Outline

- Last time:
  - LSI Implementation: PLAs, PLDs, PALs
  - "PLA Format"
  - PLD Structures
  - PLA-Based FSM Design
  - Intro. to FPGAs
- This lecture:
  - Field-Programmable Logic Arrays
  - FPGA-Based FSM Design
  - Intro. to the Xilinx FPGA

FSM Design with FPGAs

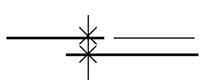
Actel Programmable Gate Arrays

Rows of programmable logic building blocks

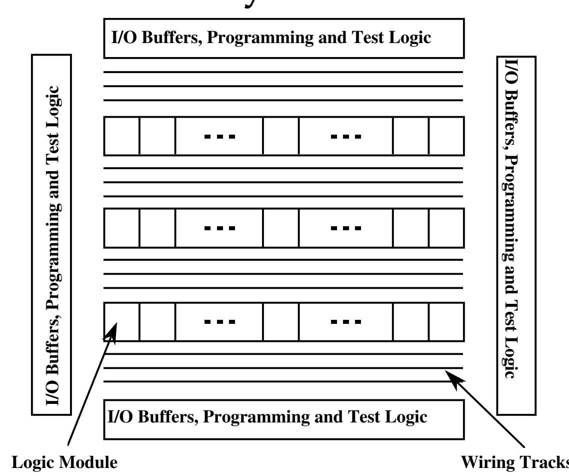
+

rows of interconnect

Anti-fuse Technology: Program Once



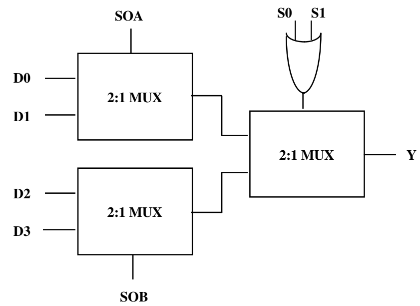
Use Anti-fuses to build up long wiring runs from short segments



8 input, single output combinational logic blocks
FFs constructed from discrete cross coupled gates

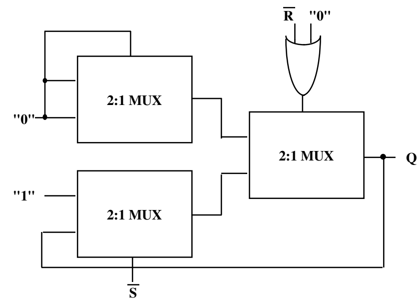
FSM Design with FPGAs

Actel Logic Module



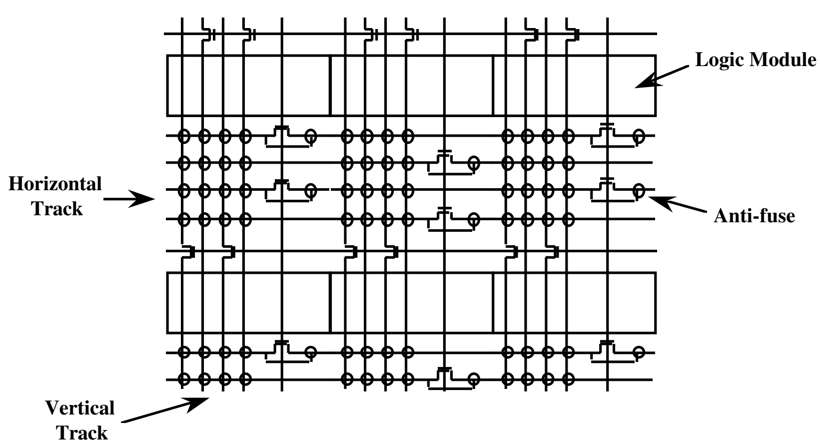
Basic Module is a Modified 4:1 Multiplexer

