Digital Electronics: Sequential Logic

Further Considerations

Elimination of Redundant States

- Sometimes, when designing state machines it is possible that unnecessary states may be introduced
- In general, reducing the number of states may reduce the number of FFs required and may also reduce the complexity of the next state logic owing to the presence of more unused states (don’t cares)

Elimination of Redundant States - Example

- Consider the following State Table that corresponds with a Mealy Machine implementation
- This is so, since the inputs and outputs from the machine are on the transitions (arcs) between states
- The following state table is drawn in a compact form by incorporating the 2 possible input values as parallel columns within both the next state and output columns of the table

Example

Current State	Next State X=0	Next State X=1	Output (Z) X=0	Output (Z) X=1
A	B	C	0	0
B	D	E	0	0
C	F	G	0	0
D	H	I	0	0
E	J	K	0	0
F	L	M	0	0
G	N	P	0	0
H	A	A	0	0
I	A	A	0	0
J	A	A	0	1
K	A	A	0	0
L	A	A	0	1
M	A	A	0	0
N	A	A	0	0
P	A	A	0	0

- From the table, we see that there is no way of telling states H and I apart, so we can replace I with H when it appears in the Next State portion of the table

Example

Current State	Next State X=0	Next State X=1	Output (Z) X=0	Output (Z) X=1
A	B	C	0	0
B	D	E	0	0
C	F	G	0	0
D	H	H	0	0
E	J	K	0	0
F	L	M	0	0
G	N	P	0	0
H	A	A	0	0
J	A	A	0	1
K	A	A	0	0
L	A	A	0	1
M	A	A	0	0
N	A	A	0	0
P	A	A	0	0

- We also see that there is now no way to get to state I so we can remove row I from the table
- Similarly, rows K, M, N and P have the same next state and output as H and can be replaced by H

Example

Current State	Next State X=0	Next State X=1	Output (Z) X=0	Output (Z) X=1
A	B	C	0	0
B	D	E	0	0
C	F	G	0	0
D	H	H	0	0
E	J	H	0	0
F	L	H	0	0
G	H	H	0	0
H	A	A	0	0
J	A	A	0	1
L	A	A	0	1

- Similarly, there is now no way to get to states K, M, N and P and so we can remove these rows from the table
- Also, the next state and outputs are identical for rows J and L, thus L can be replaced by J and row L eliminated from the table

Example

Current State	Next State X=0	Next State X=1	Output (Z) X=0	Output (Z) X=1
A	B	C	0	0
B	D	E	0	0
C	F	G	0	0
D	H	H	0	0
E	J	H	0	0
F	J	H	0	0
G	H	H	0	0
H	A	A	0	0
J	A	A	0	1

- Now rows D and G are identical, as are rows E and F.
- Consequently, G can be replaced by D, and row G eliminated. Also, F can be replaced by E and row F eliminated from the table

Example

Current State	Next State X=0	Next State X=1	Output (Z) X=0	Output (Z) X=1
A	B	C	0	0
B	D	E	0	0
C	E	D	0	0
D	H	H	0	0
E	J	H	0	0
H	A	A	0	0
J	A	A	0	1

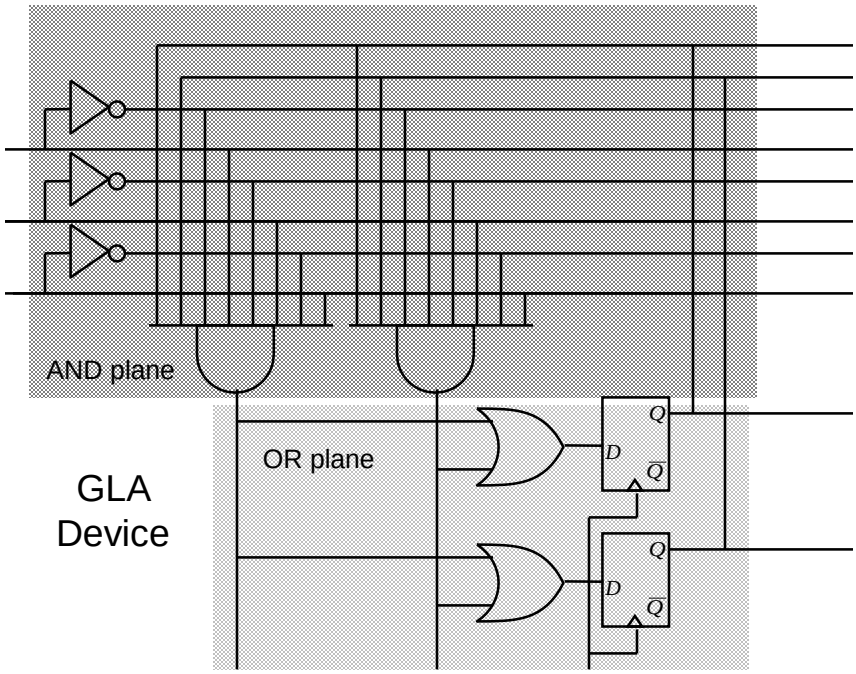
- The procedure employed to find equivalent states in this example is known as *row matching*.
- However, we note row matching is not sufficient to find all the equivalent states except for certain special cases

