Tassadaq Hussain

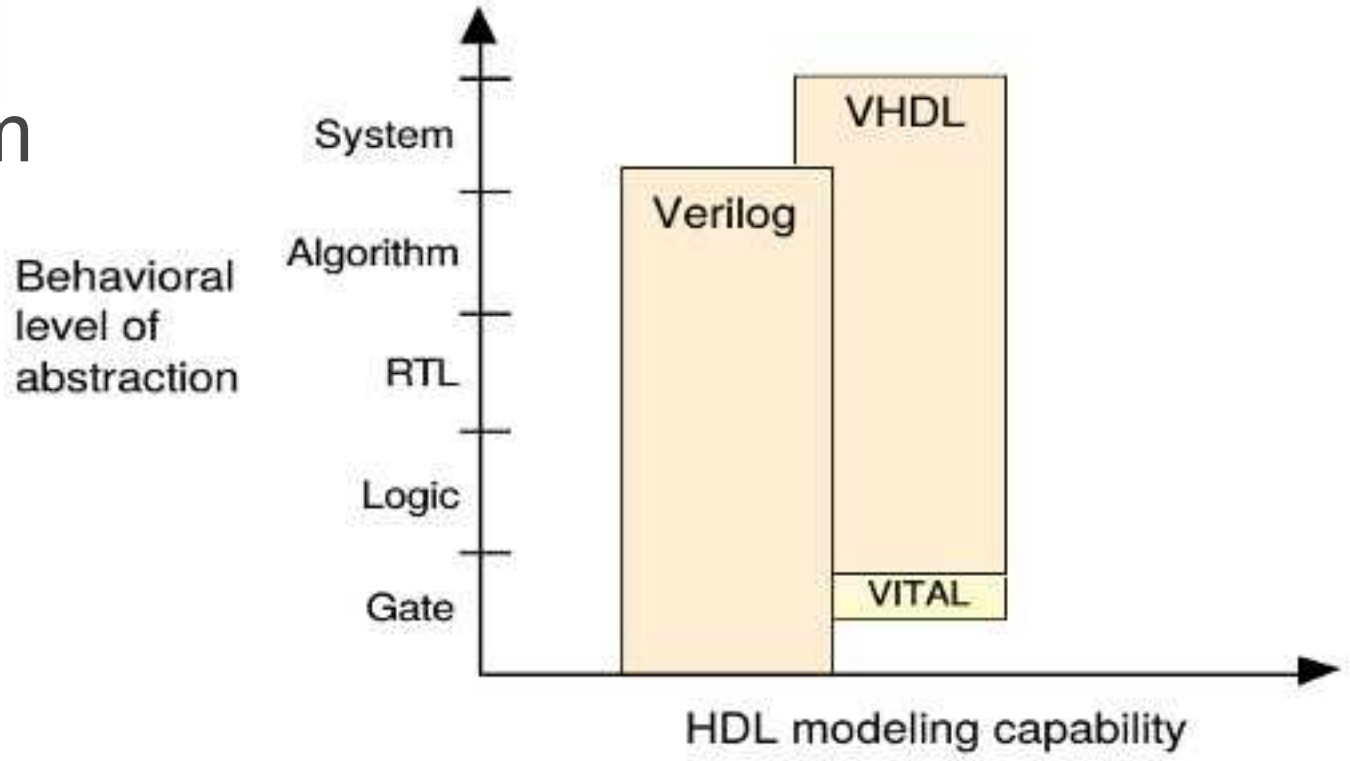
VLSI Design

Register Transfer Level Modeling

Outline

Verilog

Modelsim



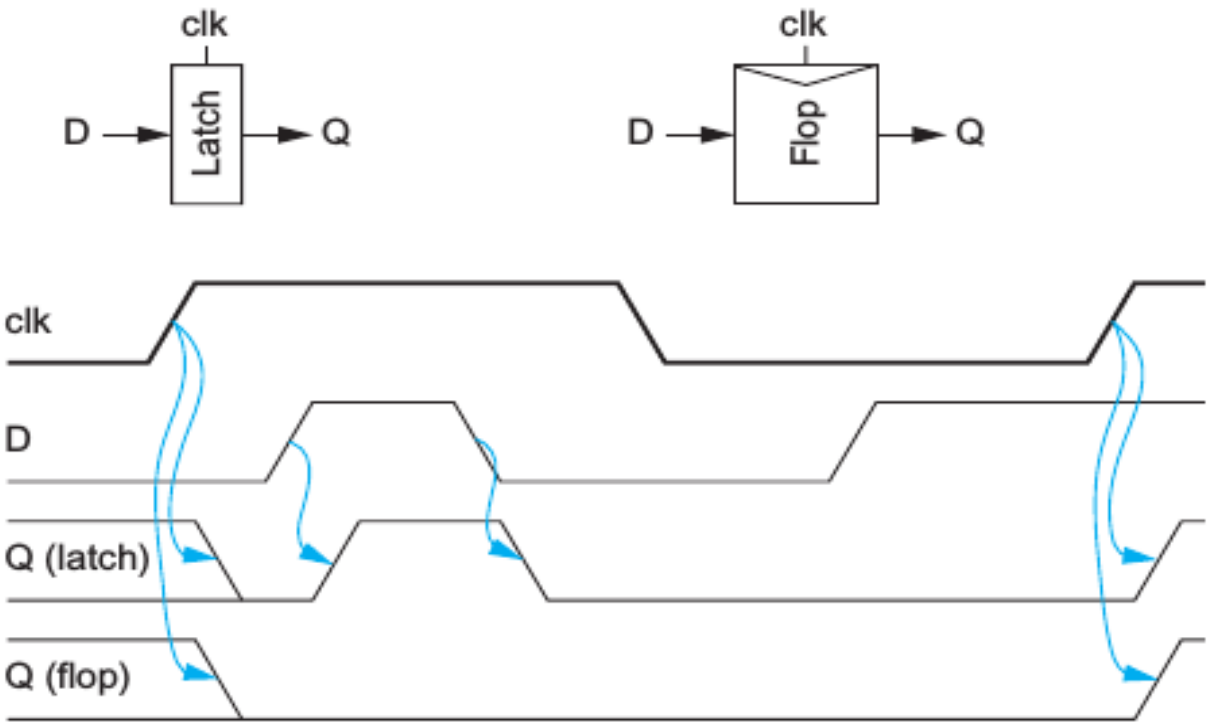
Difference between Combination Logic and Sequential Logic

In **combinational circuits** the output is a function of the current inputs.

In **sequential circuits** the output depends on previous as well as current inputs; such circuits are said to have state.

Finite state machines and pipelines are two important examples of sequential circuits.

Latch and Flipflop



Latch and FlipFlops

```
always @( posedge Clock or reset)
begin
Sig1 = A & B;
Y1
 = Sig1 &C: //single flip-flop inferred
end

Process (posedge clock)
```

Term	Name
t_{pd}	Logic Propagation Delay
t_{cd}	Logic Contamination Delay
t_{pcq}	Latch/Flop Clock-to-Q Propagation Delay
t_{ccq}	Latch/Flop Clock-to-Q Contamination Delay
t_{pdq}	Latch D-to-Q Propagation Delay
t_{cdq}	Latch D-to-Q Contamination Delay
t_{setup}	Latch/Flop Setup Time
t_{hold}	Latch/Flop Hold Time

Contamination Delay

contamination delay is the minimum amount of time from when an input changes until any output starts to change its value.

Propagation Delay

The propagation delay, or gate delay, is the length of time till when the output of a logic gate becomes stable and valid.

Setup Time

Setup time is the minimum amount of time the data signal should be held steady before the clock event so that the data is reliably sampled by the clock.

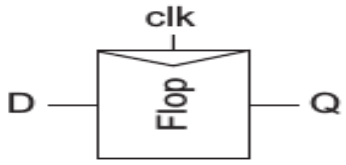
Hold Time

Hold time is the minimum amount of time the data signal should be held steady after the clock event so that the data are reliably sampled.

