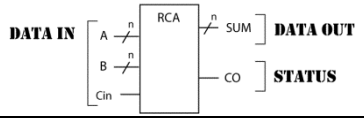
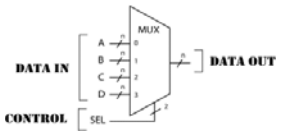
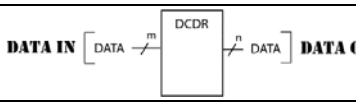
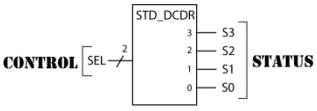
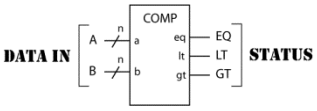
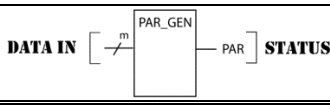
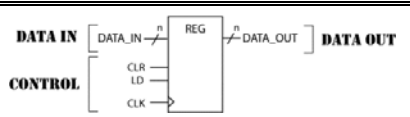
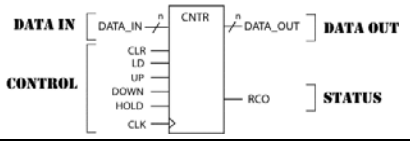
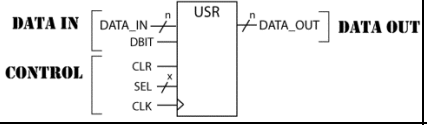
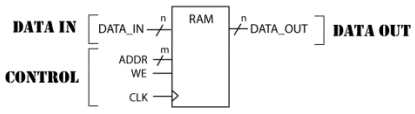
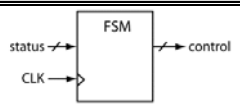
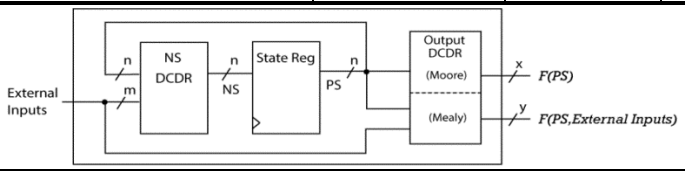
Digital Design Foundation Model Cheatsheet

		Circuit Diagram	Data IN	Control IN	Data OUT	Status OUT
CombInatorial	**RCA**		A B Cin	-	SUM	Co
	MUX		Multiple DATA_IN	SEL	DATA_OUT	-
	Generic Decoder (LUT)		DATA	-	DATA	-
	Standard Decoder		-	SEL	-	STATUS
	Comparator		A B	-	-	EQ GT LT
	Parity Generator		DATA_IN	-	-	PAR
SequentIal	**Register**		DATA_IN	CLK LD CLR	DATA_OUT	-
	Counter		DATA_IN	CLK LD, CLR UP/DOWN HOLD	DATA_OUT	RCO
	Shift Register		DATA_IN, DBIT	CLK, SEL CLR, DBIT DATA_IN	DATA_OUT	-
	RAM		IN_DATA -	CLK WE ADDR	OUT_DATA -	IN_DATA -
				Inputs		**Outputs**
	FSM		-	CLK status	-	control
FSM Model						

Mealy's Laws of Digital Design

Mealy's First Law of Digital Design: If in doubt, draw some black box diagrams.

Justification: You always know enough to draw the top-level BBD interface. When you start drawing black boxes and listing what you know, you generate ideas on how to solve the problem.

Mealy's Second Law of Digital Design: If your digital design is running into weird obstacles that require kludgy solutions, toss out the design and start over from square one.

Justification: There is never one correct circuit to solve a digital design problem, which means there are many paths to take when working on a digital design problem. You're inevitably going to take the wrong path, so be ready to realize as much, and switch to a different path.

Mealy's Third Law of Digital Design: Every digital design problem can have many different but equivalent solutions; the absolute right solution is eternally elusive.

Justification: Digital circuitry is inherently flexible, which allows you to solve digital design problems in different ways. The digital design must be familiar enough with digital circuitry to be able to create their designs to satisfy the design criteria and to then verify their designs work as expected.

Mealy's Fourth Law of Digital Design: The digital design process is circular, not linear. If you think you're going to generate the correct solution with the first pass, you're bound for disappointment. The digital design process is circular; always make going backwards a few steps to fix issues part of the design process. Don't try to make your design perfect from the get-go, make it simple to understand so that you can fix issues as they arise.

Justification: Based on Mealy's Second and Third Laws, you always need to be willing to go temporarily backwards on your designs. Design, go back and make necessarily refinement, design some more, repeat.

Mealy's Fifth Law of Digital Design: Model circuits using many smaller sub-modules as opposed to fewer larger sub-modules; as this approach supports testing and increases the chances module reuse.

Justification: Large designs are harder to understand and test, particularly if you're first passing off your models to an HDL synthesizer. Make your designs reliant upon a strong foundation of basic digital modules is always the best approach.

Mealy's Sixth Law of Digital Design: Don't rely on the HDL synthesizer; create your HDL models by having a remote vision of what underlying hardware should look like in terms of standard digital modules.

Justification: Although HDLs give you the ability to model digital circuits, they are not magic. The HDL synthesizer's task is to convert pages of text into circuits; the more the options you give to the synthesizer, the less probable the synthesizer will successfully generate a circuit that works as you intended.

Mealy's Seventh Law of Digital Design: Always first consider modeling a digital circuit or part of a digital circuit using some type of decoder. Decoders in digital design are anything we can describe in a tabular format, so they are essentially look-up tables (LUTs).

Justification: The basis of all digital design is defining circuits in a tabular format, whenever possible. Although this approach represents low-level design, HDL tools have strong support for table-based models.