EECS 3201: Digital Logic Design Lecture 14

Ihab Amer, PhD, SMIEEE, P.Eng.

Modular Approach

- A digital system is a sequential logic system constructed with flip-flops and gates
- Number of states in actual digital systems is typically high
- This makes it difficult to represent large digital systems with state tables
- Hence, digital systems are typically designed using a modular approach
- A system is partitioned into modular subsystems, each of which performs some functional task
- Examples of subsystems are registers, decoders, multiplexers, arithmetic elements, and control logic
- Modules are interconnected with common data and control paths

Register Transfer Level (RTL)

- A digital system is represented at the RTL level by the following three components:
  - The set of registers in the system
  - Operations performed on data stored in registers (e.g. transfer, arithmetic, logic, and shift)
  - Control that supervises the sequence of operations in the system

Examples of RTL Statements

■ R2 $\leftarrow$ R1	
■ If(T1 = 1) then (R2 $\leftarrow$ R1)	
■ If(T3 = 1) then (R2 $\leftarrow$ R1, R1 $\leftarrow$ R2)	
■ R1 $\leftarrow$ R1 + R2	(Add contents of R2 to R1)
■ R3 $\leftarrow$ R3 + 1	(Increment R3 by 1)
■ R4 $\leftarrow$ shr R4	(Shift right R4)
■ R5 $\leftarrow$ 0	(Reset R5 to 0)

RTL in HDL

```
assign S = A + B;
```

Combinational Logic

```
always @ (A or B)
    S = A + B;
```

Sequential Logic

```
always @ (posedge clock)
    begin
        RA = RA + RB;
        RD = RA
    end
```

```
always @ (negedge clock)
    begin
        RA <= RA + RB;
        RD <= RA
    end
```

Accurately models synchronous sequential circuits

Looping Statements

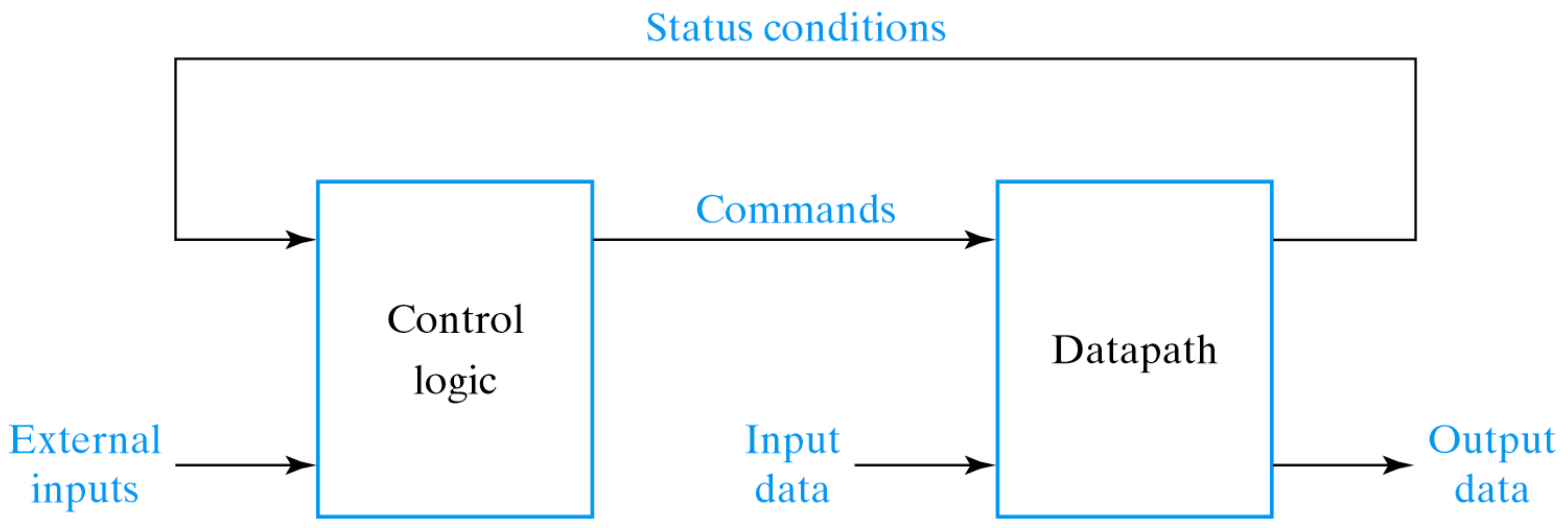
```
//description of 2x4 decoder
//using for-loop statement
module decoder (IN, Y);
   input  [1:0] IN;     //Two binary inputs
   output [3:0] Y;      //Four binary outputs
   reg [3:0] Y;
   integer I;           //control variable for loop
   always @ (IN)
            for (I = 0; I <= 3; I = I + 1)
              if (IN == I) Y[I] = 1;
              else Y[I] = 0;
endmodule
```