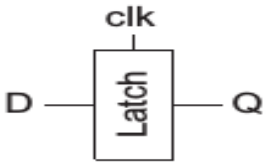
Timing


A
Combinational Logic
Y

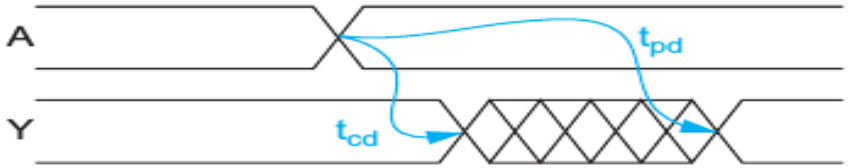
(a)

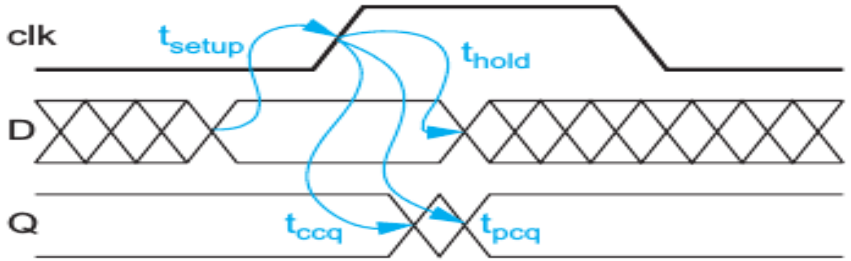

clk
D
Flop
Q

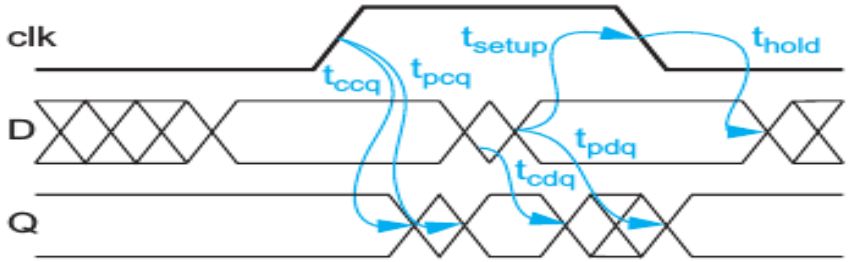
(b)