Example:
Implementation of S-R̄ L̄atch



FSM Design with FPGAs

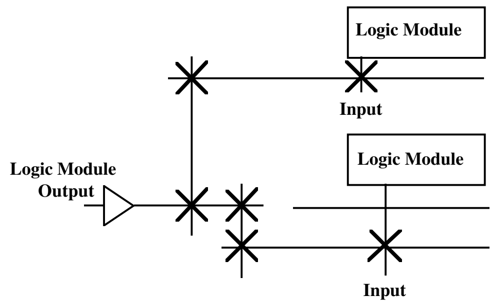
Actel Interconnect



Interconnection Fabric

FSM Design with FPGAs

Actel Routing Example



Jogs cross an anti-fuse

minimize the # of jobs for speed critical circuits

2 - 3 hops for most interconnections

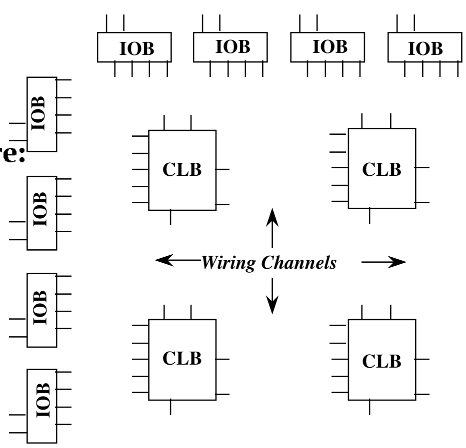
FSM Design with FPGAs

Xilinx Logic Cell Arrays

CMOS Static RAM Technology: programmable on the fly!
All personality elements connected into serial shift register
Shift in string of 1's and 0's on power up

General Chip Architecture:

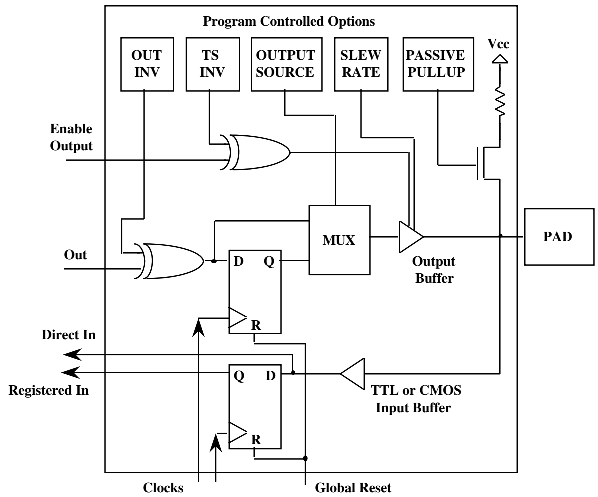
- **Logic Blocks (CLBs)**
- **IO Blocks (IOBs)**
- **Wiring Channels**



FSM Design with FPGAs

Xilinx LCA Architecture

- **Inputs:**
  Tri-state enable bit to output input, output clocks
- **Outputs:**
  input bit
- **Internal FFs for input & output paths**
- **Fast/Slow outputs**
  5 ns vs. 30 ns rise
- **Pull-up used with unused IOBs**

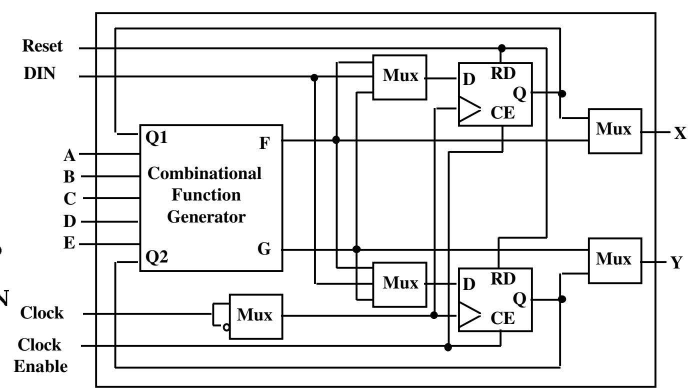


FSM Design with FPGAs

Xilinx LCA Architecture

Configurable Logic Block: CLB

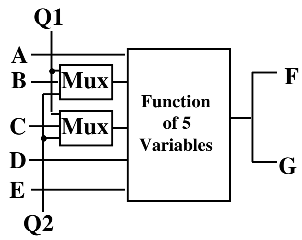
- **2 FFs**
- **Any function of 5 Variables**
- **Global Reset**
- **Clock, Clock Enb**
- **Independent DIN**



