EE-421: Digital System Design

Finite State Machines (FSM): Impact of State Encoding on FSM Optimization & EDA Tool Behavior

Dr. Rehan Ahmed [rehan.ahmed@seecs.edu.pk]

NUST

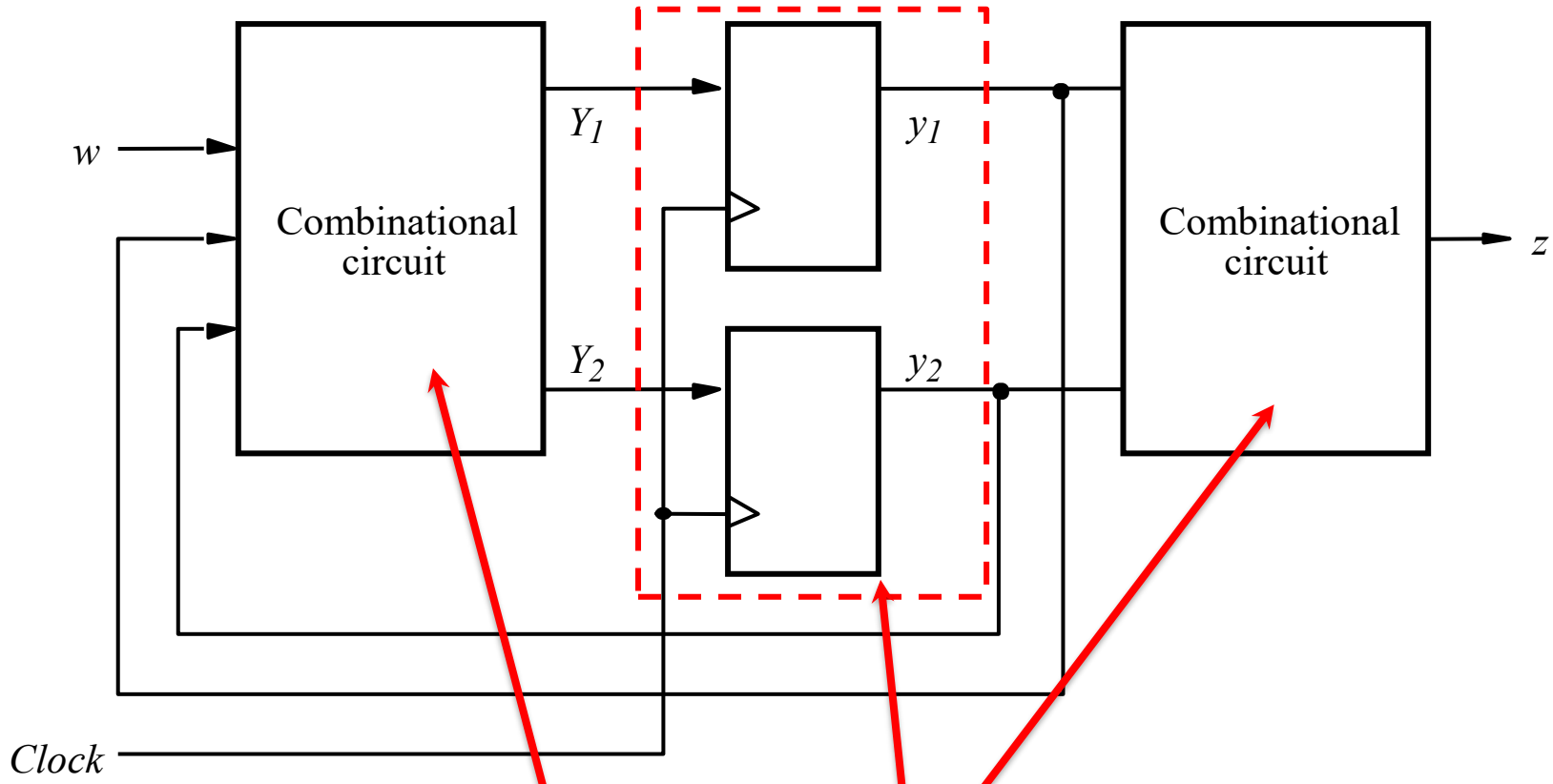
SCHOOL OF ELECTRICAL ENGINEERING & COMPUTER SCIENCE (SEECS)