Looping statements (e.g. *repeat*, *forever*, *while*, and *for*) must appear inside an **initial** or **always** block

Refer to Mano textbook for examples of other types of loops

```
if (IN == 00) Y[0] = 1;  else Y[0] = 0;
if (IN == 01) Y[1] = 1;  else Y[1] = 0;
if (IN == 10) Y[2] = 1;  else Y[2] = 0;
if (IN == 11) Y[3] = 1;  else Y[3] = 0;
```

Structure of a Typical Digital System



Control and Datapath Interaction

Control Unit (Control)

- Controls Data Movements in the Execution Unit by Switching Multiplexers and Enabling or Disabling Resources
- Follows Some ‘Program’ or Schedule
- Often Implemented as Finite State Machine or collection of Finite State Machines

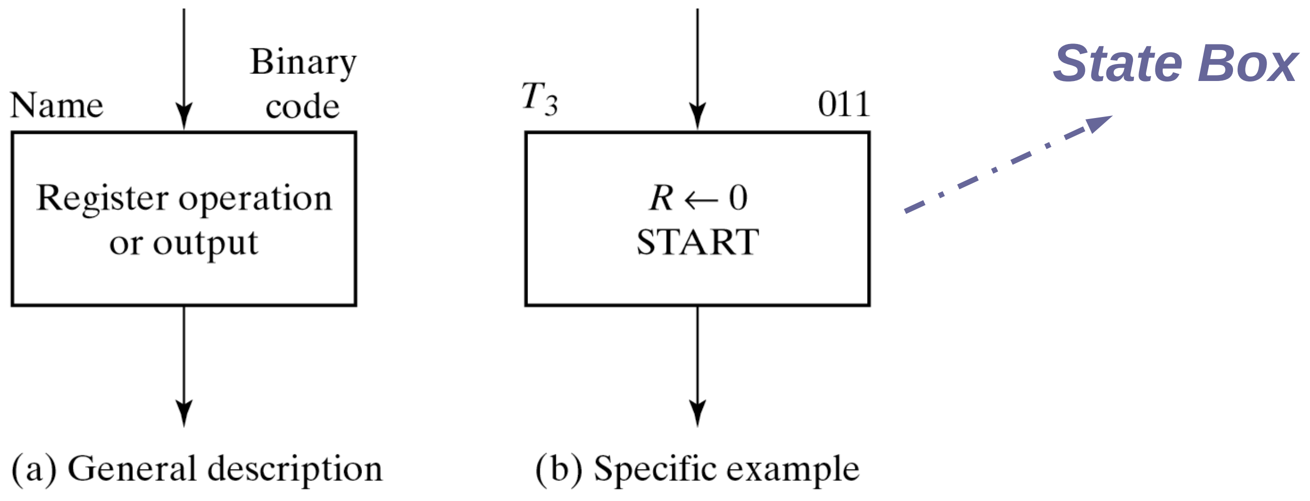
Execution Unit (Datapath)

- Provides All Necessary Resources and Interconnects Among Them to Perform Specified Task
- Examples of Resources
  - Adders, Multipliers, Registers, Memories, etc.