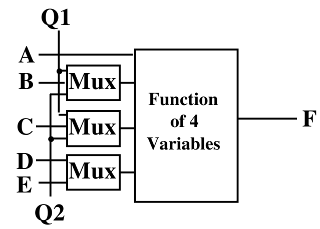
FSM Design with FPGAs

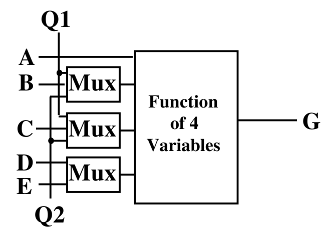
Xilinx LCA Architecture

CLB Function Generator



Any function of 5 variables



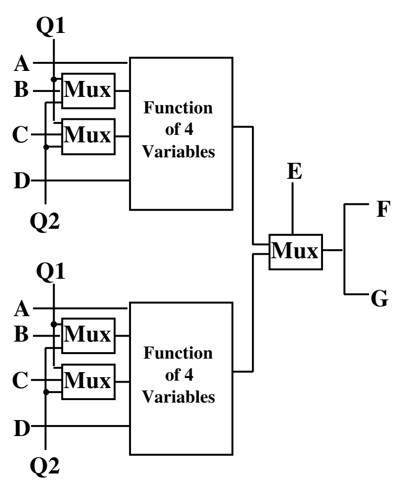


Two Independent Functions
of 4 variables each

FSM Design with FPGAs

Xilinx LCA Architecture

CLB Function Generator



Certain Limited
Functions of 6 Variables

FSM Design with FPGAs

Xilinx Application Examples

5-Input Parity Generator

Implemented with 1 CLB:

$F = \overline{A \text{ xor } B \text{ xor } C \text{ xor } D \text{ xor } E}$

2-bit Comparator: A B = C D or A B > C D

Implemented with 1 CLB:

(GT) $F = A\,\overline{C} + A\,B\,\overline{D} + B\,\overline{C}\,\overline{D}$

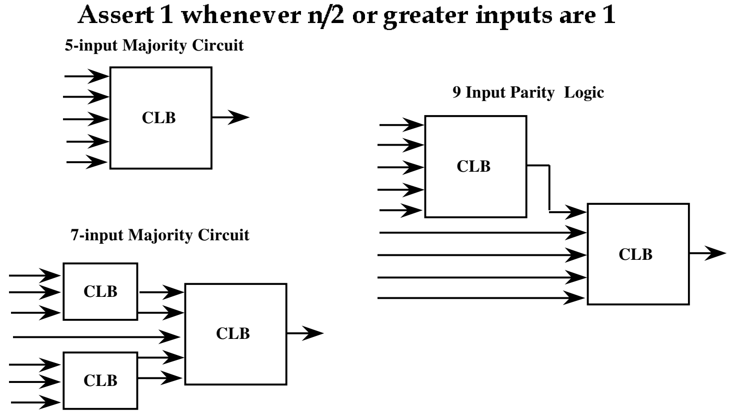
(EQ) $G = \overline{A}\,\overline{B}\,\overline{C}\,\overline{D} + \overline{A}\,B\,\overline{C}\,D + A\,B\,C\,D + A\,\overline{B}\,C\,\overline{D}$

FSM Design with FPGAs

Xilinx Application Examples

n-Input Majority Circuit

Assert 1 whenever n/2 or greater inputs are 1



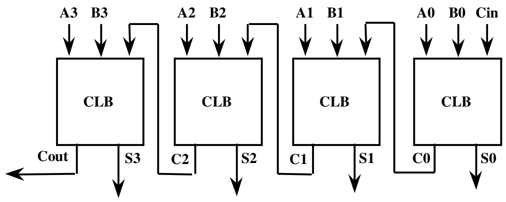
n-input Parity Functions

5 input = 1 CLB, 2 Levels of CLBs yield up to 25 inputs!