Implementation of FSMs

- We saw previously that programmable logic can be used to implement combinational logic circuits, i.e., using PLA devices
- PAL style devices have been modified to include D-type FFs to permit FSMs to be implemented using programmable logic
- One particular style is known as Generic Logic Array (GLA)

GLA Devices

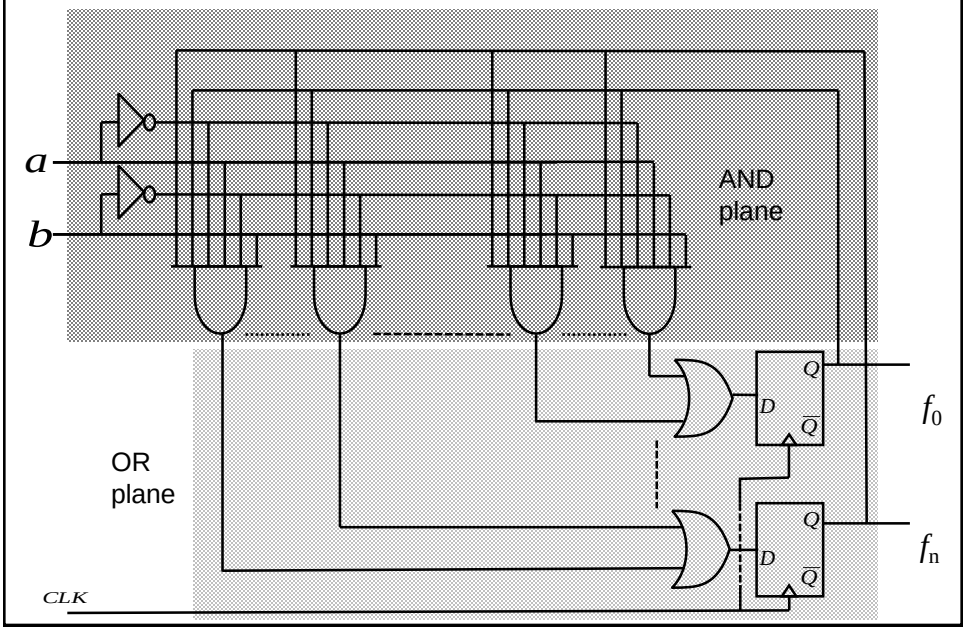
- They are similar in concept to PLAs, but have the option to make use of a D-type flip-flops in the OR plane (one following each OR gate). In addition, the outputs from the D-types are also made available to the AND plane (in addition to the usual inputs)
  - Consequently it becomes possible to build programmable sequential logic circuits



GLA Devices

- A modified form of a GLA known as a Generic Array Logic (GAL) is used in the Hardware Laboratory classes to implement various FSMs.

