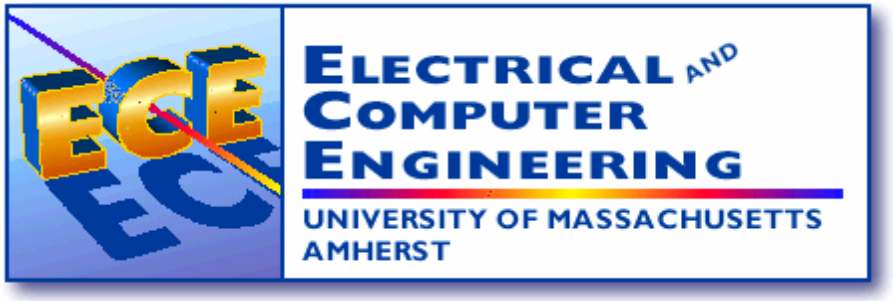
ENGIN 112

Intro to Electrical and Computer Engineering

Lecture 37

Register Transfer Level

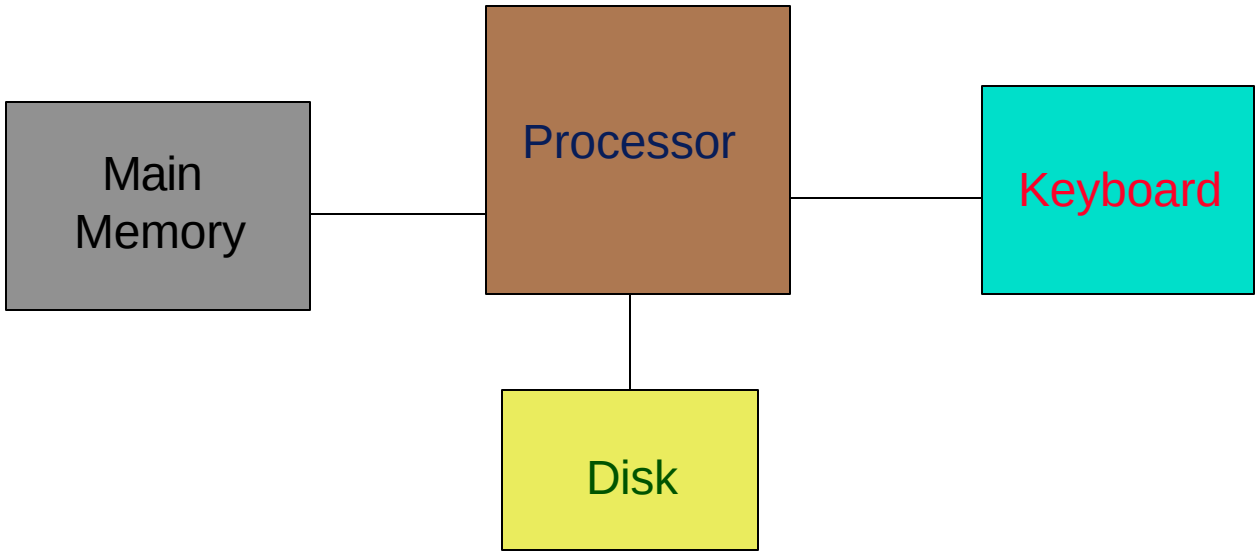


Overview

- System design must be modular
  - Easier to represent designs with system-level blocks
- Register transfer level represents transfers between clocked system registers
  - Shifts, arithmetic, logic, etc.
- Algorithmic state machine
  - Alternate approach to representing state machines
- Status signals from datapath used for control path
- Algorithmic state machine chart shows flow of computation