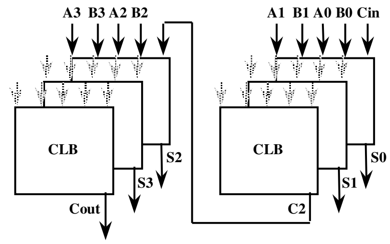
FSM Design with FPGAs

Xilinx Application Examples

4-bit Binary Adder



Full Adder, 4 CLB delays to final carry out



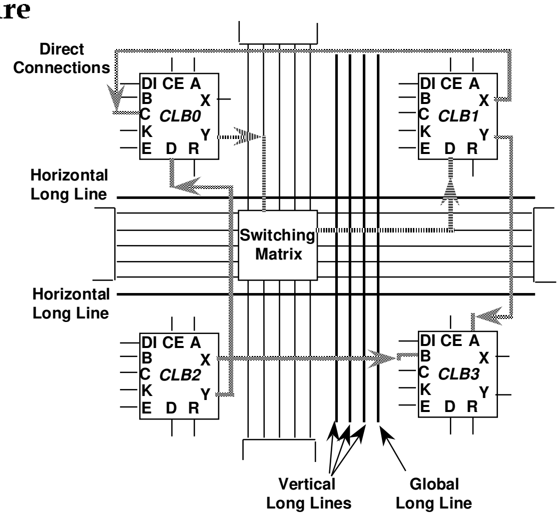
2 x Two-bit Adders (3 CLBs each) yields 2 CLBs to final carry out

FSM Design with FPGAs

Xilinx LCA Architecture

- Interconnect
- Direct Connections
- Global Long Line
- Horizontal/Vertical Long Lines

vSwitching Matrix Connections



FSM Design with FPGAs: Example

Xilinx LCA Architecture

Implementing the BCD to Excess 3 FSM

$Q2+ = \overline{Q2} \bullet Q0 \; + \; Q2 \bullet \overline{Q0}$

$Q1+ = \overline{X} \bullet \overline{Q2} \bullet \overline{Q1} \bullet Q0 \; + \; X \bullet \overline{Q2} \bullet \overline{Q0} \; + \; \overline{X} \bullet Q2 \bullet \overline{Q0} \; + \; Q1 \bullet \overline{Q0}$

$Q0+ = \overline{Q0}$

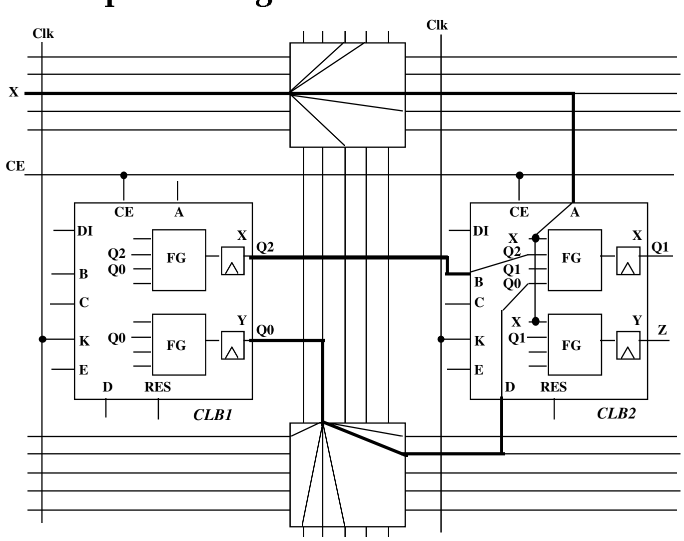
$Z = Z \bullet Q1 \; + \; \overline{X} \bullet \overline{Q1}$

- **No function more complex than 4 variables**
  4 FFs implies 2 CLBs
- **Synchronous Mealy Machine**
- **Global Reset to be used**
- **Place Q2+, Q0+ in once CLB**
  Q1, Z in second CLB
  maximize use of direct & general purpose interconnections

FSM Design with FPGAs: Example

Xilinx LCA Architecture

Implementing the BCD to Excess 3 FSM



FSM Design with Counters

Synchronous Counters: CLR, LD, CNT

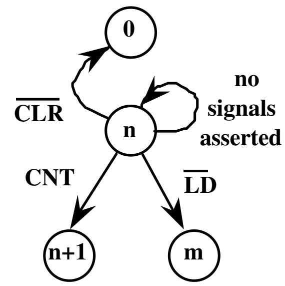
Four kinds of transitions for each state:

(1) to State 0 (CLR)

(2) to next state in sequence (CNT)

(3) to arbitrary next state (LD)