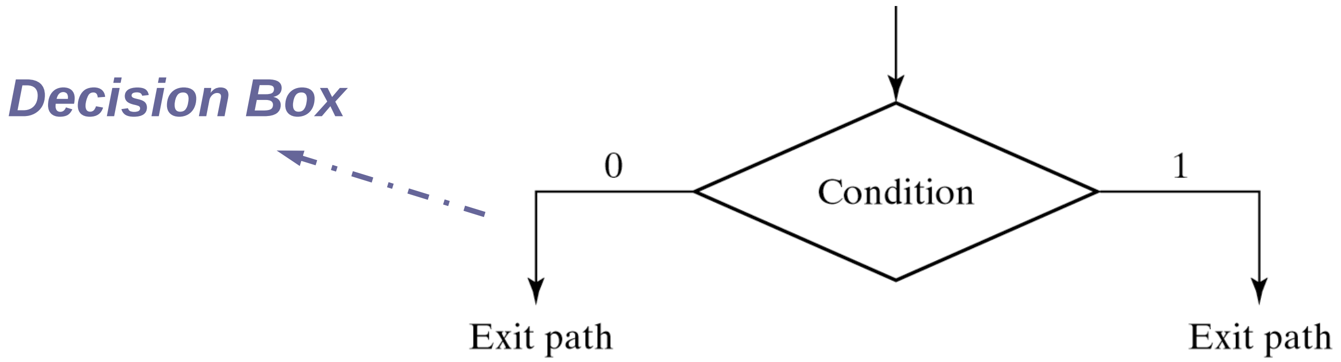
Algorithmic State Machine (ASM)

- Representation of a Finite State Machine that is suitable for digital systems with a larger number of inputs, outputs, and states compared to FSMs that are expressed using state diagrams and state tables

Elements Used in ASM Charts (1/2)

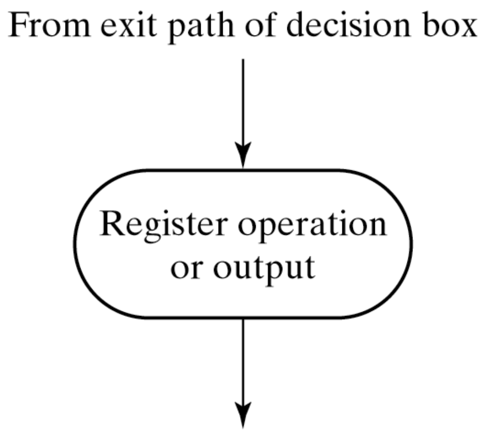


(a) General description (b) Specific example

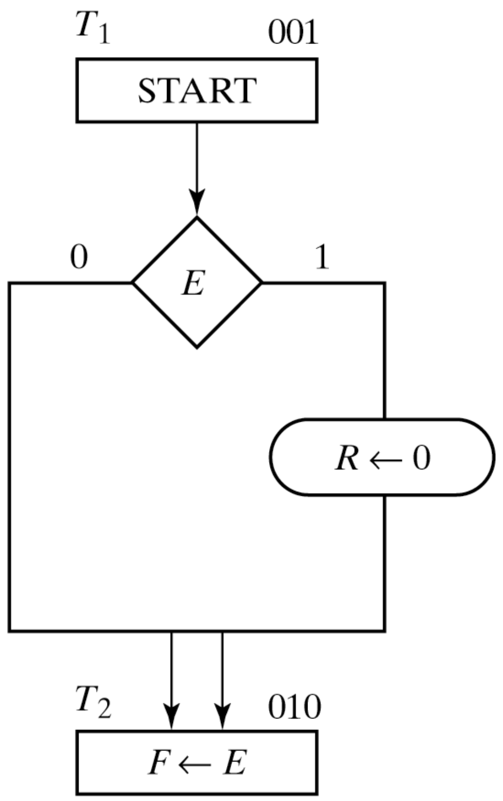


Elements Used in ASM Charts (2/2)