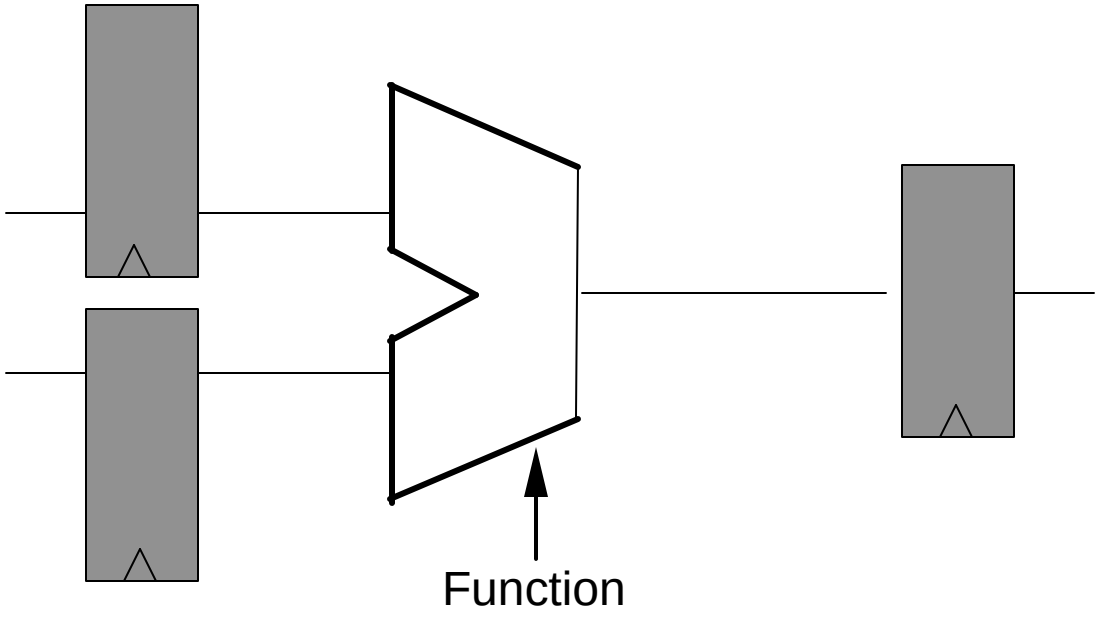
System-level Design

- Difficult to represent a design with just one state machine
- A series of control and data paths used to build computer systems
- Helps simplify design and allow for design changes



Registers and Data Operations

- **Activity and performance in computers defined on register-to-register paths**
- **Digital system at register transfer level specified by three components**
  - **The set of registers in the system**
  - **Operations to be performed on registers**
  - **Control that is applied to registers and sequence of operations**



Representation of Register Transfer Flow

- **Arrow indicates transfer from one register to another**
  - **R2 ? R1**
- **Conditional statements can help in selection**
  - **If (T1 = 1) then R2 ? R1**
- **Clock signal is not generally included in register transfer level statement**
  - **Sequential behavior is implied**
- **Multiple choices also possible**
  - **If (T1 = 1) then (R2 ? R1, R2 ? R2)**

How could these statements be implemented in hardware?

Other representative RTL operations

- **Addition**
  - **R1 ? R1 + R2**
- **Increment**
  - **R3 ? R3 + 1**
- **Shift right**
  - **R4 ? R4**
- **Clear**
  - **R5 ? 0**
- **Transfer doesn't change value of data begin moved**

How could these statements be implemented in hardware?

Algorithmic State Machines (ASM)

- Flowchart specifies a sequence of procedural steps and decision steps for a state machine
- Translates word description into a series of operations with conditions for execution
- Allows for detailed description of control and datapath