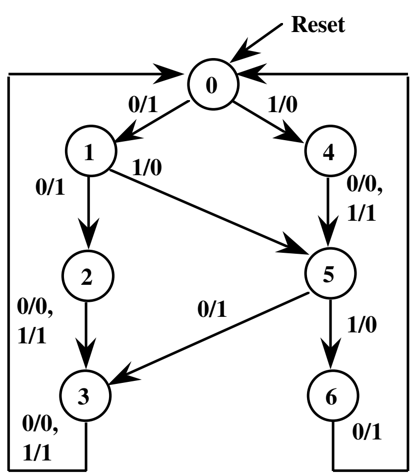
(4) loop in current state



Careful state assignment is needed to reflect basic sequencing of the counter

FSM Design with Counters: Example

Excess 3 Converter Revisited



Note the sequential nature of the state assignments

FSM Design with Counters

Excess 3 Converter

Inputs/Current State				*Next State*			*Outputs*						
X	Q2	Q1	Q0	Q2+	Q1+	Q0+	Z	$\overline{CLR}$	$\overline{LD}$	EN	C	B	A
0	0	0	0	0	0	1	1	1	1	1	X	X	X
0	0	0	1	0	1	0	1	1	1	1	X	X	X
0	0	1	0	0	1	1	0	1	1	1	X	X	X
0	0	1	1	0	0	0	0	0	X	X	X	X	X
0	1	0	0	1	0	1	1	1	1	1	X	X	X
0	1	0	1	0	1	1	0	1	0	X	0	1	0
0	1	1	0	0	0	0	1	0	X	X	X	X	X
0	1	1	1	X	X	X	X	X	X	X	X	X	X
1	0	0	0	1	0	0	0	1	0	X	1	0	0
1	0	0	1	1	0	1	0	1	0	X	1	0	1
1	0	1	0	0	1	1	1	1	1	1	X	X	X
1	0	1	1	0	0	0	1	0	X	X	X	X	X
1	1	0	0	1	0	1	0	1	1	1	X	X	X
1	1	0	1	1	1	0	1	1	1	1	X	X	X
1	1	1	0	X	X	X	X	X	X	X	X	X	X
1	1	1	1	X	X	X	X	X	X	X	X	X	X

CLR signal dominates LD which dominates Count

FSM Design with Counters

Excess 3 Converter

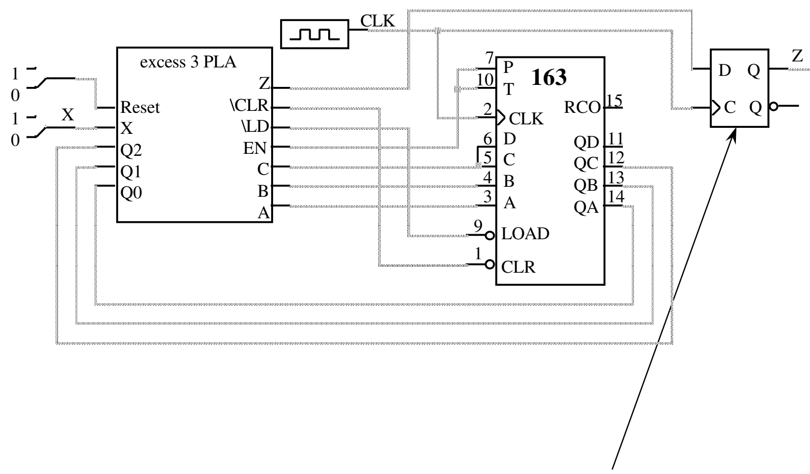
Espresso Input File

```
.i 5
.o 7
.ilb res x q2 q1 q0
.ob z clr ld en c b a
.p 17
1---- -0-----
00000 1111---
00001 1111---
00010 0111---
00011 00-----
00100 0111---
00101 110-011
00110 10-----
00111 -------
01000 010-100
01001 010-101
01010 1111---
01011 10-----
01100 1111---
01101 0111---
01110 -------
01111 -------
.e
```

Espresso Output File

```
.i 5
.o 7
.ilb res x q2 q1 q0
.ob z clr ld en c b a
.p 10
0-001 0101101
-0-01 1000000
-11-0 1000000
0-0-0 0101100
-000- 1010000
-0--0 0010000
0-10- 0101011
--11- 1000000
-11-- 0010000
-1-1- 1010000
.e
```

