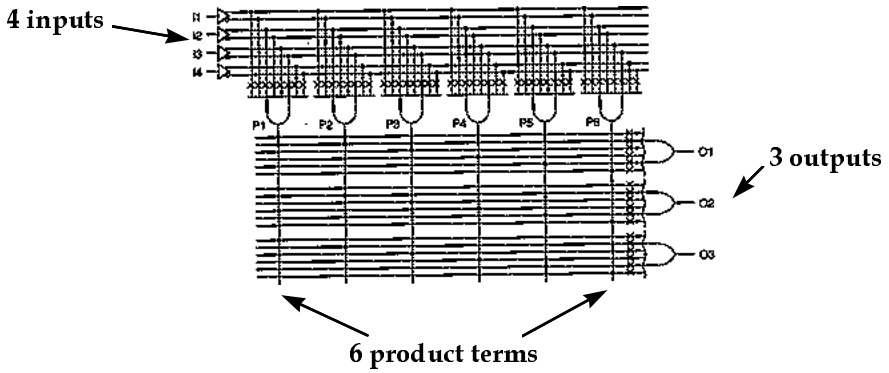
Outline

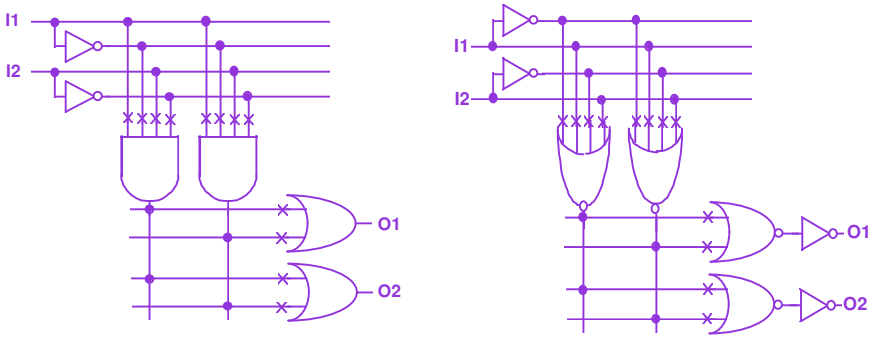
- Last time:
  - ROM, RAM
  - Bussing, Tri-state, multi-porting
  - Switch De-bouncing
  - FSM Design With ROMs
- This lecture:
  - LSI Implementation: PLAs, PLDs, PALs
  - "PLA Format"
  - PLD Structures
  - PLA-Based FSM Design
  - Intro. to FPGAs

Programmable Logic Arrays (PLAs)

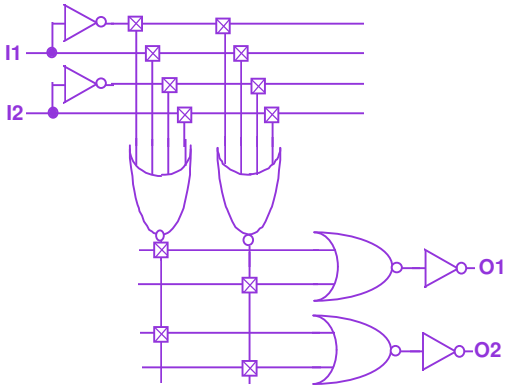
- We know that any logic function can be implemented in Truth-Table form, so can we build a general circuit structure that can implement a *simplified* ROM form directly?