Impact of State Encoding on Sequential Circuit Complexity

State Assignment Problem

A general sequential circuit with input *w*, output *z* and two state flip-flops

An obvious objective of the state-assignment process is to minimize the cost of implementation i.e Number of gates and flip-flops



How to encode the state flip-flops such that the Next-state and Output logic expressions are simplified?

Can we do better? State Assignment Problem

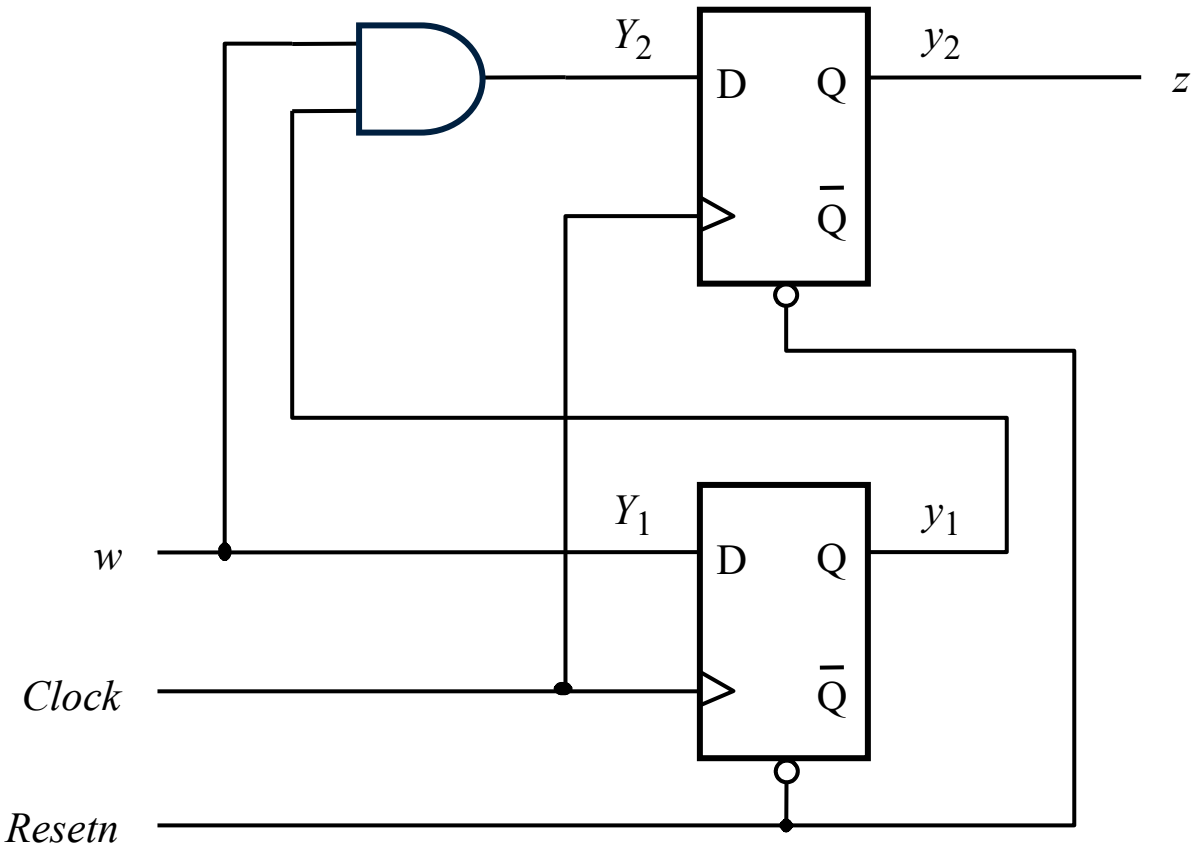
	Present state y_2y_1	Next state $w = 0$: Y_2Y_1	Next state $w = 1$: Y_2Y_1	Output z
A	00	00	01	0
B	01	00	10	0
C	10	00	10	1
	11	*dd*	*dd*	*d*