FSM Design with Counters



Synchronous Output Register

Design Case Study

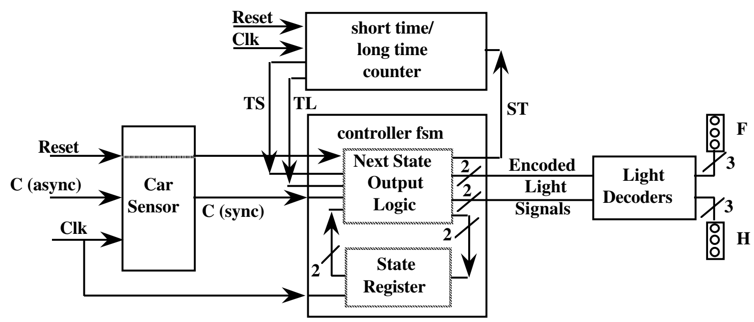
Traffic Light Controller

<u>**Decomposition into primitive subsystems**</u>

- **Controller FSM**
  next state/output functions
  state register
- **Short time/long time interval counter**
- **Car Sensor**
- **Output Decoders and Traffic Lights**

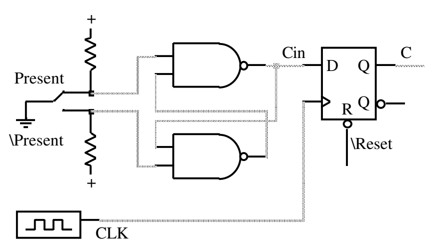
Design Case Study

Block Diagram



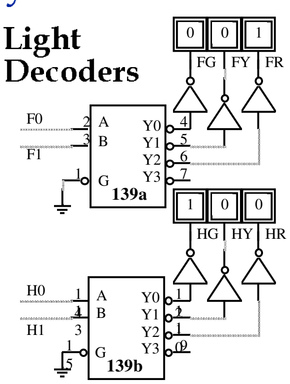
Design Case Study

Subsystem Logic

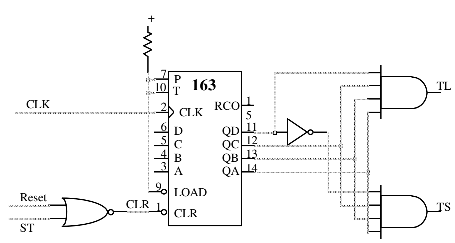


Car Detector

Light Decoders



Interval Timer



Design Case Study

Next State Logic

State Assignment: HG = 00, HY = 10, FG = 01, FY = 11

$P1 = C\ TL\ \overline{Q1} + \overline{TS}\ Q1\ \overline{Q0} + \overline{C}\ \overline{Q1}\ Q0 + \overline{TS}\ Q1\ Q0$

$P0 = TS\ Q1\ \overline{Q0} + \overline{Q1}\ Q0 + \overline{TS}\ Q1\ Q0$

$ST = C\ TL\ \overline{Q1} + \overline{C}\ \overline{Q1}\ Q0 + TS\ Q1\ \overline{Q0} + TS\ Q1\ Q0$

