Introduction

• Idea: Memory as Programmable Logic

<table>
<thead>
<tr>
<th>X</th>
<th>A1</th>
<th>00</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>01</td>
<td>01</td>
<td>1</td>
</tr>
<tr>
<td>Y</td>
<td>A0</td>
<td>10</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>11</td>
<td>11</td>
<td>0</td>
</tr>
</tbody>
</table>

Address lines as inputs
Data line as output
Truth table is the content

\[ X \oplus Y \]
PROM

**PROM: Programmable Read Only Memory**

- Combinational Circuit: Program Truth Table
- Fixed AND, Programmable OR
- $2^n$ Minterms (n-input AND gates)
- Sharing of Minterms by outputs
- Large Area because of $2^n$ AND gates
- Can we reduce AND gates?
  - Yes, But should be programmable
  - Then Minterms → Product Terms (PLA)
PLA: Programmable Logic Array

- Programmable AND
- Programmable OR
- $< 2^n$ Product Terms (2n-input AND gates)
- Sharing of Product Terms by outputs
- Programming Overhead
- For a single output, programmable OR is not required, if one can disable product terms
PAL: Programmable Array Logic

- Programmable AND
- Fixed OR
- $< 2^n$ Product Terms (2n-input AND gates)
- Dedicated Product Terms for outputs
Evolution

- PROM: Programmable Read Only Memory
  Fixed AND, Programmable OR

- PLA: Programmable Logic Array
  Programmable AND, Programmable OR

- PAL: Programmable Array Logic
  Programmable AND, Fixed OR

PAL16L8

Source: Texas Instruments Data sheets
### Input/Output structure

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Output</strong></td>
<td></td>
</tr>
<tr>
<td>Enable Tri-state</td>
<td></td>
</tr>
<tr>
<td>Blow all input</td>
<td>connections of the control AND gate, wired AND with pull up</td>
</tr>
<tr>
<td><strong>Input</strong></td>
<td></td>
</tr>
<tr>
<td>Disable Tri-state</td>
<td></td>
</tr>
<tr>
<td>Retain all connections</td>
<td>of the control AND gate</td>
</tr>
<tr>
<td><strong>Input/Output</strong></td>
<td></td>
</tr>
<tr>
<td>Program Control AND</td>
<td>gate with control product term</td>
</tr>
<tr>
<td>Disable any AND gate</td>
<td>Retain all input connections</td>
</tr>
<tr>
<td>Cascade</td>
<td>More than 7 Product Terms</td>
</tr>
<tr>
<td>Feedback</td>
<td>Latch</td>
</tr>
</tbody>
</table>

**Source:** Texas Instruments Data sheets
Cascading

- 17 Product terms

Cascading / 17 PT

Source: Texas Instruments Data sheets
Cascading / 17 PT

\[
\begin{align*}
7 + 6 + 4 & : \\
& - 3 \text{ passes} \\
& - 3 \text{ section delays} \\
& - 8 \text{ Sections (} 7 + 6 \times 7 = 49 \text{ PTs), 8 passes} \\
\end{align*}
\]

\[
\begin{align*}
(7 + 7) + 3 & : \\
& - 2 \text{ passes} \\
& - 2 \text{ section delays} \\
& - 8 \text{ sections (} 7 \times 7 = 49 \text{ PT), 2 passes} \\
\end{align*}
\]

Source: Texas Instruments Data sheets
Simple PLD’s

- **Devices**
  - PAL16L8
  - PAL16R4
  - PAL16R8
  - PAL22V10

- **Manufacturers**
  - Atmel

- **Feature**
  - Wide Decoding
  - E.g. AND Gate with 16 inputs, and 16 complements
  - Earlier used for Chip select decoding of memory and peripherals
  - Higher order address bits were decoded

Present Day Scenario

- Present Day SoC’s have
- Built in RAM, Peripherals
- Built in Configurable Chip Select decoding
- Less use of SPLD’s like PAL16L8
Present Day Scenario

- Serial Interface for external peripherals
- Clock, data
- Multiple Slaves
- Address part of Data Frames
- Read / write
- Multiple Masters, Arbitration
- SPI, I2C

