CE653 – Asynchronous Circuit Design

Instructor: C. Sotiriou

http://inf-server.inf.uth.gr/courses/CE653/

Contents

One-Hot/One-Cold Encoding

- What is unique to One-Hot/One-Cold State Codes?

State Number	Sequential Encoding	Gray Encoding	Johnson Encoding	One-hot Encoding
0	0000	0000	00000000	0000000000000001
1	0001	0001	00000001	0000000000000010
2	0010	0011	00000011	0000000000000100
3	0011	0010	00000111	0000000000001000
4	0100	0110	00001111	0000000000010000
5	0101	0111	00011111	0000000000100000
6	0110	0101	00111111	0000000001000000
7	0111	0100	01111111	0000000010000000
8	1000	1100	11111111	0000000100000000
9	1001	1101	11111110	0000001000000000
10	1010	1111	11111100	0000010000000000
11	1011	1110	11111000	0000100000000000
12	1100	1010	11110000	0001000000000000
13	1101	1011	11100000	0010000000000000
14	1110	1001	11000000	0100000000000000
15	1111	1000	10000000	1000000000000000

One-Hot/One-Cold Encoding

- One-Hot encoding is not actually an encoding per se
  - Simple and unique assignment of binary codes to symbolic states
- Very commonly used in industry, why?
  - Fast in terms of CAD tools time → no computation required with respect to binary encoding states
  - Fast in terms of circuit operation → no encoding/decoding logic required in the fan-in/fan-out of state variables
  - One-Hot FSM is compositional → easy to break to pieces
- Disadvantage?
  - Number of binary signals linearly dependent on number of states
  - In many cases this is not a problem as FSMs are usually small

One-Hot/One-Cold Encoding in Asynchronous Design

- In asynchronous design, encoded states may present **races** in state transitions
  - Presented earlier in Hazards/Race analysis Methods
- One-Hot Encoding transitions are **regular**
  - Two signal transitions per state transition
  - `1` → `0` and `0` → `1` depending on current/next state
- Is One-Hot Encoding race-free?
  - NO! But the race is regular and can thus be rectified by using a regular circuit structure
- One-hot Critical Race (for any n states add `0`s):
  - `10` → `11` → `01` (2 states – unique)
  - `100`→`110`→`010`→`011`→`001` (3 states – not unique)

One-Hot Critical Race

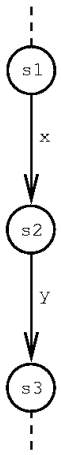
- For any two state transition (for any n states add `0`s):
  - **Correct** transition: `10` → `11` → `01`
  - *Erroneous* transition: `10` → `00` → `XX`
- Avoiding the Race imposes a Race Constraint
  - ***Enter the next state before leaving the current state***
  - Only the first transition is acceptable: `10` → `11` → `01`
  - No interference between next/current state (`11`) due to one-hot encoding!
- How can the Race Constraint be satisfied?
  - As a timing constraint for the state set and reset paths
  - By circuit modification

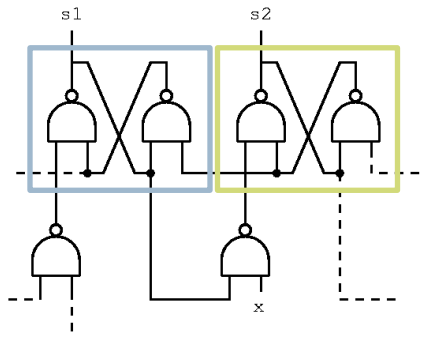
One-Hot FSMs

- Regular Template Circuit Structure
  - State is SR Latch
    - Implemented using standard-cells or as a single latch cell
  - Set and Reset conditions implemented typically in 2-level Logic
  - **Set** condition =
    `Previous State AND Enter Conditions for Current State`
  - **Reset** condition = `Next State (Entered)`
- May be implemented as SI or Fundamental Mode
  - Depends on Environment constraints
  - Depends on Local Timing constraints required for Correct operation

One-Hot FSMs – Linear FSM Example

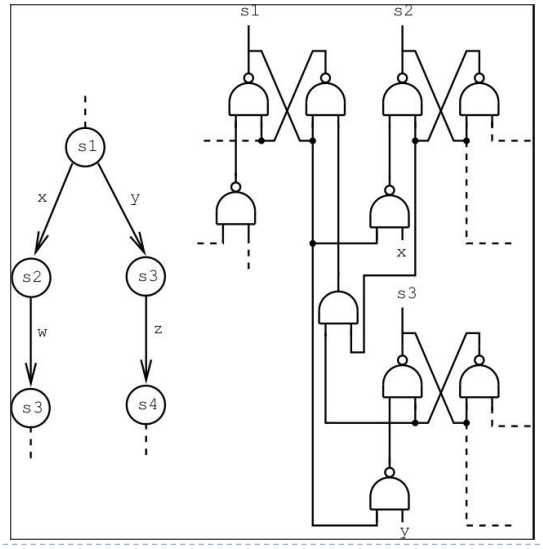
- Each One-hot state may be implemented using an SR latch
- **Set** input
  - State entering conditions
- **Reset** input
  - State exit conditions
- Both Active-low due to NAND SR latch
- Regular Circuit Structure!