GAL Devices



FPGA

- Field Programmable Gate Arrays (FPGAs) are the latest type of programmable logic
- Are an array of configurable logic blocks (CLBs) surrounded by Input Output Blocks (IOBs):
  - programmable routing channels permit CLBs to be connected to other CLBs and to IOBs
  - CLBs contain look up tables (LUTs), multiplexers (MUXs) and D-type FFs
  - The FPGA is configured by specifying the contents of the LUTs and select signals for the MUXs

FPGA – Xilinx Spartan

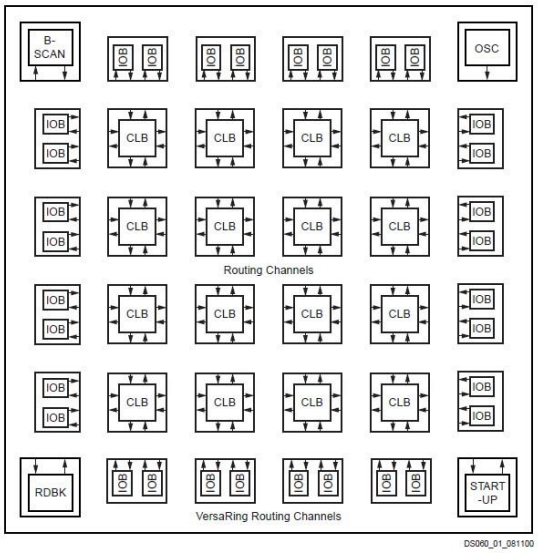
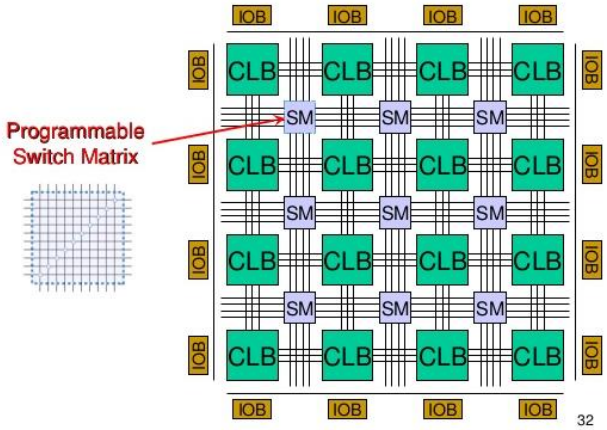


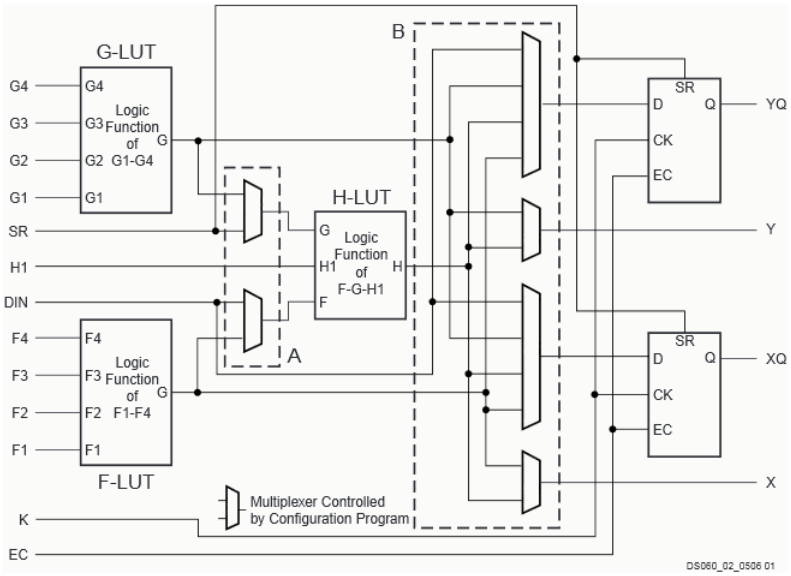
Figure 1: **Basic FPGA Block Diagram**

FPGA – Xilinx Spartan

- Simplified schematic showing CLBs and programmable routing channels, i.e., wires plus programmable switch matrices (SMs)



FPGA - Spartan CLB



FPGA - Spartan CLB

- Has 2, 4-input LUTs (F and G) and 1, 3 input LUT (H)
- Has to ‘combinational’ outputs (Y and X) and 2 ‘registered’ outputs (i.e., from D-FFs) YQ and XQ
- Depending on MUX configuration Y is given by output of either G or H LUTs and X from either F or H LUTs.
- D-FF inputs come from DIN, or from F, G, or H LUTs