$HL[1] = TS\ Q1\ Q0 + \overline{Q1}\ Q0 + \overline{TS}\ Q1\ Q0$

$HL[0] = \overline{TS}\ Q1\ \overline{Q0} + TS\ Q1\ \overline{Q0}$

$FL[1] = \overline{Q0}$

$FL[0] = TS\ Q1\ Q0 + \overline{TS}\ Q1\ Q0$

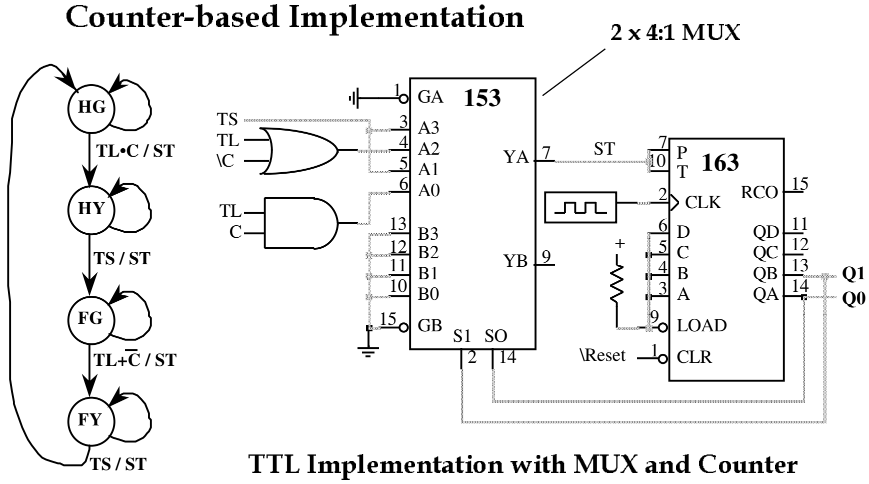
PAL/PLA Implementation:
- **5 inputs, 7 outputs, 8 product terms**
- **PAL 22V10 -- 11 inputs, 10 prog. IOs, 8 to 14 prod terms per OR**

ROM Implementation:
- **32 word by 8-bit ROM (256 bits)**
- **Reset may double ROM size**

Design Case Study

Next State Logic

Counter-based Implementation



TTL Implementation with MUX and Counter

ST = Count

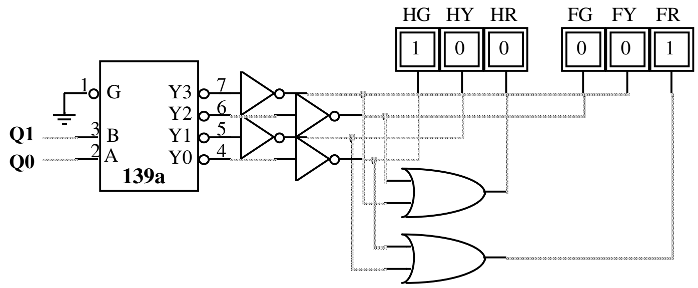
Can we reduce package count by using an 8:1 MUX?

Design Case Study

Next State Logic

Counter-based Implementation

Dispense with direct output functions for the traffic lights
Why not simply decode from the current state?



ST is a Synchronous Mealy Output

Light Controllers are Moore Outputs

Design Case Study

LCA-Based Implementation

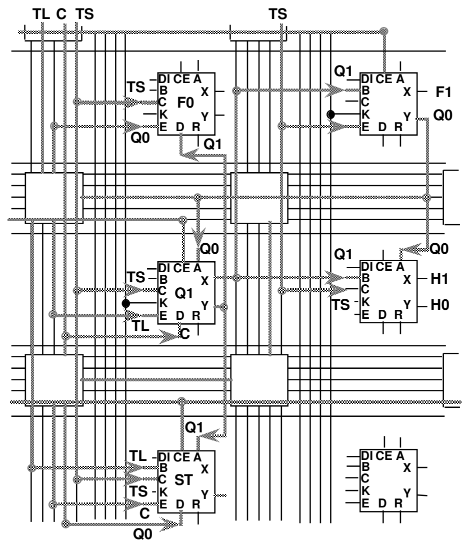
Discrete Gate Method:

- **None of the functions exceed 5 variables**
- **P1, ST are 5 variable (1 CLB each)**
- **P0, HL1, HL0, FL0 are 3 variable (1/2 CLB each)**
- **FL1 is 1 variable (1/2 CLB)**
- **4 1/2 CLBs total!**

Design Case Study

LCA-Based Implementation

Placement of functions selected to maximize the use of direct connections



Design Case Study

LCA-Based Implementation

Counter/Multiplexer Method:

- 4:1 MUX, 2 Bit Upcounter
- MUX: six variables (4 data, 2 control)
  >> but this is the kind of 6 variable function that can be implemented in 1 CLB!
  >> 2nd CLB to implement TL • C and TL + C'
  >> But note that ST/Cnt is really a function of TL, C, TS, Q1, Q0
  >> 1 CLB to implement this function of 5 variables!
- 2 Bit Counter: 2 functions of 3 variables (2 bit state + count)
  >> Also implemented in one CLB
- Traffic light decoders: functions of 2 variables (Q1, Q0)
  >> 2 per CLB = 3 CLB for the six lights
- Total count = 5 CLBs