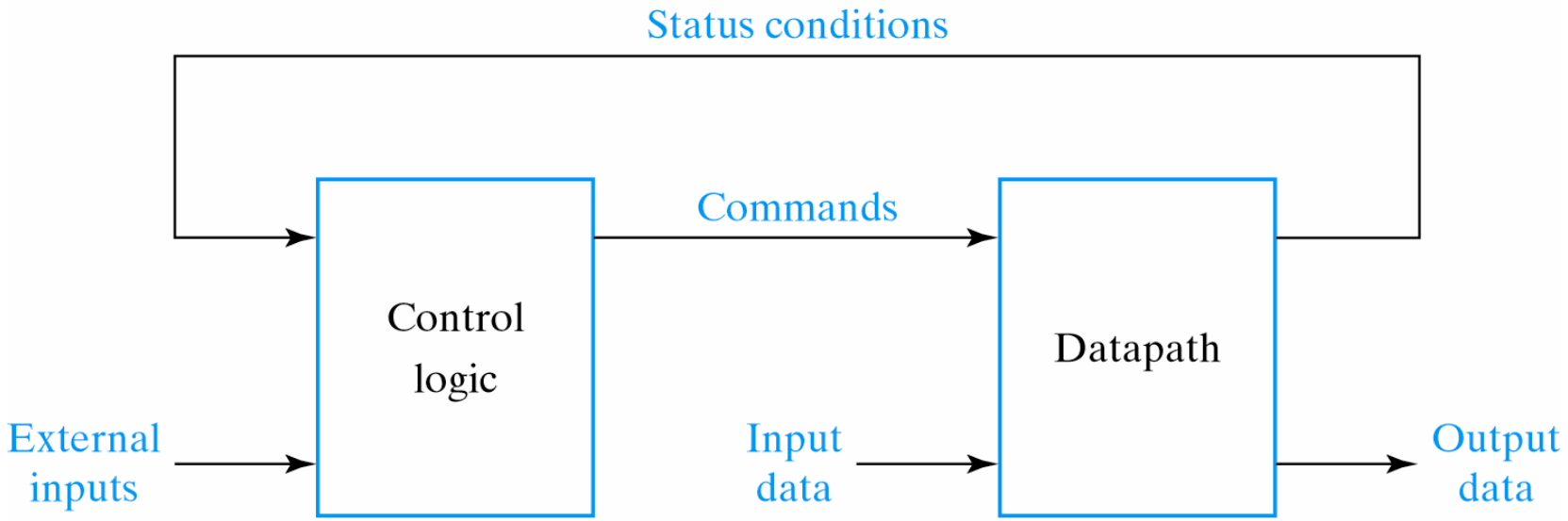


Fig. 8-2 Control and Datapath Interaction

Algorithmic State Machines – State Box

- **ASM describes operation of a sequential circuit**
- **ASM contains three basic elements**
  - **State box**
  - **Decision box**
  - **Condition box**
- **State box indicates an FSM state**
  - **Box also indicates operation to be performed**
- **Binary code and state name also included**

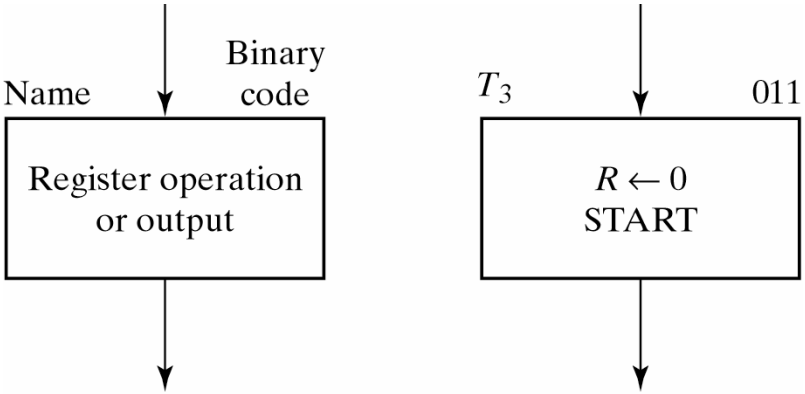


Fig. 8-3 State Box

Decision Box

- Describes the impact of input on control system
- Contains two exit paths which indicate result of condition
- More complicated conditions possible
- Implemented in hardware with a magnitude comparator