Conditional Box

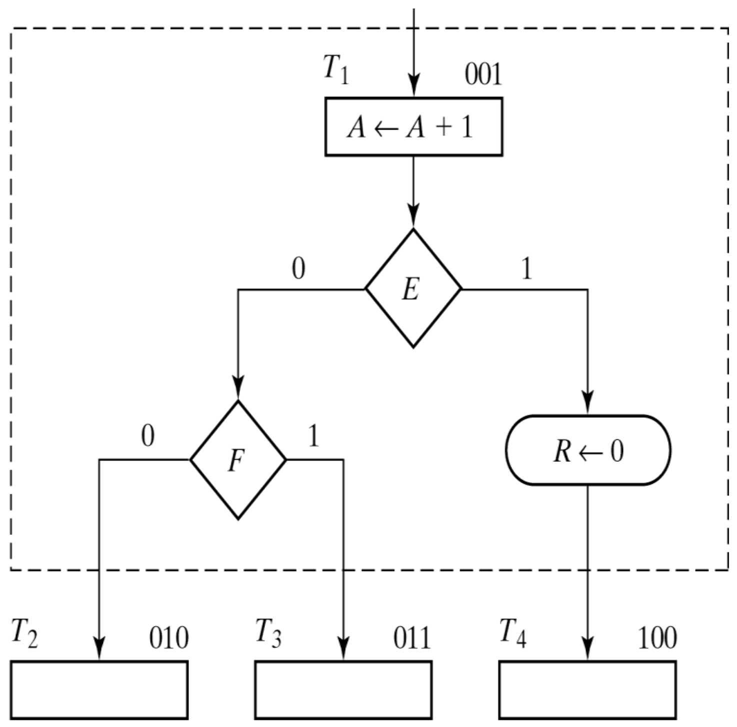


(a) General description

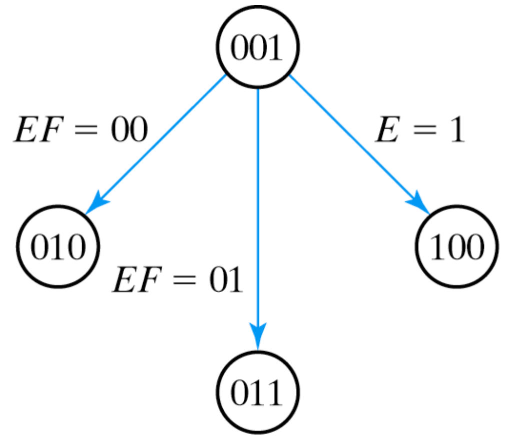


(b) Example with conditional box

ASM Block



ASM Block



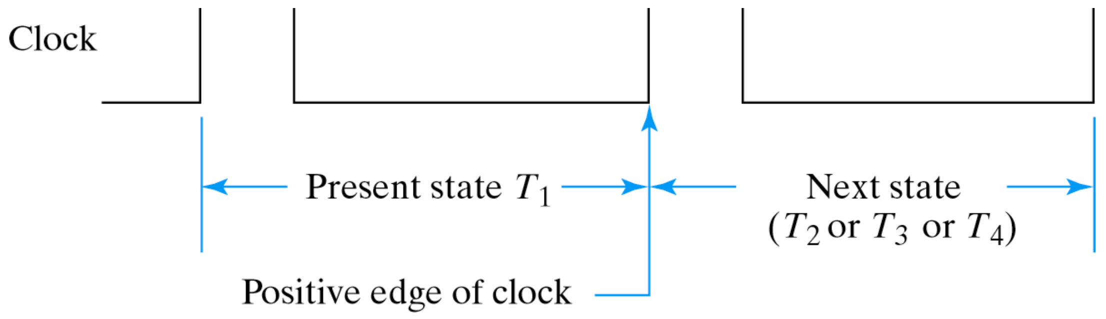
State Diagram Equivalent to the ASM Chart