AND-OR and NOR-NOR PLAs

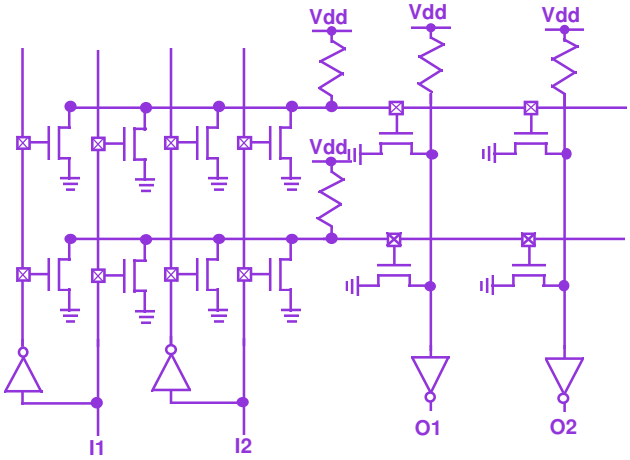


Alternate Programming Approach



- Connections between horizontal and vertical lines on integrated circuit make by inserting a "via" or "cut" in the SiO2 between the mask layers.

Transistor-Level Implementation

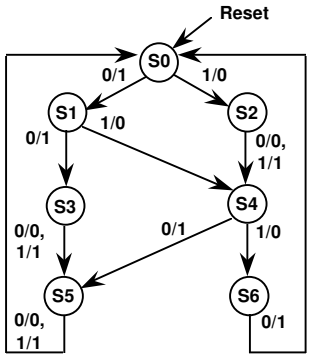


Design Example: Seven-Segment Decoder

PLA-Based FSM Design: Example

Present State	Next State X=0	Next State X=1	Output X=0	Output X=1
S0	S1	S2	1	0
S1	S3	S4	1	0
S2	S4	S4	0	1
S3	S5	S5	0	1
S4	S5	S6	1	0
S5	S0	S0	0	1
S6	S0	--	1	--

State Transition Table



Derived State Diagram

PLA-Based FSM Design: Example

State Assignment with NOVA

```
0 S0 S1 1
1 S0 S2 0
0 S1 S3 1
1 S1 S4 0
0 S2 S4 0
1 S2 S4 1
0 S3 S5 0
1 S3 S5 1
0 S4 S5 1
1 S4 S6 0
0 S5 S0 0
1 S5 S0 1
0 S6 S0 1
```

NOVA input file

S0 = 000
S1 = 001
S2 = 011
S3 = 110
S4 = 100
S5 = 111
S6 = 101

NOVA derived state assignment

9 product term implementation

PLA-Based FSM Design: Example

```
.i 4
.o 4
.ilb x q2 q1 q0
.ob d2 d1 d0 z
.p 16
0 000 001 1
1 000 011 0
0 001 110 1
1 001 100 0
0 011 100 0
1 011 100 1
0 110 111 0
1 110 111 1
0 100 111 1
1 100 101 0
0 111 000 0
1 111 000 1
0 101 000 1
1 101 - - - -
0 010 - - - -
1 010 - - - -
.e
```

Espresso Inputs

```
.i 4
.o 4
.ilb x q2 q1 q0
.ob d2 d1 d0 z
.p 9
0001      0100
10-0      0100
01-0      0100
1-1-      0001
-0-1      1000
0-0-      0001
-1-0      1000
- -10     0100
- - -0    0010
.e
```

"-" means "don't connect"

Espresso Outputs

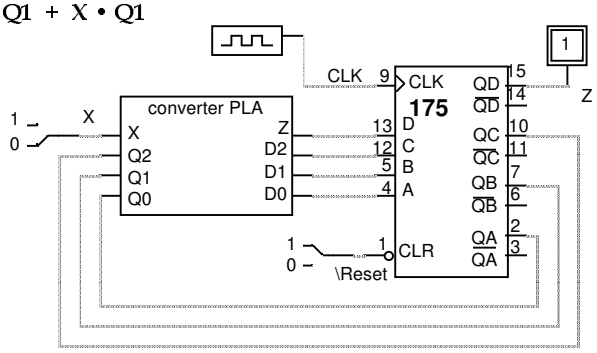
PLA-Based FSM Design: Example

$D2 = \overline{Q2} \cdot Q0 + Q2 \cdot \overline{Q0}$

$D1 = \overline{X} \cdot \overline{Q2} \cdot \overline{Q1} \cdot Q0 + X \cdot \overline{Q2} \cdot \overline{Q0} + \overline{X} \cdot Q2 \cdot \overline{Q0} + Q1 \cdot \overline{Q0}$

$D0 = \overline{Q0}$

$Z = X \cdot Q1 + \overline{X} \cdot \overline{Q1}$



Remember: Good tools cannot guarantee a good design, but if you undertstand their limitations, they can reduce the time spent on the less creative aspects of design.