Kuruvilla Varghese

PAL16R4

Source: Texas Instruments Data sheets
PAL16R4

Datapath
Datapath => PAL

Source: Texas Instruments Data sheets

FSM

Source: Texas Instruments Data sheets
FSM => PAL

Source: Texas Instruments Data sheets

FSM

Kuruvilla Varghese
**Substitution**

- **Substitution**
  - \( x = ab/ + cd \)
    - 2 PT’s 1 pass delay
  - \( y = x + ef + ghi/ \)
    - 3 PT’s 2 pass delay

- **Substitute x in y**
  - \( y = ab/ + cd + ef + ghi/ \)
    - 4 PTs 1 Pass delay

- **Virtual Substitution**
  - Uses unused PTs
  - Reduces delay

Source: Texas Instruments Data sheets
Substitution

- Pathological case: 
  \[ a \text{xor} \ b = \frac{ab}{2} + \frac{a}{b} \]
  2 PTs

- \[ a \text{xor} \ b \text{xor} \ c \]
  4 PTs

- \[ a \text{xor} \ b \ldots \text{xor} \ n \]
  \(2^{n-1}\) PTs

- Attribute to turn of virtual substitution

- Priority Encoders
- Adders/Subtractors
PAL22V10 - Macrocell

Source: Texas Instruments Data sheets
PAL22V10

- Variable Product Terms
- Asynchronous Reset
  Product Term
- Synchronous Preset
  Product term
- Combinational / Registered
  output
- Product Term Optimization
  by Inversion

- PT Optimization
  - \( Y = A/C + AB + BC + AC/ \)
  - \( Y/ = AB/C + A/C/ \)

- Timing
  - \( t_{pd} \)
  - \( t_{en}, t_{dis} \)
  - \( t_{cq}, t_{s}, t_{n}, t_{res} \)
  - With / without feedback

---

PLD 22V10 Fitting

- Is it possible to implement an 8 bit odd parity generator in a PLD 22V10? i.e. parity generator has 8 data inputs and one parity output
- One has to check the following I/O requirements and product term requirements.
- No: of inputs = 8 < 12 dedicated inputs of 22V10
- No: of outputs = 1 < 10 I/Os of 22V10
- Ex-or of n variables results in \( 2^{n-1} \) product terms.
- We assume tool expands the EXOR’s to complete product terms than implementing with Internal nodes.

---

Kuruvilla Varghese
PLD 22V10 Fitting

- For 8 data inputs odd parity generator, Number of product terms are 128 (We need to compute the product terms as PLD22V10 doesn’t have XOR gates)
- Now PLD 22V10 has \((2 \times 16 + 2 \times 14 + 2 \times 12 + 2 \times 10 + 2 \times 8) = 120\) product terms among 10 outputs.
- For cascading we need a Macro cell/section with at least 9 product terms, so we need to use a macro cell with 10 sections, this leaves us with 110 product terms.
- Hence, the product term resources aren’t enough.

---

PLD 22V10 Fitting

- This assumes that cascaded terms are substituted. If there is no product term substitution, it is possible to fit, if any of the following options are chosen
  - cascade of 7 ex-or gates with 6 internal nodes
  - log structure of 2 input exor gates 9
  - log structure of 3 input exor gates
  - log structure of 2, 4 input exor gates with second stage of 1, 2 input exor gates.
  - different cascaded schemes in which fewer than 7 exor gates are substituted.
Applications of SPLD

- Glue Logic
- Counter
- FSM
- Wide decoding is not required for many applications
- Less FFs

Programming Technology

- Fuse
- EPROM (UV Erasable)
- EEPROM
- Flash
EPROM Transistor

Control Gate (Wordline) — Interpoly Dielectric
Oxide (=150 Å) — Floating Gate (Storage element)
Source — n+ — Drain (Bitline)
Substrate (p-type)

PAL16L8

Source: Texas Instruments Data sheets
EPROM Transistor Wired-AND

- **Programmable AND**
  - Wired NOR with inputs using complements
  - Transistors are EPROM
  - When programmed, No connection

- **Fixed OR**
  - Wired NOR followed by Inverter
  - Normal n-type transistors are used
Flash / EEPROM Transistor