Timing Considerations

- In the previous ASM block, the following operations occur in synchronism during the clock edge transition (simultaneously):
  - Register A is incremented
  - If E = 1, register R is cleared
  - Control transfers to the next state

Operations in datapath

Change in control logic

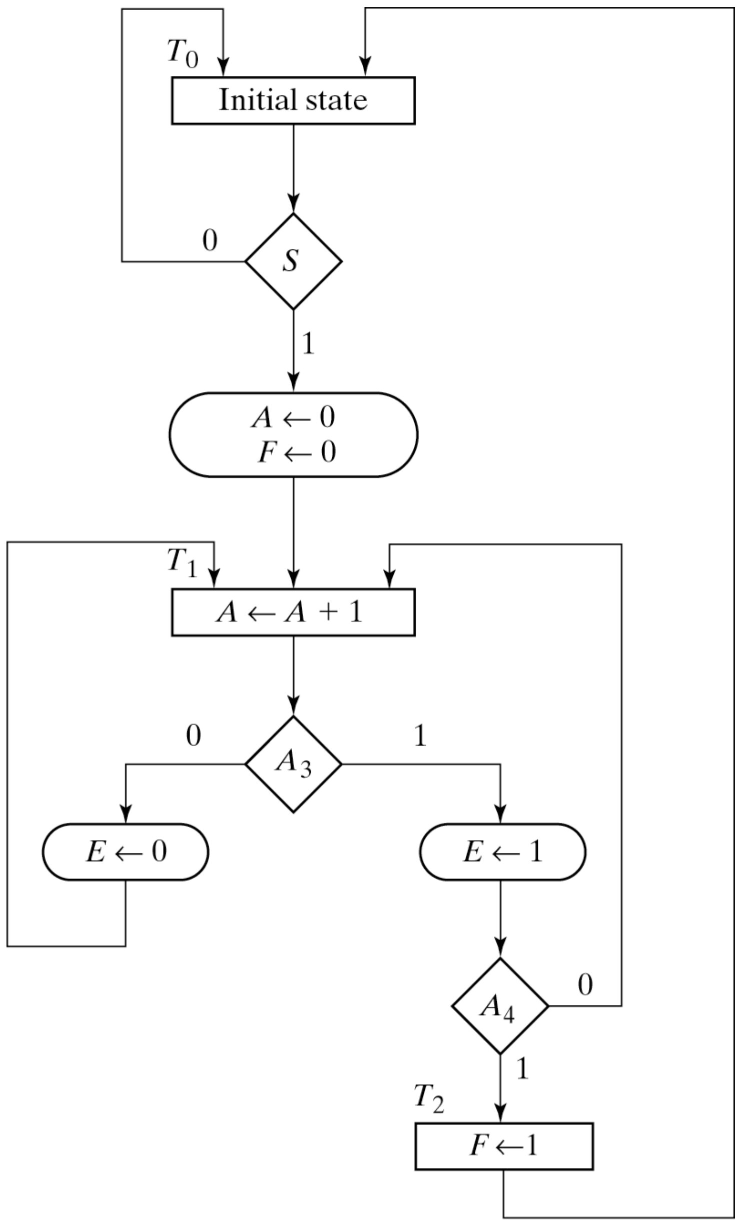


Transition Between States

Design Example

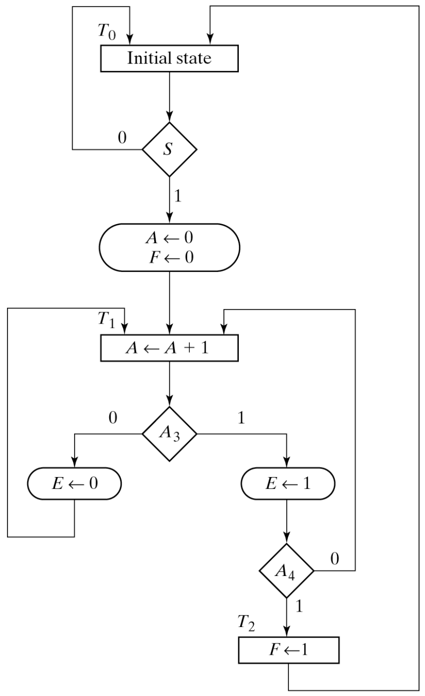
- Design a digital system with two flip-flops, E & F, and one 4-bit binary counter A ($A_4A_3A_2A_1$). A start signal S initiates system operation by clearing the counter A and flip-flop F. The counter is then incremented by one starting from the next