PAL-Based FSM Design: Example

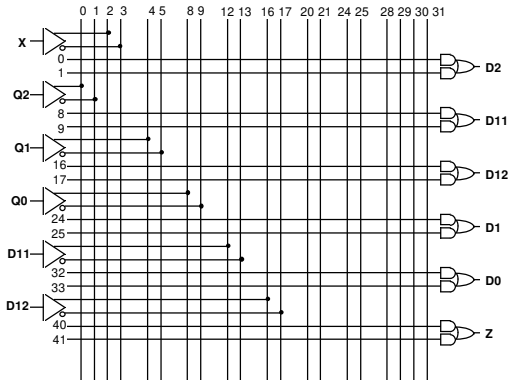
10H8 PAL: 10 inputs, 8 outputs, 2 product terms per OR gate

D1 = D11 + D12

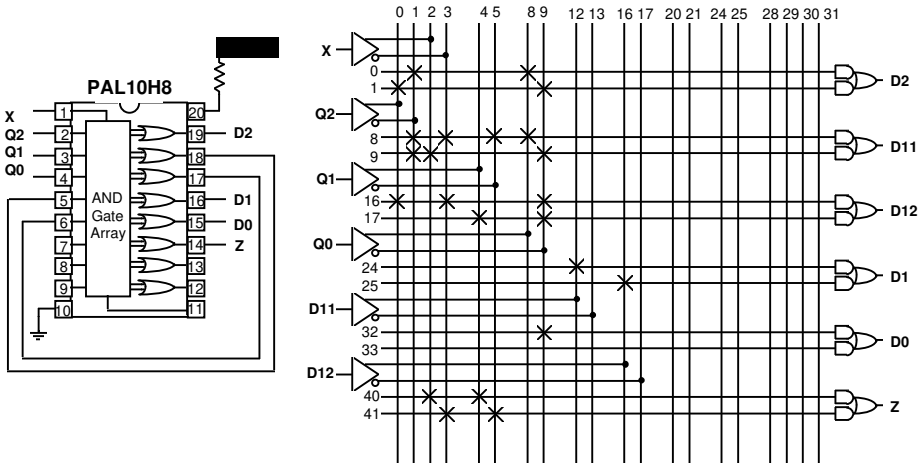
D11 = $\overline{X}$ • $\overline{Q2}$ • $\overline{Q1}$ • Q0 + X • $\overline{Q2}$ • $\overline{Q0}$

D12 = $\overline{X}$ • Q2 • $\overline{Q0}$ + Q1 • $\overline{Q0}$

0. Q2' • Q0
1. Q2 • Q0'
8. X' • Q2' • Q1' • Q0
9. X • Q2' • Q0'
16. X' • Q2 • Q0'
17. Q1 • Q0'
24. D11
25. D12
32. Q0'
33. not used
40. X • Q1
41. X '• Q1'