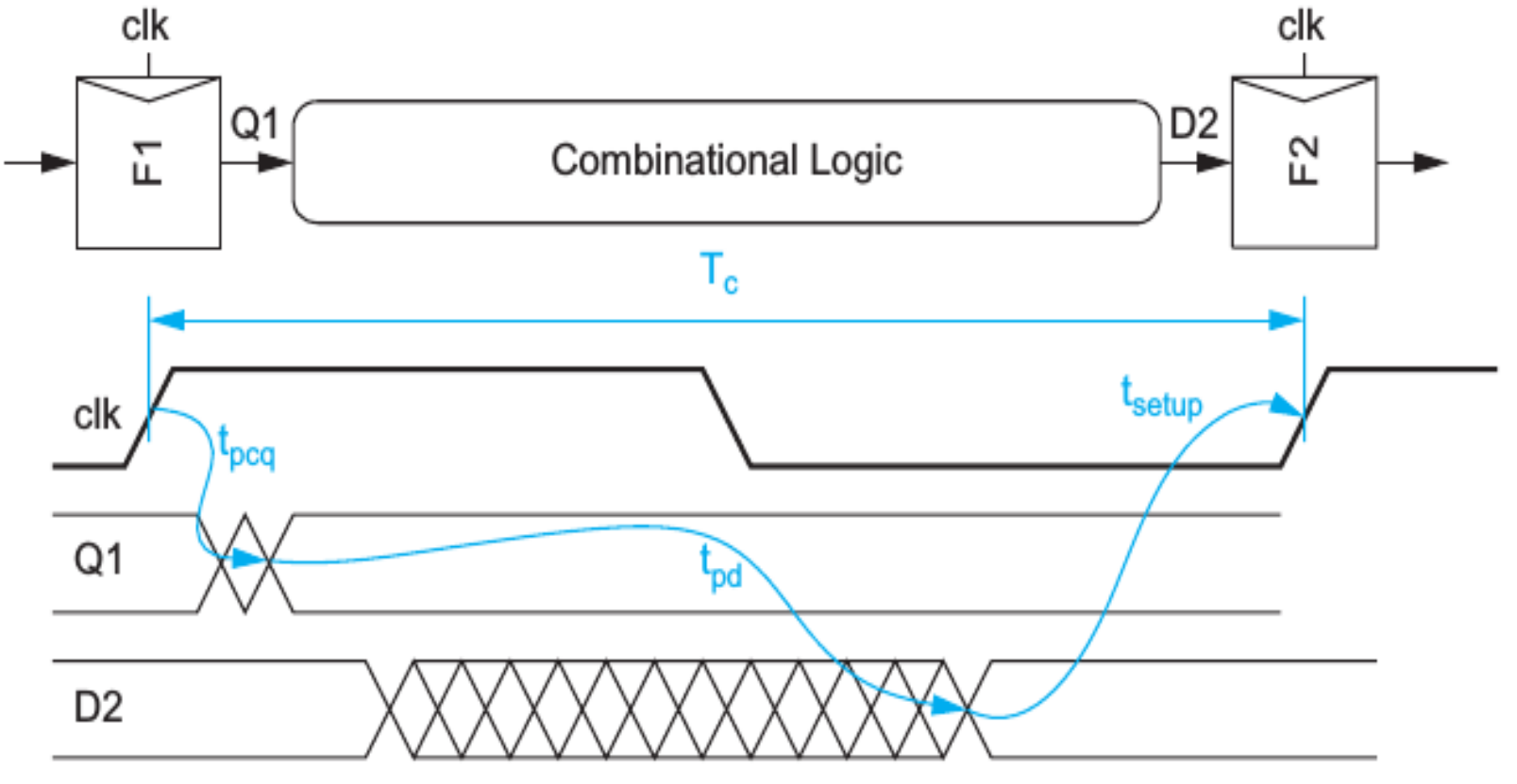

clk
D
Latch
Q

(c)


A
Y
t_{pd}
t_{cd}


clk
D
Q
t_{setup}
t_{hold}
t_{ccq}
t_{pcq}


clk
D
Q
t_{ccq}
t_{pcq}
t_{setup}
t_{hold}
t_{pdq}
t_{cdq}


clk
clk
F1
Q1
Combinational Logic
D2
F2
T_c
clk
t_{pcq}
t_{setup}
Q1
t_{pd}
D2

Clock Skew

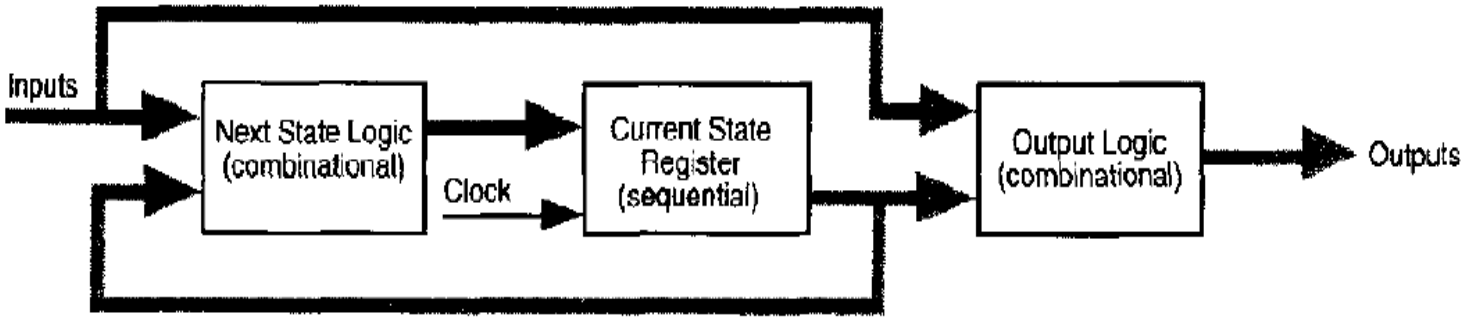
In reality, clocks have some uncertainty in their arrival times called clock skew.