clock pulse and continues to increment until the operations stop. Bits A_3 and A_4 determine the sequence of operations as follows:
  - If $A_3 = 0$, E is cleared and count continues
  - If $A_3 = 1$, E is set to 1; then if $A_4 = 0$, count continues, but if $A_4 = 1$, F is set to 1 on the next clock pulse and the system heads to initial state the clock pulse after
  - Then if S = 0, the system remains in the initial state, but if S = 1, the operation cycle repeats

ASM Chart



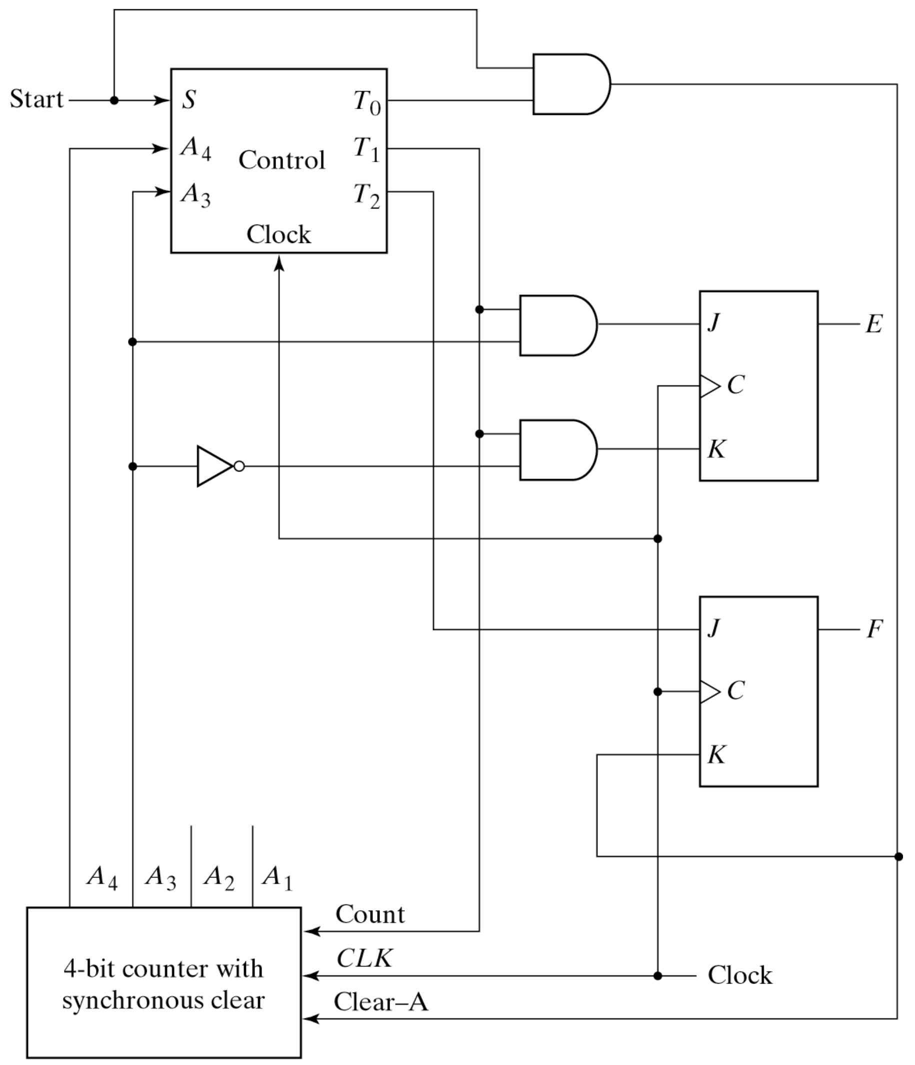
T_0
Initial state
S
0
1
$A \leftarrow 0$
$F \leftarrow 0$
T_1
$A \leftarrow A + 1$
A_3
0
1
$E \leftarrow 0$
$E \leftarrow 1$
A_4
0
1
T_2
$F \leftarrow 1$