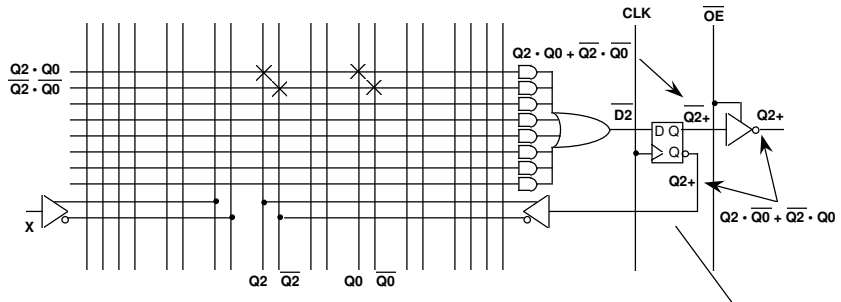


PAL-Based FSM Design: Example



Advanced PAL-Based FSM Design: Example

Registered PAL Architecture



$\overline{D2} = Q2 \bullet Q0 + \overline{Q2} \bullet \overline{Q0}$

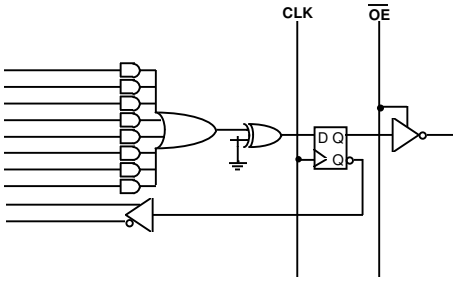
$\overline{D1} = \overline{X} \bullet \overline{Q2} \bullet \overline{Q1} \bullet \overline{Q0} + X \bullet Q2 + X \bullet Q0 + Q2 \bullet Q0 + Q1 \bullet Q0$

$\overline{D0} = Q0$

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

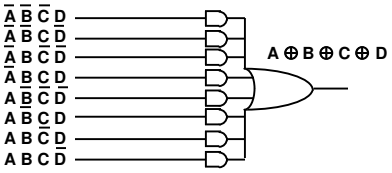
PAL-Based FSM Design: Example

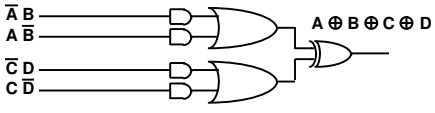
Programmable Output Polarity/XOR PALs



Buried Registers: decouple FF from the output pin

Advantage of XOR PALs: Parity and Arithmetic Operations





PAL-Based FSM Design

Specifying PALs with ABEL

P10H8 PAL

```
module bcd2excess3
title 'BCD to Excess 3 Code Converter State Machine'
u1 device 'p10h8';

"Input Pins
 X,Q2,Q1,Q0,D11i,D12i  pin  1,2,3,4,5,6;

"Output Pins
 D2,D11o,D12o,D1,D0,Z  pin  19,18,17,16,15,14;

INSTATE = [Q2, Q1, Q0];
S0 = [0, 0, 0];
S1 = [0, 0, 1];
S2 = [0, 1, 1];
S3 = [1, 1, 0];
S4 = [1, 0, 0];
S5 = [1, 1, 1];
S6 = [1, 0, 1];

equations
 D2 = (!Q2 & Q0) # (Q2 & !Q0);
 D1 = D11i # D12i;
 D11o = (!X & !Q2 & !Q1 & Q0) # (X & !Q2 & !Q0);
 D12o = (!X & Q2 & !Q0) # (Q1 & !Q0);
 D0 = !Q0;
 Z = (X & Q1) # (!X & !Q1);

end bcd2excess3;
```

Explicit equations for partitioned output functions

PAL-Based FSM Design

Specifying PALs with ABEL

P12H6 PAL

```
module bcd2excess3
title 'BCD to Excess 3 Code Converter State Machine'
u1 device  'p12h6';

"Input Pins
 X, Q2, Q1, Q0  pin  1, 2, 3, 4;

"Output Pins
 D2, D1, D0, Z  pin  17, 18, 16, 15;

INSTATE = [Q2, Q1, Q0]; OUTSTATE = [D2, D1, D0];
S0in = [0, 0, 0];  S0out = [0, 0, 0];
S1in = [0, 0, 1];  S1out = [0, 0, 1];
S2in = [0, 1, 1];  S2out = [0, 1, 1];
S3in = [1, 1, 0];  S3out = [1, 1, 0];
S4in = [1, 0, 0];  S4out = [1, 0, 0];
S5in = [1, 1, 1];  S5out = [1, 1, 1];
S6in = [1, 0, 1];  S6out = [1, 0, 1];

equations
 D2 = (!Q2 & Q0) # (Q2 & !Q0);
 D1 = (!X & !Q2 & !Q1 & Q0) # (X & !Q2 & !Q0) #
  (!X & Q2 & !Q0) # (Q1 & !Q0);
 D0 = !Q0;
 Z = (X & Q1) # (!X & !Q1);

end bcd2excess3;
```