Latch and FF based Design

```
always @( reset or clock or d)
begin
if(reset)
begin
end
end
always @( posedge clock or posedge d)
If (clock ==1)
```

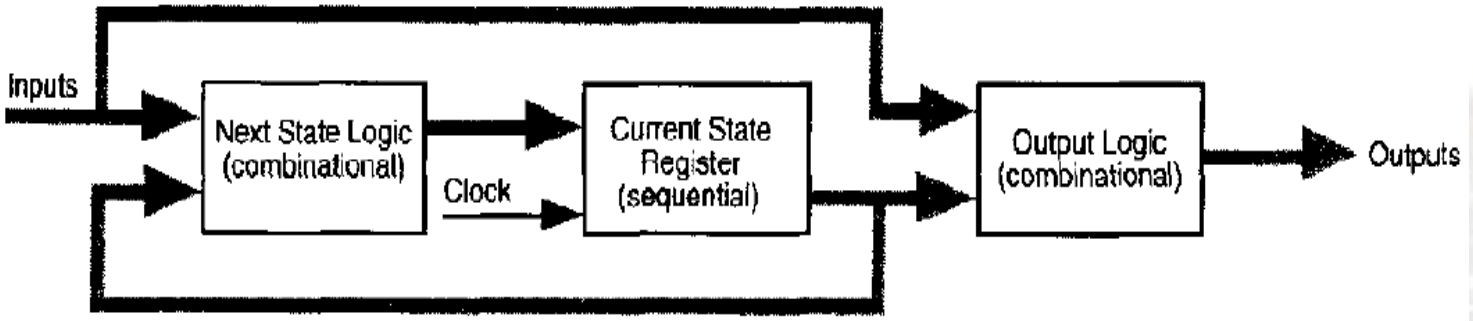
Finite State Machine

A FSM is any circuit specifically designed to sequence through specific patterns of states in a predetermined sequential manner, and which conforms to the structure shown in Figure. A state is represented by the binary value held on the current state register.
The FSM structure consists of three parts and may, or may not, be reflected in the structure of the HDL code that is used to model it.