Sequence of Operations

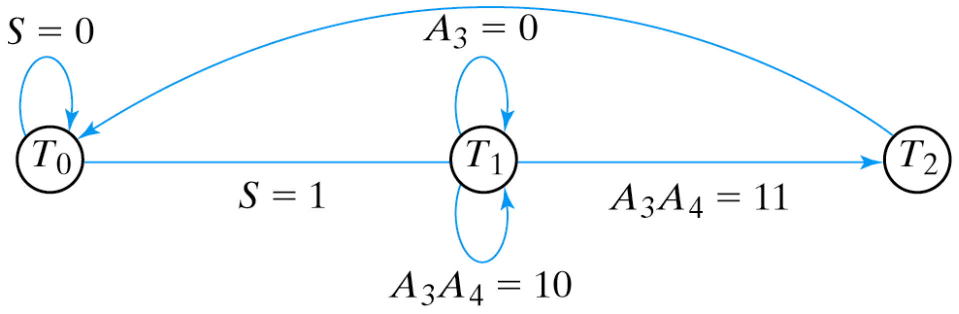


Counter				Flip-Flops			
A_4	A_3	A_2	A_1	E	F	**Conditions**	**State**
0	0	0	0	1	0	$A_3 = 0, A_4 = 0$	T_1
0	0	0	1	0	0		
0	0	1	0	0	0		
0	0	1	1	0	0		
0	1	0	0	0	0	$A_3 = 1, A_4 = 0$	
0	1	0	1	1	0		
0	1	1	0	1	0		
0	1	1	1	1	0		
1	0	0	0	1	0	$A_3 = 0, A_4 = 1$	
1	0	0	1	0	0		
1	0	1	0	0	0		
1	0	1	1	0	0		
1	1	0	0	0	0	$A_3 = 1, A_4 = 1$	
1	1	0	0	1	0		T_2
1	1	0	0	1	1		T_0