FPGA - Spartan CLB

- Thus each CLB can perform up to 2 combinational and/or 2 registered functions
- All functions can involve at least 4 input variables (e.g., G1 to G4, and F1 to F4), but can be up to 9 (owing to the possibility of implementing 2-level combinational logic functions), i.e., G1 to G4, F1 to F4, H1.
- Created using either a schematic (block) diagram or more likely a Hardware Description Language (HDL) of the design

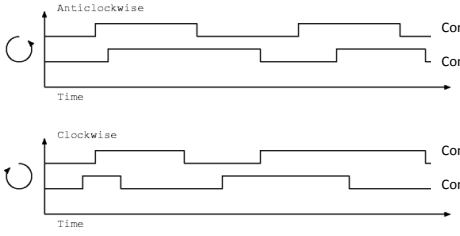
FPGA - Spartan CLB

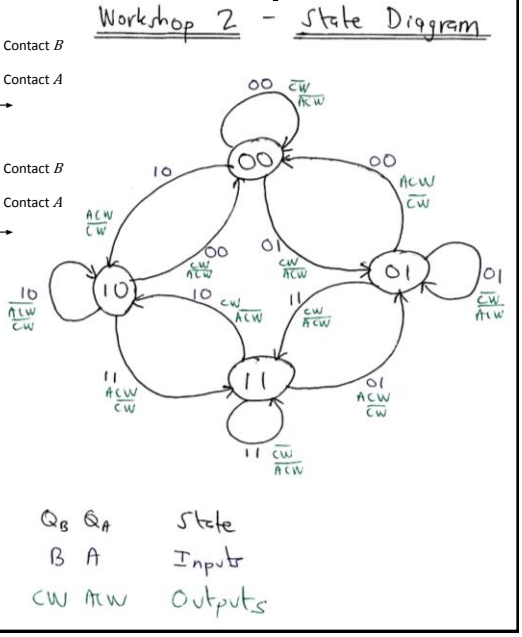
- The synthesis tool determines how the LUTs, MUXs and routing channels are configured
- This configuration information is then downloaded to the FPGA
- Xilinx devices store their configuration information in static RAM (SRAM) so can be easily reprogrammed
- The SRAM contents can be downloaded either from a computer or from an EEPROM device when the system is powered-up

FPGA

- Other FPGA manufacturers are available, e.g., Altera.
- Particular manufacturers have many different product lines
- Main differences will be the no. of CLBs, the structure of the CLBs, internal or external ROM, additional features such as specialised arithmetic blocks, user RAM etc.

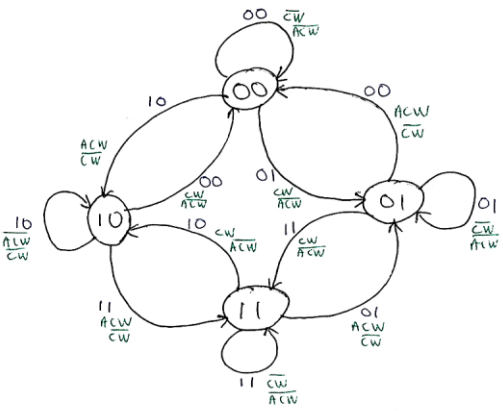
Appendix – Workshop 2





Appendix – Workshop 2

Workshop 2 – State Diagram

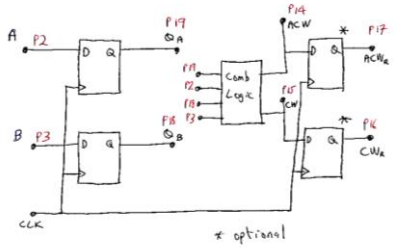


Q_B Q_A State

B A Inputs

CW ACW Outputs

Workshop 2 MEALY SOLUTION – REGISTERED



Q_B Q_A State register outputs

B, A Switch contacts

P2, P3 PAL Inputs

Current State Q_B P18	Current State Q_A P19	Next State = Input B P3	Next State = Input A P2	Output CW P15	Output ACW P14
0	0	0	0	0	0
0	0	0	1	1	0
0	0	1	0	0	1
⋮		⋮		⋮	
Plus don't cares,					
0	0	1	1	x	x
0	1	1	0	x	x
1	0	0	1	x	x
1	1	0	0	x	x