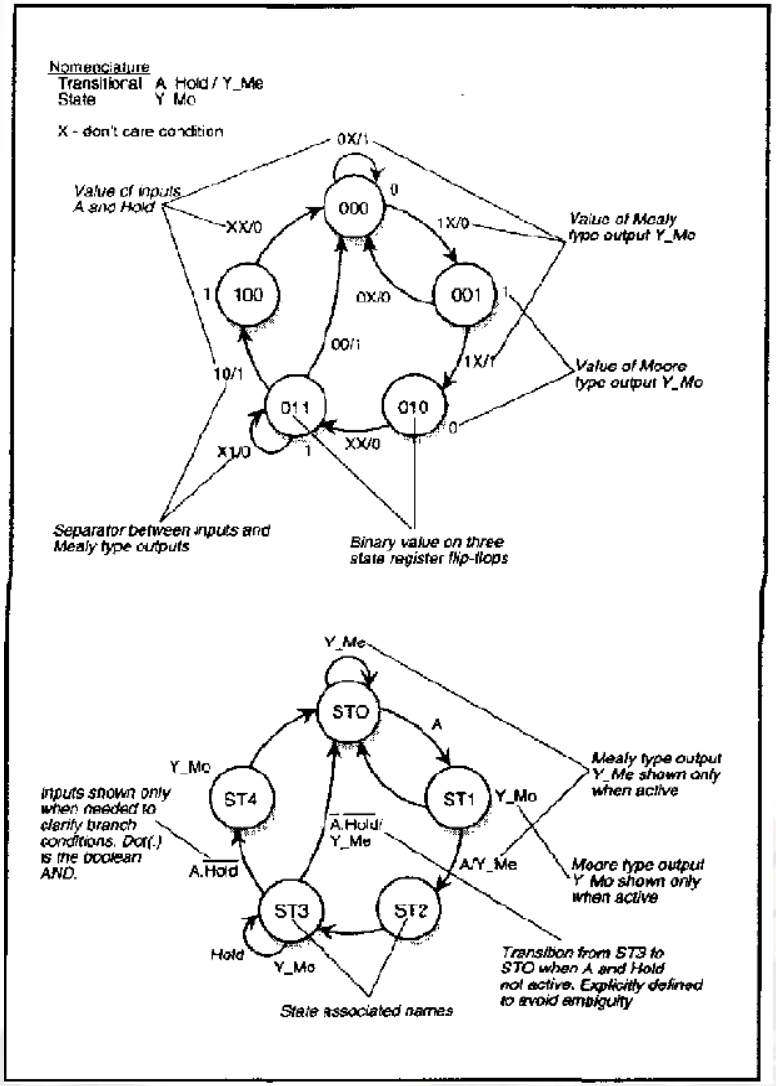
FSM

1. Current State Register. Register of n-bit flip-flops used to hold the current state of the FSM. Its value represents the current stage in the particular sequence of operations being performed. When operating, it is clocked from a free running clock source.
2. Next State Logic. Combinational logic used to generate the next stage (state) in the sequence. The next state output is a function of the state machine's inputs and its current state.
3. Output Logic. Combinational logic is used to generate required output signals. Outputs are a function of the state register output and possibly state machine inputs.



State table and state diagrams

Inputs		Current state		Next state		Outputs	
A	Hold					Y_Me	Y_Mo
0	X	000	(ST0)	000	(ST0)	1	0
1	X	000	(ST0)	001	(ST1)	0	0
0	X	001	(ST1)	000	(ST0)	0	1
1	X	001	(ST1)	010	(ST2)	1	1
X	X	010	(ST2)	011	(ST3)	0	0
X	1	011	(ST3)	011	(ST3)	1	1
0	0	011	(ST3)	000	(ST0)	1	1
1	0	011	(ST3)	100	(ST4)	0	1
X	X	100	(ST4)	000	(ST0)	0	1



Condition/State Operation

if

```
if (condition)
    statements;

if (condition)
    Statements;
else
    statements;

if (condition)
    statements;
else if (condition)
    Statements;
else
    statements;
```

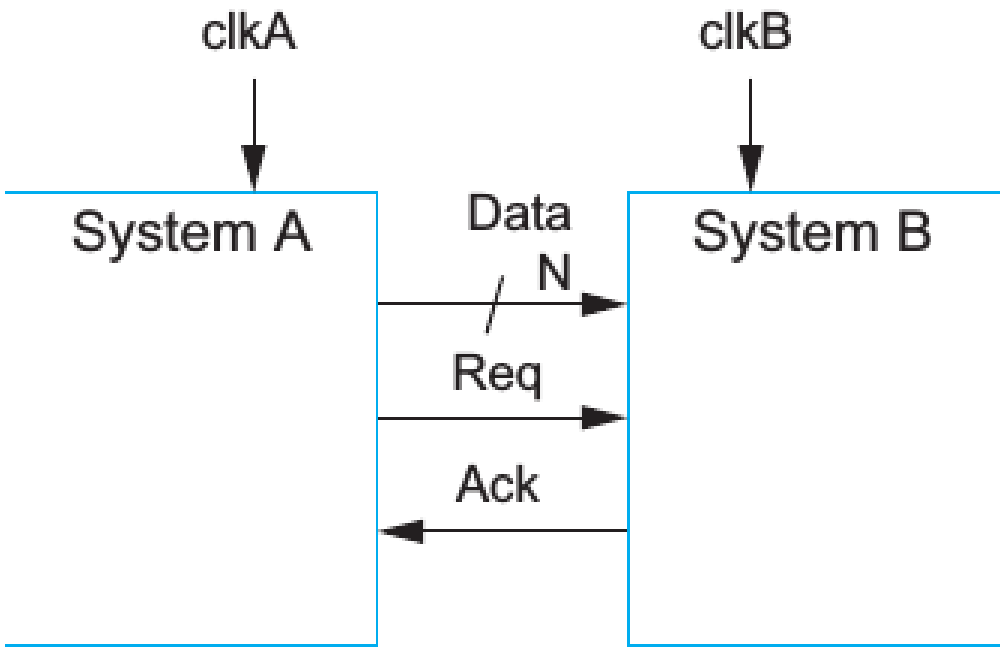
Case statement

```
reg [1:0] sel;
case (sel)
0 : y = a;
1 : y = b;
2 : y = c;
3 : y = d;
default : $display("Error in SEL");
endcase
```

Synchronization

A synchronizer is a circuit that accepts an input that can change at arbitrary times and produces an output aligned to the synchronizer's clock.