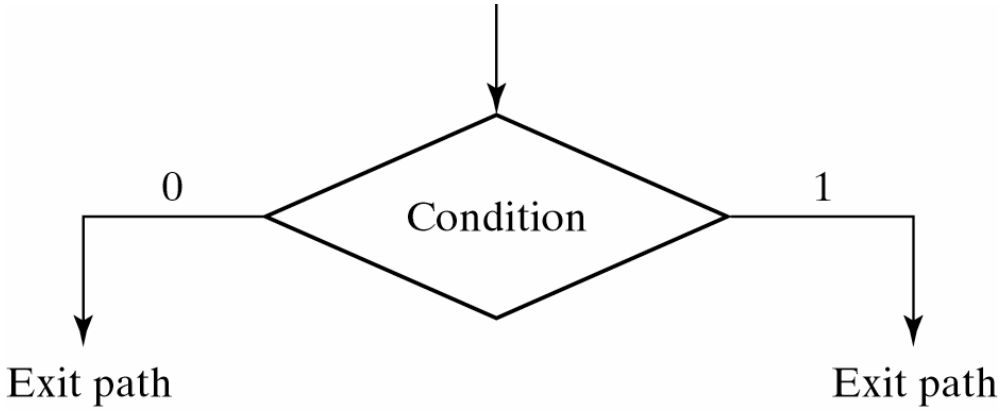


Fig. 8-4 Decision Box

Conditional Box

- Indicates assignments following a decision box
- Generally indicates data transfer