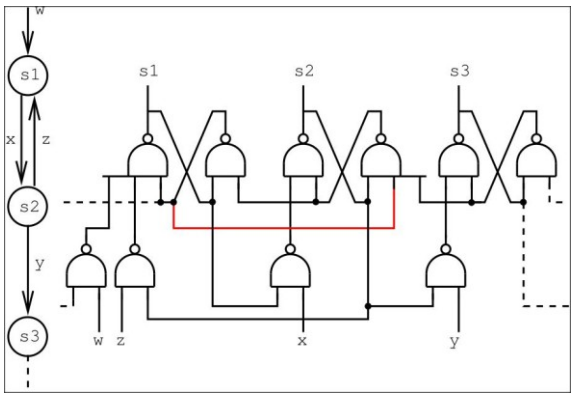
One-Hot FSMs – Choice FSM Example

- Choice can easily be implemented
  - Reset condition for state `s1` is the Set condition of `s2` OR `s3`
  - Transforms to AND due to active-low Reset
- Similarly for return from Choice
  - Reset for predecessors is Set of successor



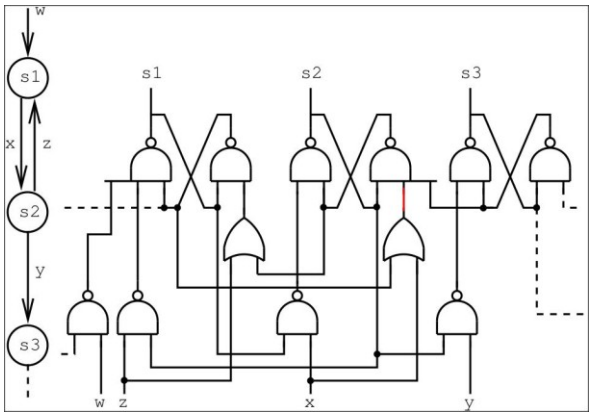
Scale-of-Two Loops

- For correct operation *Set* and *Reset* conditions must be **Mutually-Exclusive**
- If the predecessor and successor states are the same this correctness condition is violated if Set/Reset conditions overlap in time
- What if both `x` and `z` are high in RHS?



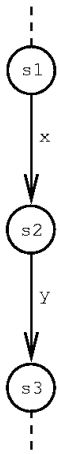
Scale-of-Two Loops - Solutions

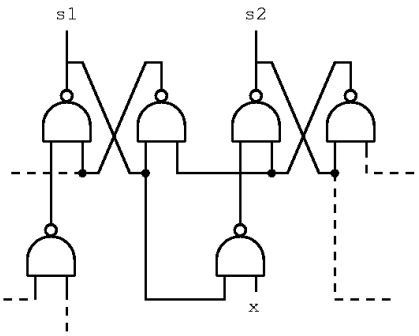
- To ensure Mutually-Exclusive Set and Reset signals we can either:
  - **Add intermediate *dummy* state** (SI operation)
  - **Add transition signals to Reset conditions** (normally ignored)



One-Hot Race

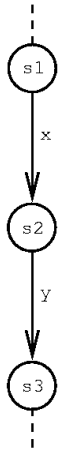
- Is this approach Race-Free?
  - NO!!!
  - Why not?
- It ***does not enforce*** One-Hot Race-Free behaviour:
  - 10 → 11 → 01
- Can it be made Race-Free?

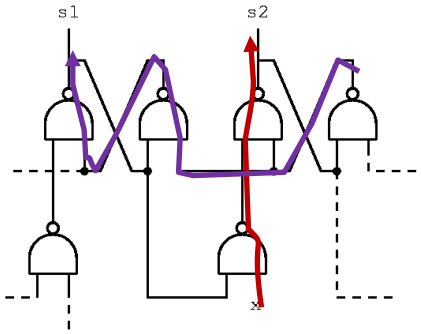




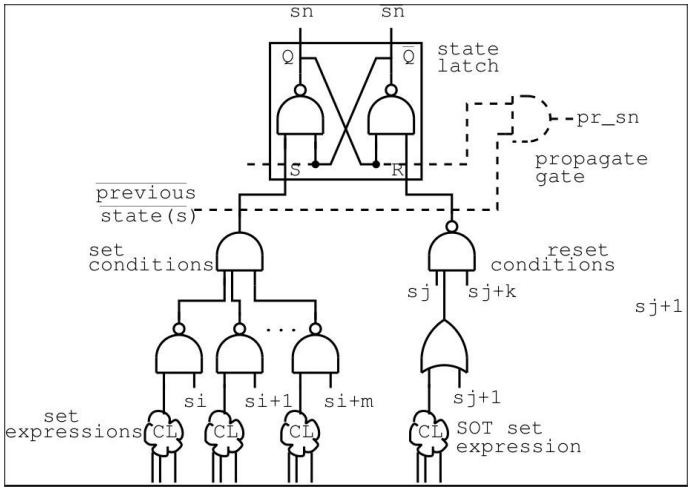
One-Hot Race

- Can it be made Race-Free?
  - Yes
- Two approaches:
  - Ensure Set path is faster than Reset path (for two or *more states*!!!)
- SI Solution:
  - Add a propagate gate (current state and NOT previous state)
  - `PR = CS.PS'`
  - Use `PR_sn` instead of sn





General One-Hot FSM State Structure



- Propagate gate (`pr_sn`) ensures SI operation

What about Hazards? Is this a Hazard-free Approach?

- It is possible for the Set and Reset Logic to contain Logic Hazards
- How can this be resolved?
- Hazards may be analysed using Fundamental or SI Analysis Techniques
- Logic Hazards can be removed by restructuring Set/Reset Logic

Other Observations about One-Hot FSMs

- It is possible to extend One-Hot FSMs to model FSMs with Forks and Joins!
  - Allow multiple One-hot states to be high between fork and join states
- How?
  - Modify reset conditions

CMOS One-Hot FSMs

- *Possible to implement One-Hot FSMs at Transistor-Level*
- Set and Reset Conditions become **Pull-up** and **Pull-down** networks of State-storing gates