Binary Encoding

	Present state y_2y_1	Next state $w = 0$: Y_2Y_1	Next state $w = 1$: Y_2Y_1	Output z
A	00	00	01	0
B	01	00	11	0
C	11	00	11	1
	10	*dd*	*dd*	*d*

Gray-code Encoding

Implementation with Gray-Code State Encoding



Fewer Gates ->-> Lesser Implementation Cost

Other State-Encoding Options

- One-Hot:
  - Another interesting possibility is to use as many state variables as there are states.
  - In this method, for each state all but one of the state variables are equal to 0.
  - The variable whose value is 1 is deemed to be "hot." The approach is known as the *one-hot* encoding method.

- Two-Hot:
  - Two variables are 1 in any given state.

- Johnson:
  - Output of Johnson counter is used to encode the states,
    - complement output of the last flip-flop $\overline{Q_n}$ is back-fed to the first flip-flop in the chain.

Choosing THE Best State Encoding

- In general, circuits are much larger than our example, and different state assignments can have a substantial effect on the cost of the final implementation.
- It is often impossible to find the best state assignment for a large circuit.
- CAD tools usually perform the state assignment using heuristic techniques:
  - These techniques are usually proprietary, and their details are seldom published.

Specifying the State Encoding in an EDA Tool

- Verilog compilers usually have a capability to search for different assignments that may give better results:
  - such as attempting to use the one-hot encoding etc.

- The user can either allow the compiler to use its FSM-handling capability, or suppress it in which case the compiler simply deals with the Verilog statements in the usual way.

Specifying State Encoding in Intel Quartus Prime

General
Files
Libraries
IP Settings
IP Catalog Search Locations
Design Templates
Operating Settings and Conditions
Voltage
Temperature
Compilation Process Settings
Incremental Compilation
EDA Tool Settings
Design Entry/Synthesis
Simulation
Formal Verification
Board-Level
Compiler Settings
VHDL Input
Verilog HDL Input
Default Parameters
TimeQuest Timing Analyzer
Assembler
Design Assistant
SignalTap II Logic Analyzer
Logic Analyzer Interface
PowerPlay Power Analyzer Settings
SSN Analyzer

Compiler Settings

Specify high-level optimization settings for the Compiler (including integrated synthesis and fitting). These settings control the optimization focus and algorithms that will be performed throughout design compilation.

Advanced Analysis & Synthesis Settings

Specify the settings for the logic options in your project. Assignments made to an individual node or entity in the Assignment Editor will override the option settings in this dialog box.

state

Show: All

Name:	Setting:
Extract VHDL State Machines	On
Extract Verilog State Machines	On
Safe State Machine	Off
State Machine Processing	Auto

Auto
Gray
Johnson
Minimal Bits
One-Hot
Sequential
User-Encoded

Description:

Specifies the processing style used to compile a state machine. You can use your own 'User-Encoded' style, or select 'One-Hot', 'Minimal Bits', 'Gray', 'Johnson', 'Sequential' or 'Auto' (Compiler-selected) encoding.

Reset
Reset All
OK
Cancel
Help

'High Performance Effort' mode will cause the compiler to target increased positive timing margin (via Standard Fit compilation), increase the timing optimization effort applied during placement and routing, and enable timing-related Physical Synthesis optimizations (as allowed by the register optimization settings below). Each of these additional optimizations can increase compilation time. 'Aggressive Performance' mode enables the same optimizations as 'High Performance Effort' mode, and additionally enables options during Analysis & Synthesis to maximize design performance at a potential increase to logic area. If design utilization is

Example (Earlier Moore FSM): State Encoding Assigned by Xilinx Synthesizer

Using FSM-Auto Encoding Option:

Analyzing FSM <FSM_0> for best encoding.
Optimizing FSM <y1/FSM> on signal <y1[1:2]> with gray encoding.

```
-------------------
 State | Encoding
-------------------
 00    | 00
 01    | 01
 10    | 11
-------------------
```

Using FSM- One-Hot Encoding Option:

Optimizing FSM <y1/FSM> on signal <y1[1:3]> with one-hot encoding.

```
-------------------
 State | Encoding
-------------------
 00    | 001
 01    | 010
 10    | 100
-------------------
```

Example (Earlier Moore FSM): State Encoding Assigned by Xilinx Synthesizer

Using FSM- Gray Encoding Option:

Optimizing FSM <y1/FSM> on signal <y1[1:2]> with gray encoding.

```
-------------------
 State | Encoding
 00    | 00
 01    | 01
 10    | 11
-------------------
```

Using FSM- Sequential Encoding Option:

Optimizing FSM <y1/FSM> on signal <y1[1:2]> with sequential encoding.

```
-------------------
 State | Encoding
-------------------
 00    | 00
 01    | 01
 10    | 10
```

Results: Courtesy of Aneela Sabir [MSEE-8]

Watch Out!!!

Watch Out!!!

- Intel Quartus Prime software doesn't infer a state machine if the state transition logic uses arithmetic similar to the following example:

```
case (state)
    0: begin
        if (ena) next_state <= state + 2;
        else next_state <= state + 1;
        end
    1: begin
    ...
endcase
```

- Intel Quartus Prime software doesn't infer a state machine if the state variable is an output.
- Intel Quartus Prime software doesn't infer a state machine for signed variables.

https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/hb/qts/archives/qts-qpp-5v1-17-1.pdf

Template vs. IP

Recall: Two Main Styles of HDL Implementation

- **Structural (Gate-Level)**
  - The module body contains gate-level description of the circuit
  - Describe how modules are interconnected
  - Each module contains other modules (instances)
  - … and interconnections between these modules
  - Describes a hierarchy

- **Behavioral**
  - The module body contains functional description of the circuit
  - Contains logical and mathematical operators
  - **Level of abstraction is higher than gate-level**
    - Many possible gate-level realizations of a behavioral description

- **Practical circuits would use a combination of both**

HDL Templates

- Using HDL templates:
  - Predictable logic blocks are inferred by the synthesizer
  - Portable design across different devices.
  - Lesser control over advanced parameters.

DEMO

IP Core

- Instantiate IP Cores:
  - Allows to take advantage of the architectural feature in the target device
    - » Memory
    - » DSP blocks
    - » Multipliers
    - » Adders
    - » Clock generation [Phase-locked loops]
    - » SERDES
    - » etc
  - Allows you the use advanced features.
  - Easy customization of ports and parameters.
  - Not portable to a different design.

DEMO

Vendor Synthesis Guide Should be your Companion!!!

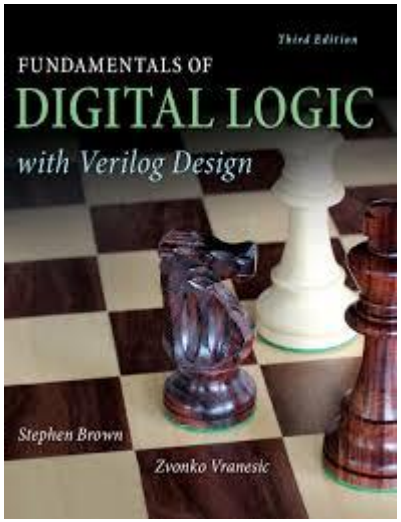
- Intel: https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/ug/ug-qps-design-recommendations.pdf

- Xilinx:

https://www.xilinx.com/support/documentation/sw_manuals/xilinx2017_1/ug901-vivado-synthesis.pdf

Recommended Reading

- Digital System Design with Verilog HDL, 3/e, b **S**tephen Brown and **Z**vonko Vranesic. [**S&Z**]
  - S&Z,
    - Chapter-6



THANK YOU