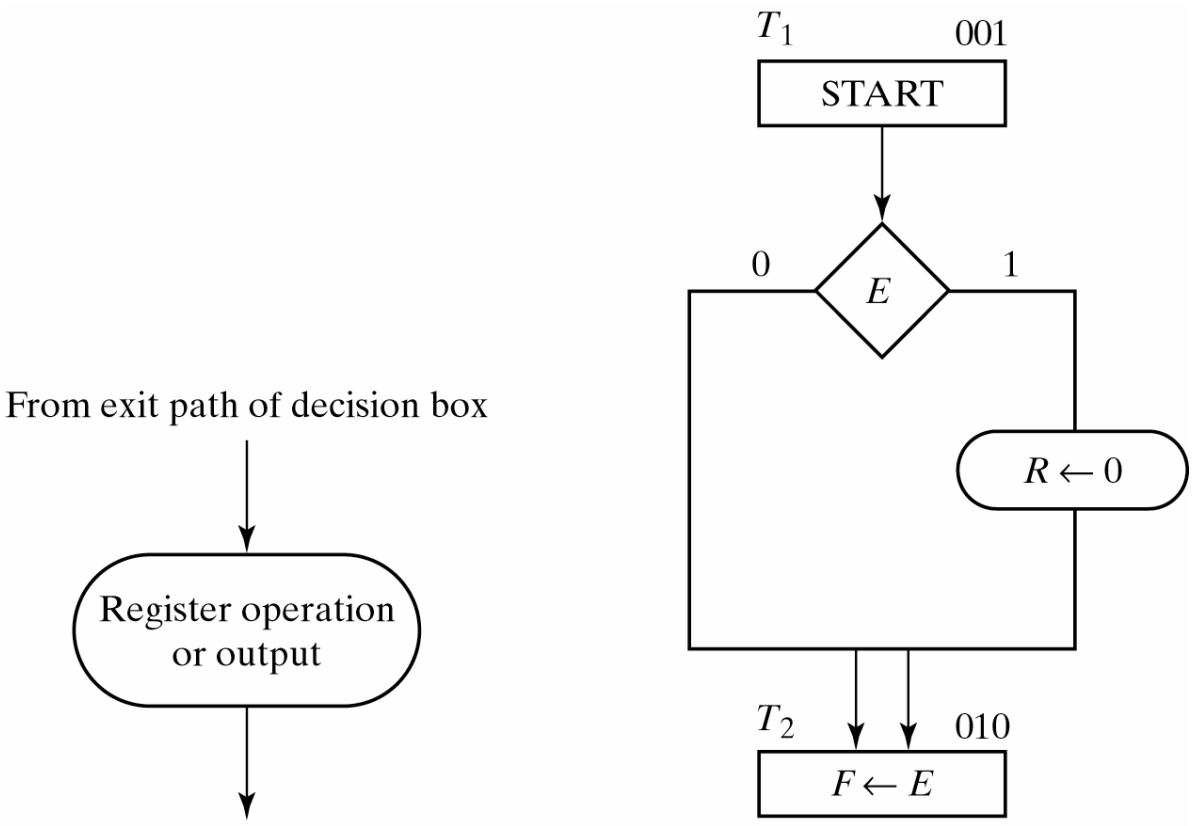


Fig. 8-5 Conditional Box

ASM Block

- Paths exist between state boxes
- Each state box equivalent to one state

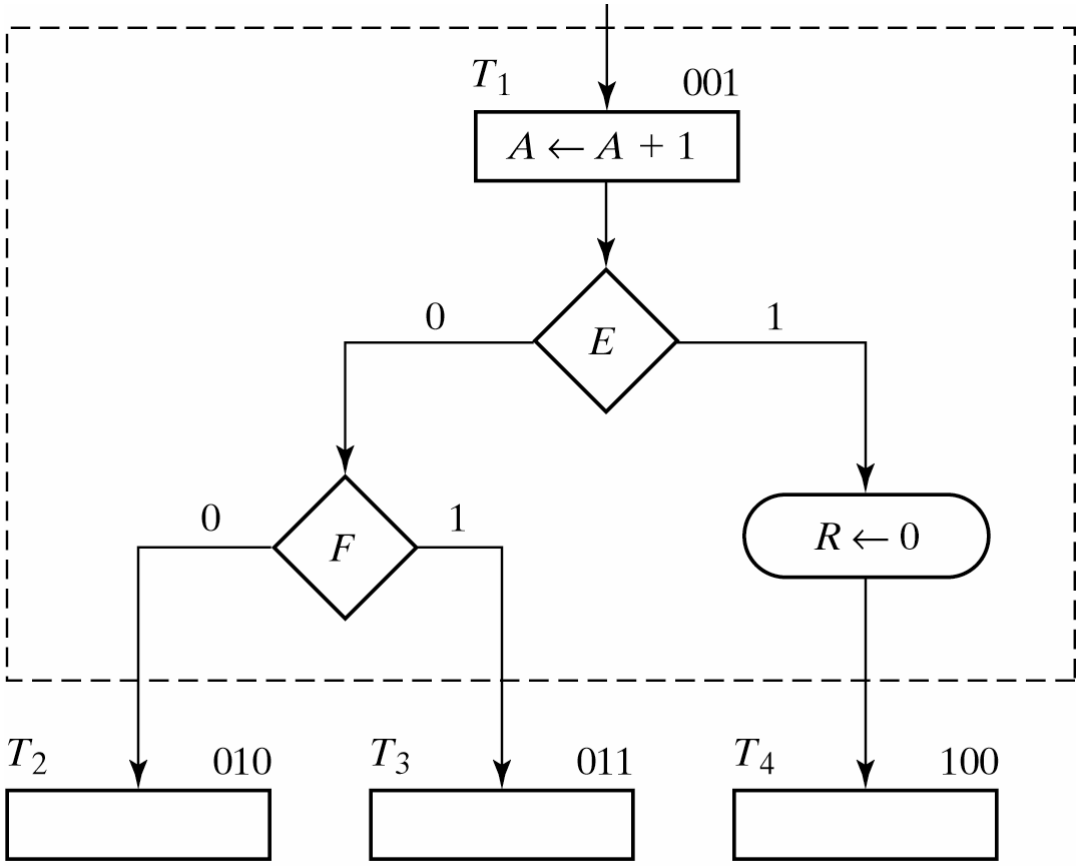


Fig. 8-6 ASM Block

ASM Block

- **Equivalent to State Diagram**

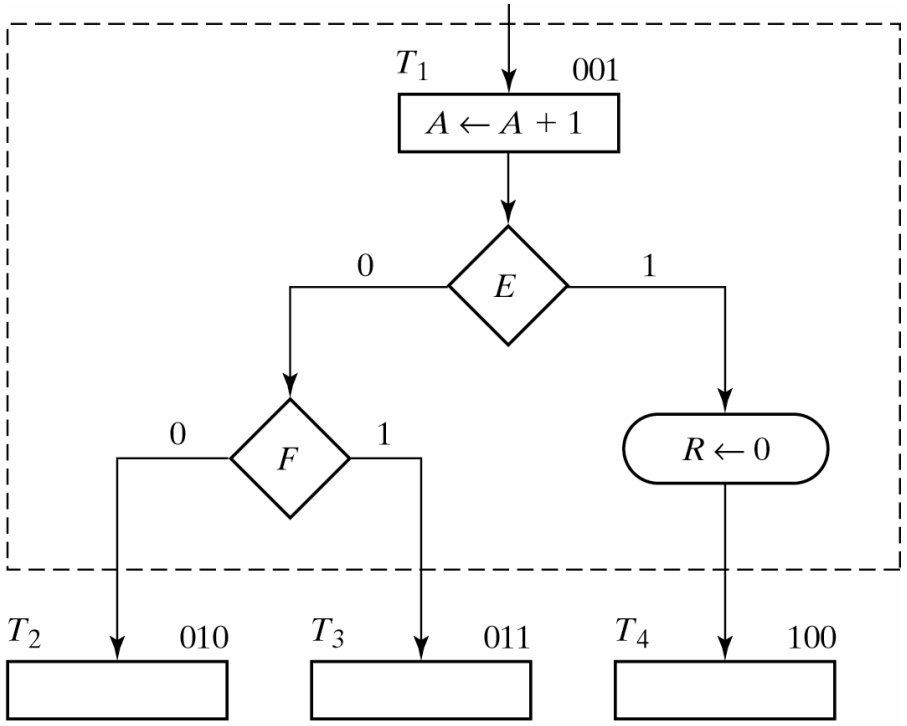


Fig. 8-6 ASM Block

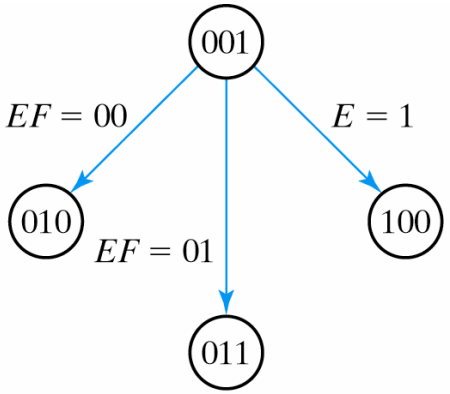
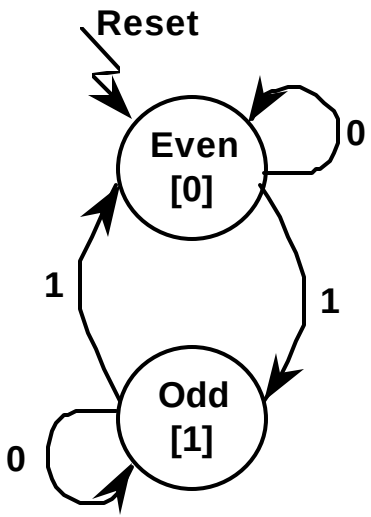


Fig. 8-7 State Diagram Equivalent to the ASM Chart of Fig. 8-6

Concept of the State Machine

Example: Odd Parity Checker

Assert output whenever input bit stream has odd # of 1's



State Diagram

Present State	Input	Next State	Output
Even	0	Even	0
Even	1	Odd	0
Odd	0	Odd	1
Odd	1	Even	1