Synchronization



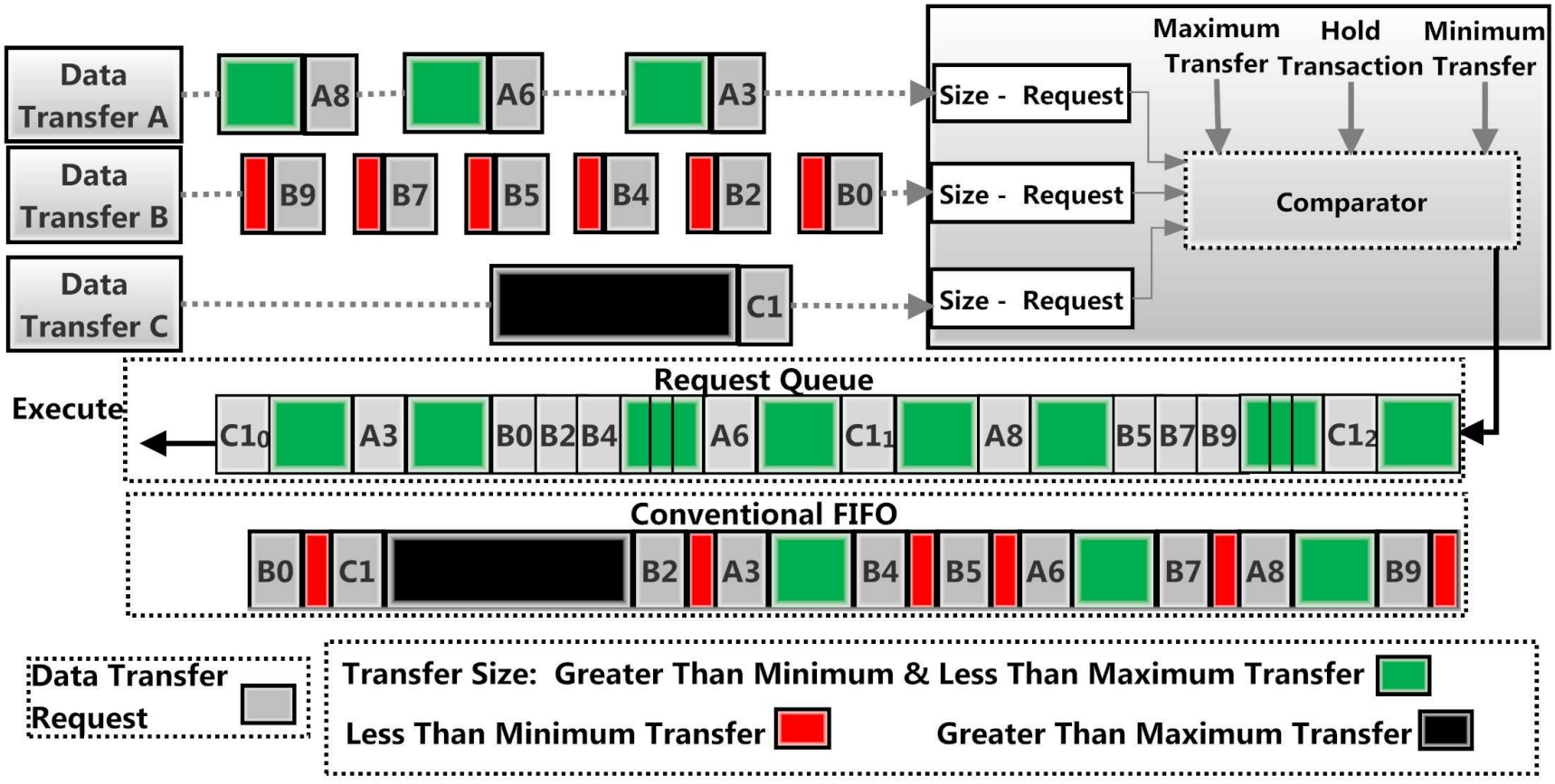
Arbiters

The arbiter determines which of two (multiple) inputs arrived first.

Scheduler

Scheduler creates a link (allocate resource) to one of the multiple inputs based on programmed or run-time priorities

Request, Hold and Grant



Tasks

Write Encoder and Decoder IN RTL

Write A StateMachine RTL code for Two Traffic Signals

Understand Blocking and Nonblocking Assignments with codes.

Write Unique code in RTL