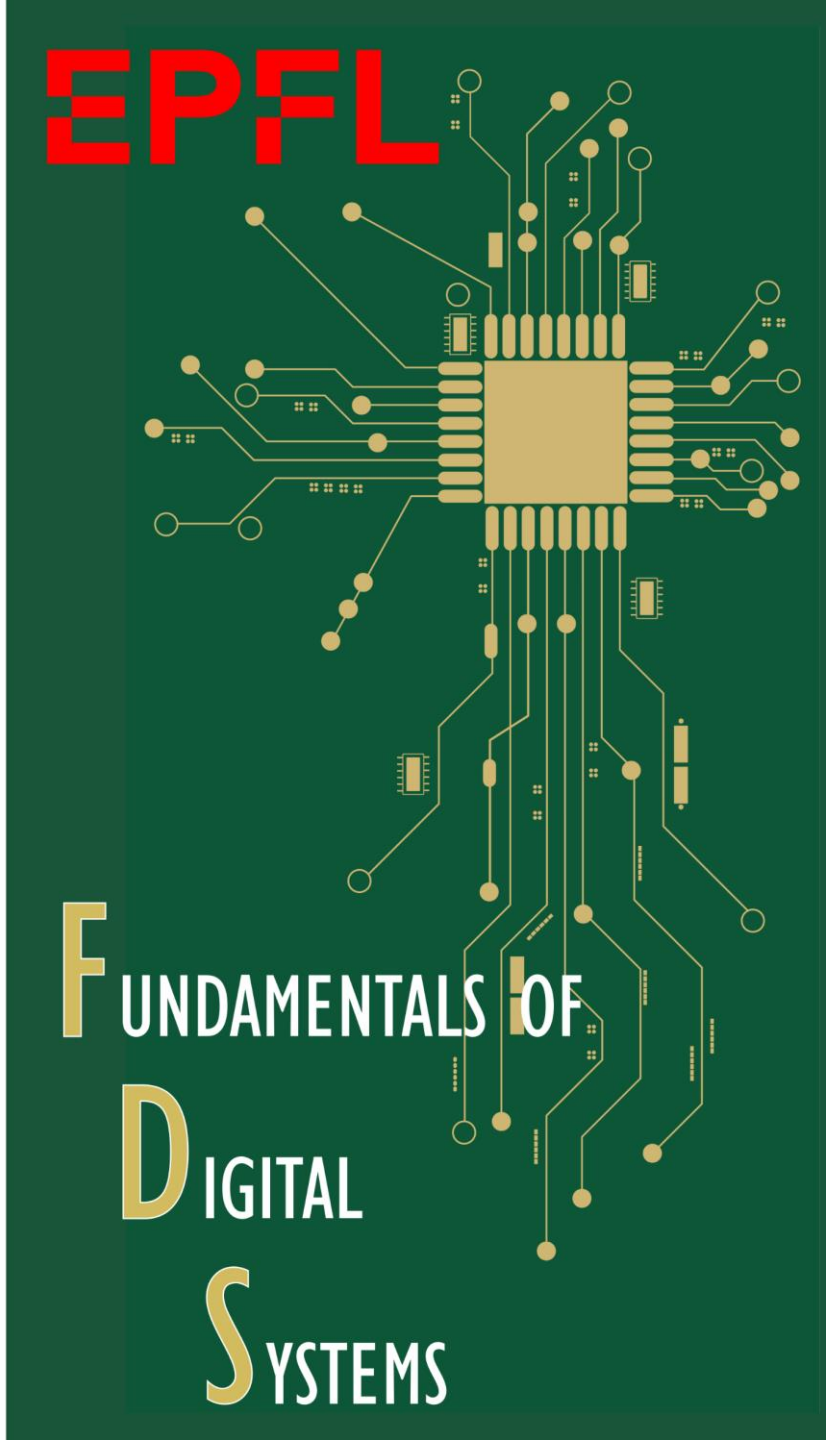


Verilog

Coding Style Guide

CS-173 Fundamentals of Digital Systems

Mirjana Stojilović

2024

Readability Matters

- Have a well-formatted code
  - **Indent** properly and consistently
  - Insert empty lines to give **structure** and help **readability**
  - Add **comments** if the functionality is not obvious
  - Use one-line comments, starting with **//**
  - Write no more than one Verilog statement per line
- Choose names carefully for your modules, ports, signals, etc., to facilitate code understanding
- Choose a naming convention and apply it consistently