Datapath of Design



RTL Description



(a) State diagram for control

T_0: if ($S = 1$) then $A \leftarrow 0$, $F \leftarrow 0$

T_1: $A \leftarrow A + 1$

if ($A_3 = 1$) then $E \leftarrow 1$

if ($A_3 = 0$) then $E \leftarrow 0$

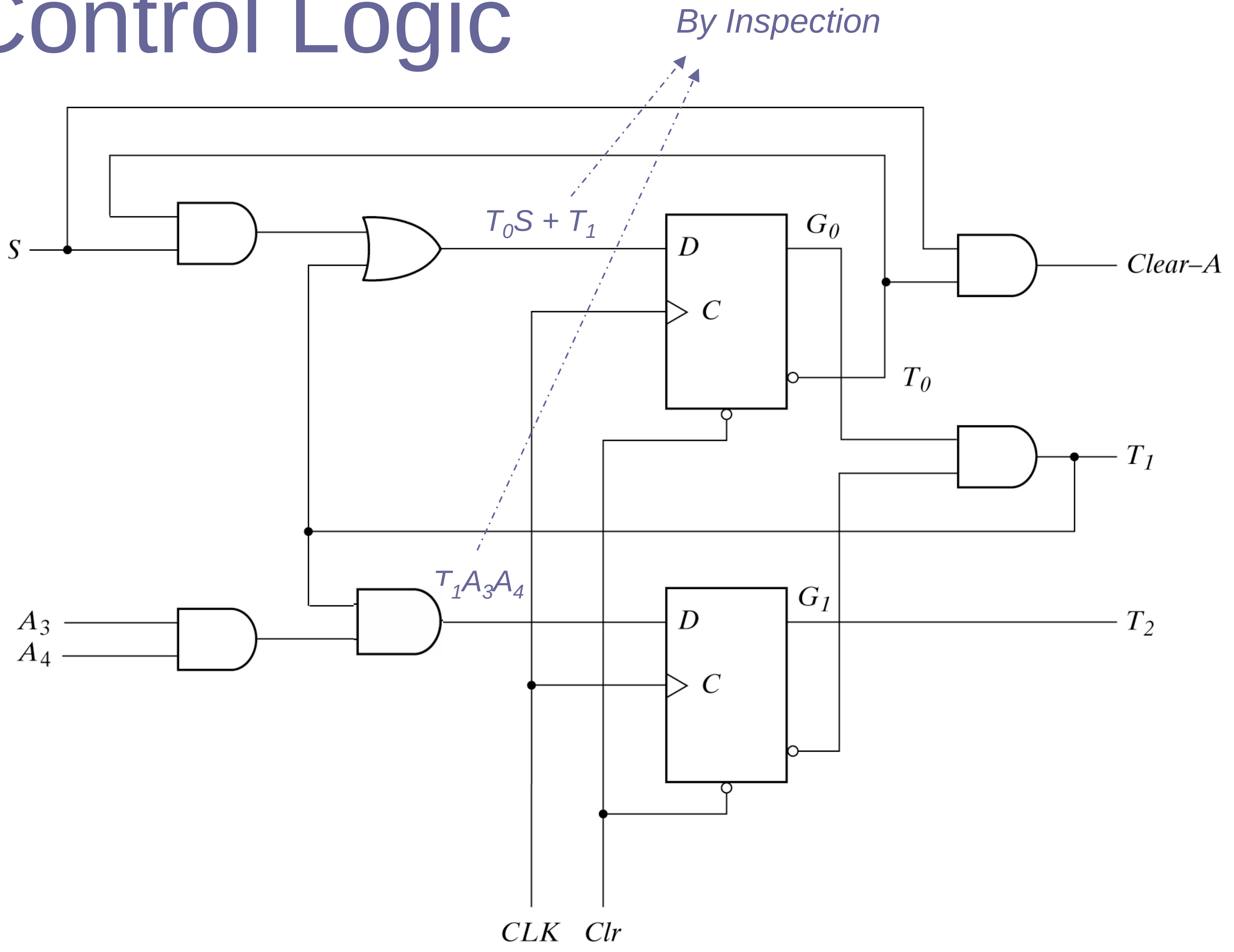
T_2: $F \leftarrow 1$

(a) Register transfer operations

State Table for Control

Present-State Symbol	Present State G_1	Present State G_0	Inputs S	Inputs A_3	Inputs A_4	Next State G_1	Next State G_0	Outputs T_0	Outputs T_1	Outputs T_2
T_0	0	0	0	X	X	0	0	1	0	0
T_0	0	0	1	X	X	0	1	1	0	0
T_1	0	1	X	0	X	0	1	0	1	0
T_1	0	1	X	1	0	0	1	0	1	0
T_1	0	1	X	1	1	1	1	0	1	0
T_2	1	1	X	X	X	0	0	0	0	1

Control Logic



References

- Digital Design, M. Morris, Mano