Simpler equations

PAL-Based FSM Design

Specifying PALs with ABEL

P16R4 PAL

```
module bcd2excess3
title 'BCD to Excess 3 Code Converter'
u1 device 'p16r4';

"Input Pins
 Clk, Reset, X, !OE     pin  1, 2, 3, 11;

"Output Pins
 D2, D1, D0, Z     pin 14, 15, 16, 13;

SREG = [D2, D1, D0];
S0 = [0, 0, 0];
S1 = [0, 0, 1];
S2 = [0, 1, 1];
S3 = [1, 1, 0];
S4 = [1, 0, 0];
S5 = [1, 1, 1];
S6 = [1, 0, 1];
```

```
state_diagram SREG
state S0: if Reset then S0
    else if X then S2 with Z = 0
    else S1 with Z = 1
state S1: if Reset then S0
    else if X then S4 with Z = 0
    else S3 with Z = 1
state S2: if Reset then S0
    else if X then S4 with Z = 1
    else S4 with Z = 0
state S3: if Reset then S0
    else if X then S5 with Z = 1
    else S5 with Z = 0
state S4: if Reset then S0
    else if X then S6 with Z = 0
    else S5 with Z = 1
state S5: if Reset then S0
    else if X then S0 with Z = 1
    else S0 with Z = 0
state S6: if Reset then S0
    else if !X then S0 with Z = 1

end bcd2excess3;
```


FSM Design with More Sophisticated PLDs

Programmable Logic Devices = PLD

PALs, PLAs = 10 - 1,000s Gate Equivalents

Field Programmable Gate Arrays = FPGAs

- Altera MAX Family
- Actel Programmable Gate Array
- Xilinx Logical Cell Array

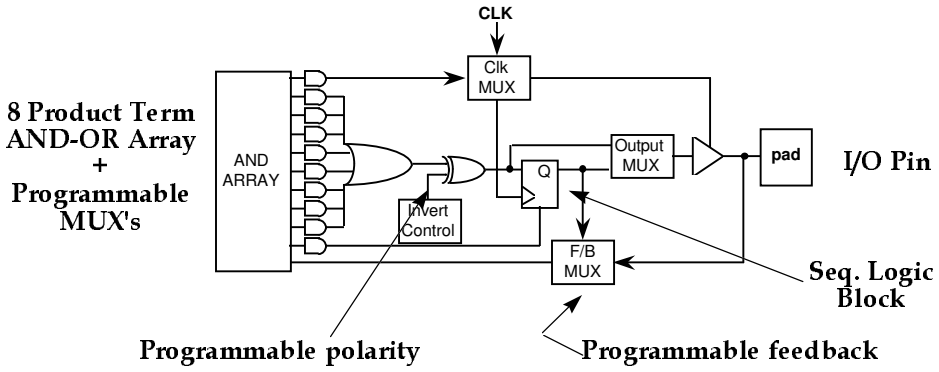
1,000s - 10,000s of Gate Equivalents!