Style Guidelines

- Many style guides exist
- Example style guide: link

Verilog HDL Coding

Semiconductor Reuse Standard

IPMXDSRSHDL0001
SRS V3.2

freescale semiconductor

Instantiating Modules as Subcircuits

- Connect ports by **name** in module instantiations
  - Do not connect ports by position
  - Connecting by name improves readability and adaptability
    - It causes fewer errors if, for instance, you decide to update the list of ports of your module in the future
- Logic expressions are not allowed in port connections
  - Concatenations and constants are allowed

Unused Module Inputs and Outputs

- Drive all unused module inputs by some other signals or by a fixed logic zero or one

- All unused module instance outputs should be connected to a “dummy” wire declaration that, for readability, indicates no connect (or open) in its name

Declare Internal Nets

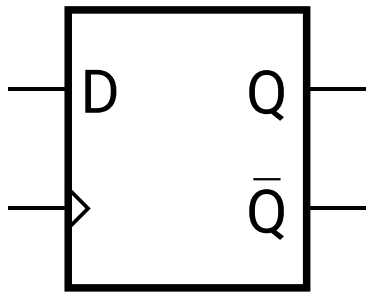
- Internal nets must be declared explicitly, not implicitly
- Port nets do not need to be redeclared in wire declarations in addition to the input/output/inout declarations
- Group all internal net declarations (wire, reg) together in one section following the input/output/inout declarations at the top of the module

Blocking and Nonblocking Assignments

In Verilog

- Use blocking (=) assignments in **combinational** `always` blocks

- Use nonblocking (<=) assignments in **edge-sensitive** sequential code blocks. Blocking assignments (=) are not allowed
- Recall that the logic symbol for an FF has a little wedge on the clock input; have that remind you to use **<=** operator



- Latches must be written with nonblocking assignments

`always@(*)` Blocks

In Verilog

- Most often used to describe **combinational logic**
- Use `*` for the sensitivity list of the combinational circuit
  - Because you want your outputs to react to a change of any of the inputs
  - Use **`always@(*)`** block when wanting to infer elements that change value as soon as one or more of the inputs change

`always@(posedge Clock)` Blocks

In Verilog

- Used to describe **sequential logic containing flip-flops**
- For clock signal named `Clock`
  - `always@(posedge Clock)`
    - active clock edge of the FF is the rising clock edge
  - `always@(negedge Clock)`
    - active clock edge of the FF is the falling clock edge

Avoid Latches

- For practical sequential systems, avoid latches
  - Latches are sensitive to glitches (hazards)
  - Latch outputs can oscillate
  - Latches are level (not edge!) sensitive and may change output many times during one clock period
- Use only D flip-flops and combinational logic
  - Write independent `always` blocks
    - Some dedicated to D flip-flops (keep them extremely simple)
    - Some dedicated to combinational logic (as complex as needed)

Specify Combinational Logic Completely

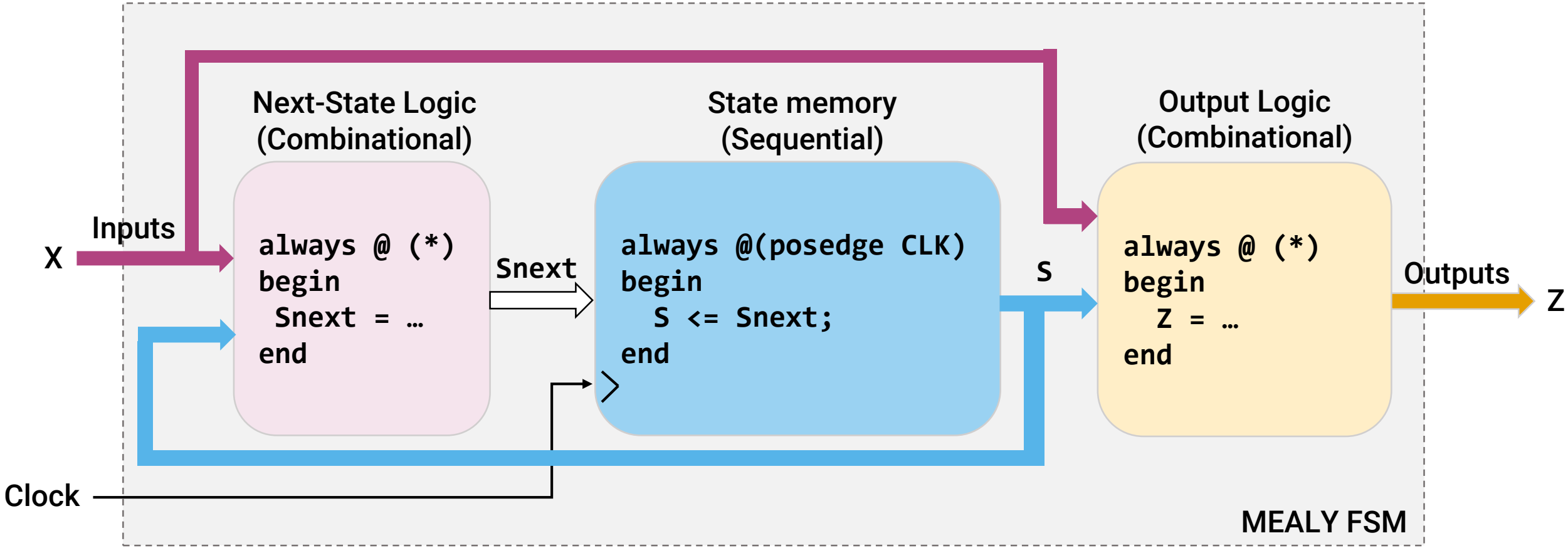
- To avoid latches to sneak into your **combinational circuits**, make sure to assign a value to the nets for all input combinations
- Recommended practice:
  - Start your `always@(*)` block with the initialization of **all** nets to some **default** logic values, so they all take a value (1 or 0) **regardless** of the code that follows; this initialization section ensures your circuit has no latches
  - After the initialization section, proceed to describe the combinational circuit functionality

