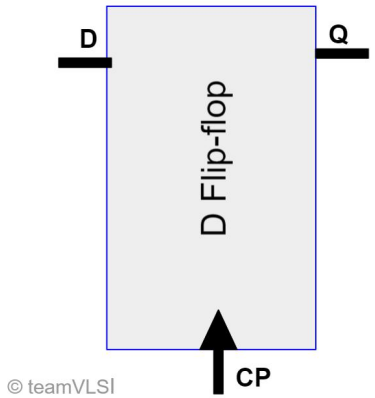
- Golden Rules
  - Dataflow & FSMs Design
  - Procedural Block Assignments
  - Do **NOT** Infer Latches
  - Reset & Clock Logic
- Hierarchical Module Design & Coding Style

Golden Rule #1: Dataflow & FSMs Design

- Design **Dataflow** & **FSM**s *prior the implementation process*
  - Dataflow:
    - **must be strictly equivalent *(1-1)*** to the instantiations of top-level module
    - in case of multiple levels in hierarchy,
      - you can provide more detailed info
        - either within the same schematic
        - or with a separate dataflow of the internal hierarchy
  - State Machine(s):
    - **Mealy** vs **Moore** ***(upcoming lectures)***

Golden Rule #2: Procedural Block Assignments

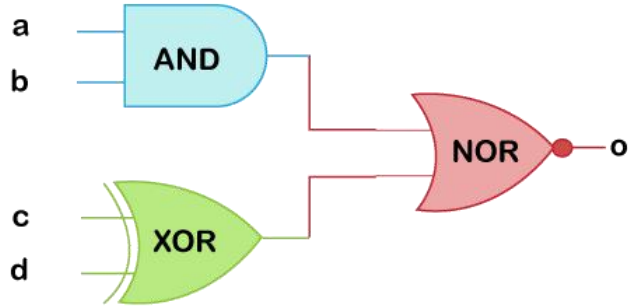
- **Sequential** vs **Combinational** always@block
  - Sequential:
    - *always @(posedge clock or …)*
      - *posedge clock*, implies the **instantiation of flip-flop*(s)***
      - every *other signal in sensitivity list*, implies an **asynchronous control** signal for the flip-flop*(s)*
      - any signal *not in the sensitivity list* and *used for multiplexing* within the block, implies a
        - **synchronous control** signal for the flip-flop*(s)*
          - *Rule of thumb:* ***Non-Blocking Assignments (<=)***



Golden Rule #2: Procedural Block Assignments

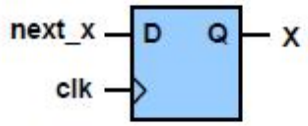
- **Sequential** vs **Combinational** always@block
  - Combinational:
    - *always @(signal_1 or … or signal_x)*
      - sensitivity list must contain **all associated input** signals
        - *Rule of thumb:* ***Blocking Assignments (=)***

ALWAYS SEPARATE combinational and sequential logic blocks
Do NOT MIX blocking and non-blocking assignments *in the same alwaysblock*
Do NOT ASSIGN the same variable *from more than one alwaysblock*
Do NOT ASSIGN as output and **USE** as input a variable *in the same always block*

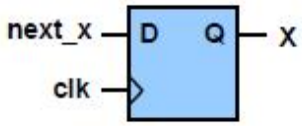


Golden Rule #2: Blocking VS Non-Blocking

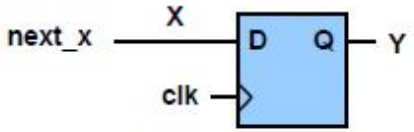
```
always @( posedge clk )
begin
  x = next_x;
end
```



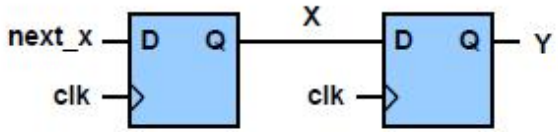

```
always @( posedge clk )
begin
  x <= next_x;
end
```




```
always @( posedge clk )
begin
  x = next_x;
  y = x;
end
```




```
always @( posedge clk )
begin
  x <= next_x;
  y <= x;
end
```



Golder Rule #3: Do NOT Infer Latches

- **Complete signal state** *for all* cases in Combinational alwaysblock
  - Latches are inferred due to
    - **NOT assigning Left Hand Side (LHS) for ALL conditions**

      or
    - **MISSING signal in the sensitivity list**

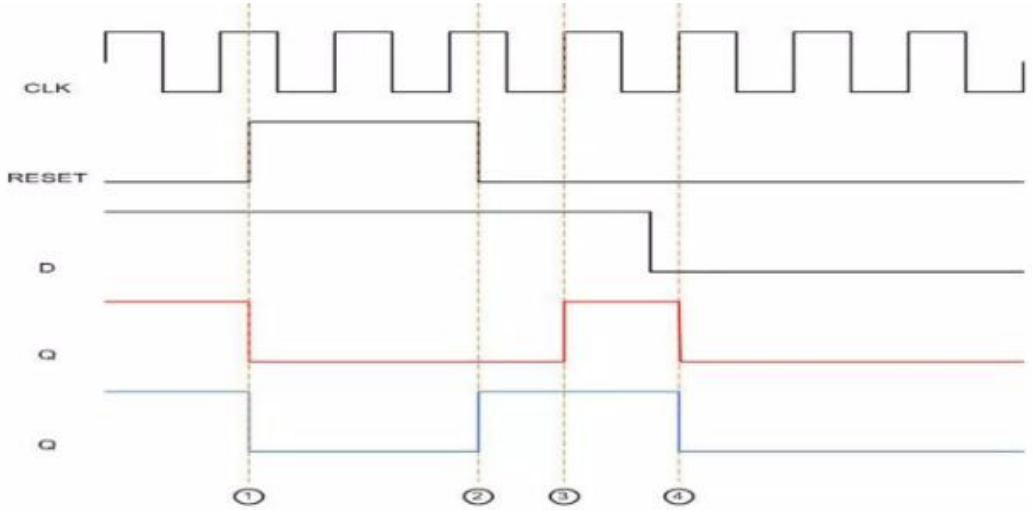
Every if has a **COMPLEMENTARY** elsestatement

Every case is **full** and is defined for the **complete state** *of the examined input signals*

Every if/else/case must **ASSIGN ALL** *LHS signals*

Golden Rule #4: Reset & Clock Logic

- **NEVER put logic** on the ***reset*** or ***clock***
  - **NEVER MIX** reset types
    - **asynchronous** vs **synchronous** reset
  - **NEVER** create clock domain crossings ***(upcoming lectures)***
  - Advanced designs have more than one clocks *which they usually run on different frequencies*



Hierarchical Module Design and Coding Style

- Top-module must contain only module instantiations
  - **NO RTL in top-level**
    - direct wire assignments are allowed
      - *e.g.:* assign $signal_X$ = $signal_Y$;
- Every module should be in a separate file
  - module header comment segment with a basic functional description
  - I/O ports description
  - utilise parameters or definitions
    - ***localparam***vs ***parameter*** vs ***`define***
  - for each instantiation/implementation add header comment
    - describe explicit corner cases
- Follow a uniform coding style across all files
  - comments
  - indentations
  - begin-end usage and format *etc.*

Any Questions?