FSM Design with More Sophisticated PLDs

Altera EPLD (Erasable Programmable Logic Devices)

Historical Perspective:
PALs – same technology as programmed once bipolar PROM
EPLDs — CMOS erasable programmable ROM (EPROM)
erased by UV light

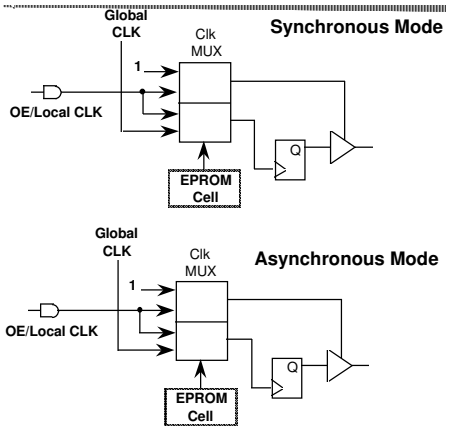
Altera building block = MACROCELL



FSM Design with More Sophisticated PLDs

Altera EPLDs contain 8 to 48 independently programmed macrocells

Personalized by EPROM bits:



Flipflop controlled by global clock signal

local signal computes output enable

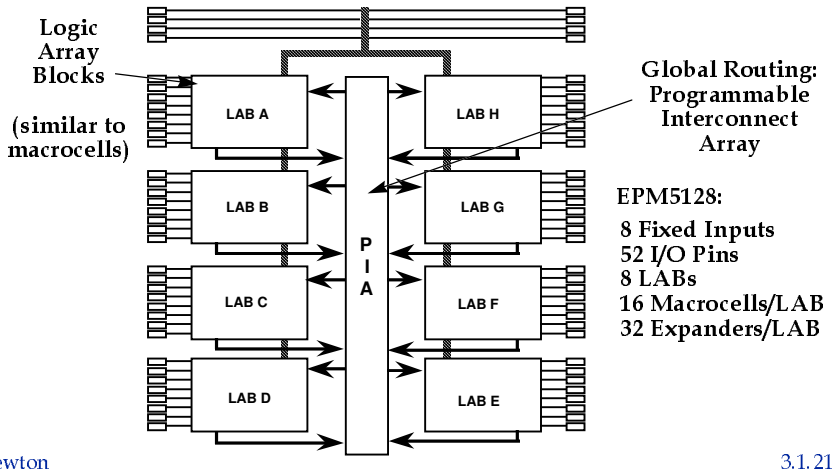
Flipflop controlled by locally generated clock signal

- Seq Logic: could be D, T positive or negative edge triggered
- product term to implement clear function

FSM Design with More Sophisticated PLDs

AND-OR structures are relatively limited
Cannot share signals/product terms among macrocells

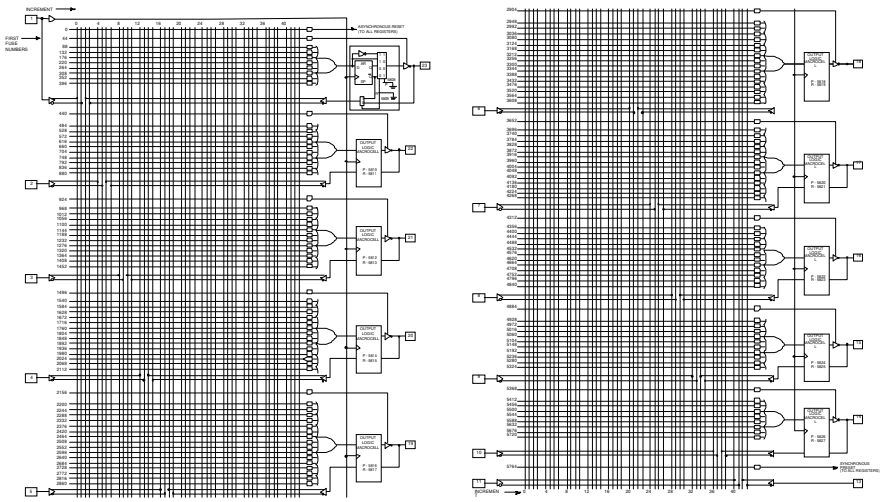
Altera solution: Multiple Array Matrix (MAX)



EPM5128:
8 Fixed Inputs
52 I/O Pins
8 LABs
16 Macrocells/LAB
32 Expanders/LAB

FSM Design with More Sophisticated PALs

P22V10 PAL



Supports large number of product terms per output
Latches and muxes associated with output pins