FSM Modeling

- To describe an FSM in Verilog, use three `always` blocks
  - one **combinational**, for next-state logic
  - one **sequential**, for the state memory (FFs)
  - one **combinational**, for the output logic
- When modeling the state memory, pay attention to
  - active clock edge: rising (`posedge`) or falling (`negedge`)
  - reset: asynchronous or synchronous, active high or low
- Whenever possible, give names to FSM states and declare them as Verilog `parameters`

Mealy FSM Modeling

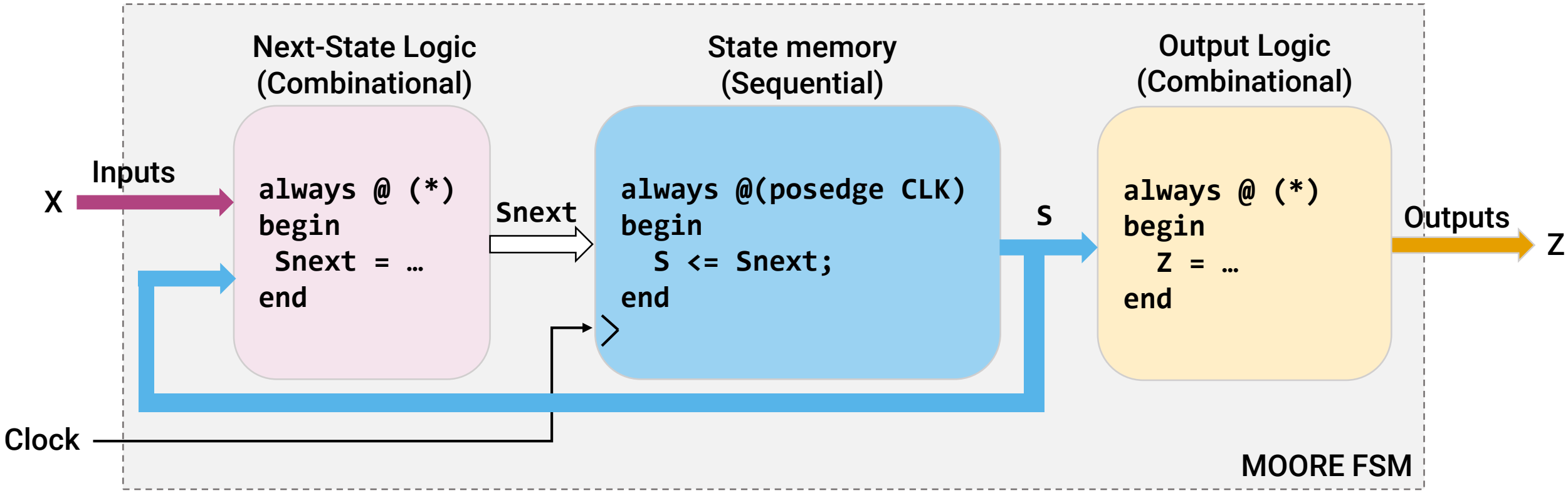
In Verilog



Note: Adjust the sensitivity list of the state memory, as required

Moore FSM Modeling

In Verilog



Note: Adjust the sensitivity list of the state memory, as required