Symbolic State Transition Table

Present State	Input	Next State	Output
0	0	0	0
0	1	1	0
1	0	1	1
1	1	0	1

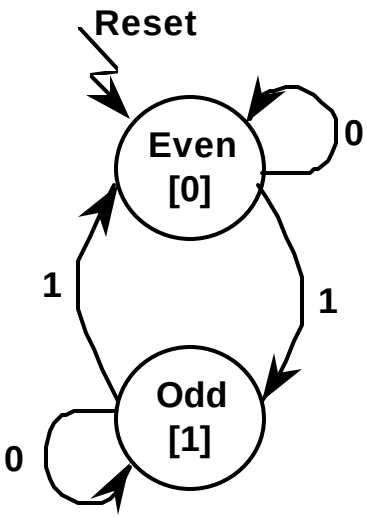
Encoded State Transition Table

- **Note: Present state and output are the same value**
  - **Moore machine**

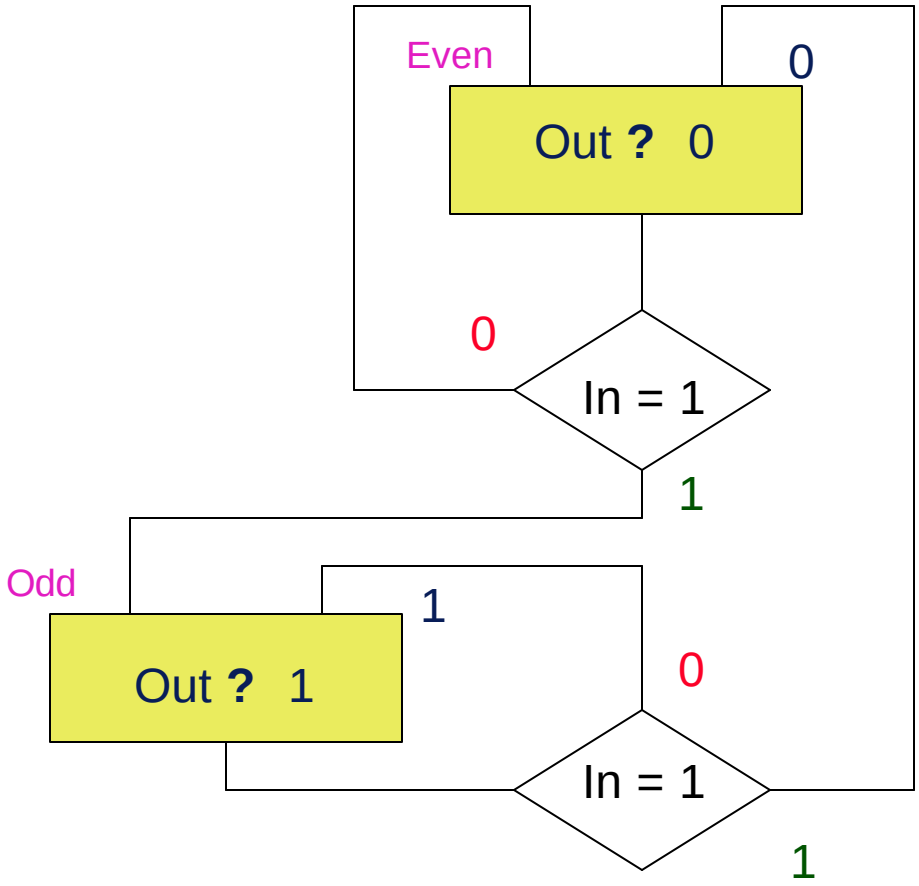
ASM for Odd Parity Checker

Example: Odd Parity Checker

Assert output whenever input bit stream has odd # of 1's



State Diagram



Verilog Representation for RTL

- **Conditional assignment statements**
  - **assign Y = S ? I1 : I0;**
- **Statement evaluation**

```
always @ (I1 or I2 or S)
    if (S) Y = I1;
    else  Y = I0;
```

Perform evaluation only when an input changes

```
always @(posedge clk)
   q = d;
```

Verilog description of a flip flop

Typical Design Flow

- **It all starts with Verilog description**

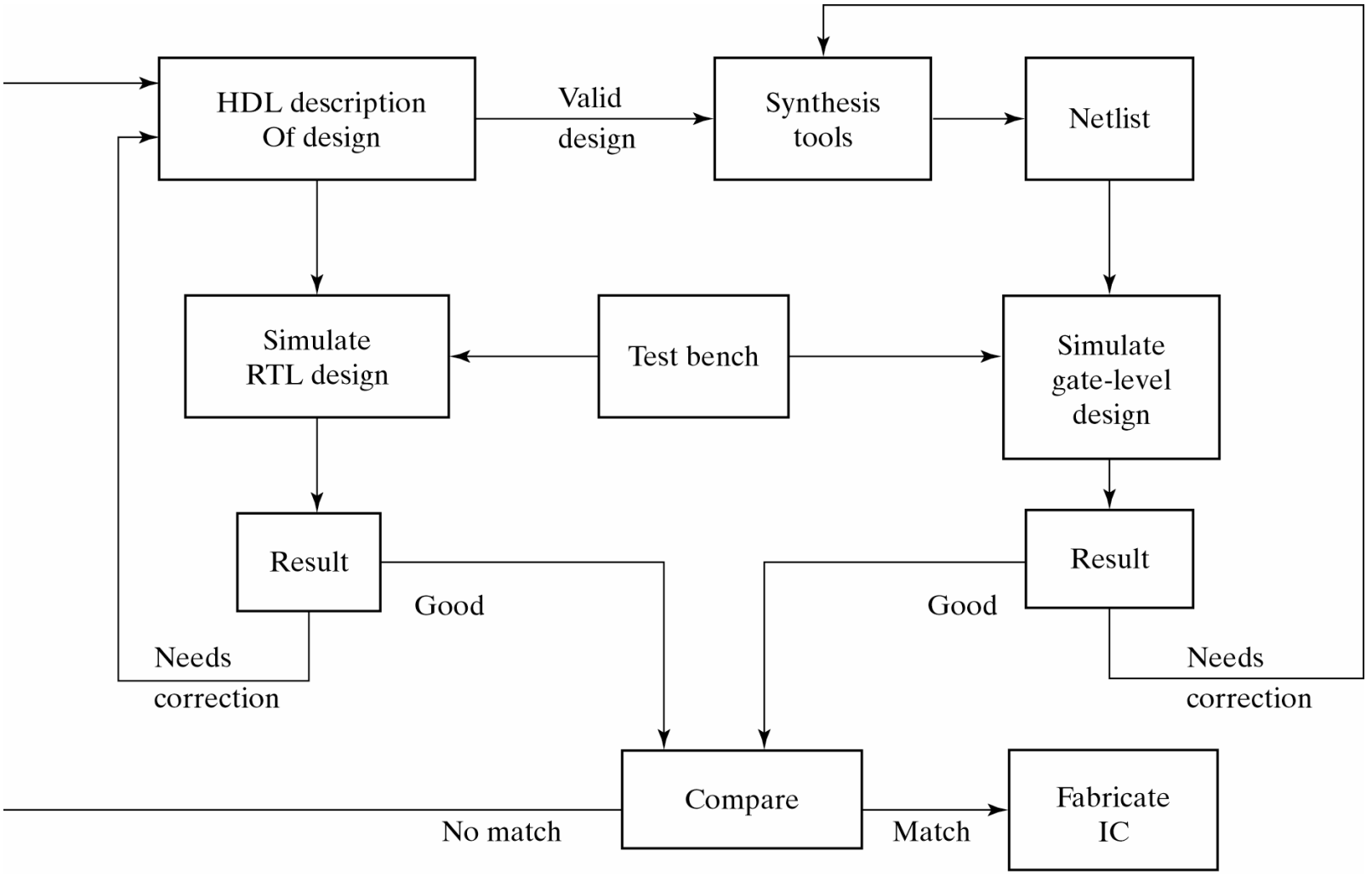


Fig. 8-1 Process of HDL Simulation and Synthesis

Summary

- Register transfer level provides a simple way to describe designs
- A series of operations take place between registers
- Algorithmic state machine another way to represent a state machine
- Direct correspondence between state diagram and algorithmic state machine
- Possible to implement state machines in Verilog
  - Also in VHDL
- Next time: programmable array logic