Control Gate (Wordline)  
Interpoly Dielectric  
Floating Gate (Storage element)  
Tunnel Oxide (~100 Å)  
Source  
Substrate (p-type)  
Drain (Bitline)  
n+  
n+

Flash Cell Write

Control Gate (Wordline)  
Floating Gate (Storage element)  
Source  
Substrate (p-type)  
Drain (Bitline)  
n+  
n+  
Inversion Region  
$V_D = V_T$  
$V_D = 6V$
Flash Cell Erase

Control Gate (Wordline)

Floating Gate (Storage element)

Source

n+

Drain (Bitline)

Substrate (p-type)

Vg = Vp

V0 = No Connection

EEPROM/Flash Transistor Wired-AND

Wired NOR, Invert the inputs

I1 (I1)

I2 (I2)

In (In)

Wired OR

Kuruvilla Varghese
Programming SPLD’s

- Programming Interface
  - JEDEC File, standard, ASCII
  - Fuse patterns in 0 & 1
- Programming methodology
  - Proprietary, uses the normal pins
  - High voltage - switch to programming mode
  - Address, data, read/write, program pins
  - Blank check, verification, security
  - UV/Electrically Erasable, Programmer

Complex PLD (CPLD)

- Not a big SPLD, as the SPLD already has wide product terms.
- What is sensible is multiple SPLD’s interconnected, such that blocks of a medium sized design can fit in to these SPLD blocks.
- Interconnection requirements.
  - Output of any block should be able to go to one or more inputs of any other blocks.
  - Any input signal should be able to go to one or more inputs of any blocks.
Complex PLD (CPLD)

- Hierarchical PLD
- Product term array
- Product term Allocator / Distributor
- Macrocells
- I/O cells
- Programmable Interconnect

PAL22V10

Source: Texas Instruments Data sheets
CPLD Manufacturers

- **Xilinx**
  - XC9500XL, CoolRunner-II
- **Altera**
  - Max 3000A, Max 7000S, Max II, Max V
- **Atmel**
  - ATF15xx
- **Lattice Semiconductor**
  - ispMACH 4000ZE

MAX7000 CPLD

Source: Altera Data sheets
PIA: Programmable Interconnect Array

- Cross bar switch satisfying the connectivity requirements discussed before.
- N x N cross bar can be implemented using, N, N-to-1 Multiplexers.
- Interconnection between blocks using just one switch.
- Simple timing, fast.
- Cross bar don’t scale well

2 x 2 Crossbar

- N x N crossbar requires N, N to 1 Multiplexers
MAX7000 Features

<table>
<thead>
<tr>
<th>Feature</th>
<th>EPM7032</th>
<th>EPM7064</th>
<th>EPM7066</th>
<th>EPM7128E</th>
<th>EPM7160E</th>
<th>EPM7192E</th>
<th>EPM7256E</th>
</tr>
</thead>
<tbody>
<tr>
<td>Usable gates</td>
<td>600</td>
<td>1,250</td>
<td>1,800</td>
<td>2,500</td>
<td>3,200</td>
<td>3,750</td>
<td>5,000</td>
</tr>
<tr>
<td>Macrocells</td>
<td>32</td>
<td>64</td>
<td>96</td>
<td>128</td>
<td>160</td>
<td>192</td>
<td>256</td>
</tr>
<tr>
<td>Logic array blocks</td>
<td>2</td>
<td>4</td>
<td>6</td>
<td>8</td>
<td>10</td>
<td>12</td>
<td>16</td>
</tr>
<tr>
<td>Maximum user I/O pins</td>
<td>36</td>
<td>68</td>
<td>76</td>
<td>100</td>
<td>104</td>
<td>124</td>
<td>164</td>
</tr>
<tr>
<td>t_{EQ} (ns)</td>
<td>6</td>
<td>6</td>
<td>7.5</td>
<td>7.5</td>
<td>10</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>t_{SU} (ns)</td>
<td>5</td>
<td>5</td>
<td>6</td>
<td>6</td>
<td>7</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>t_{SU} (ns)</td>
<td>2.5</td>
<td>2.5</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>t_{DFF} (ns)</td>
<td>4</td>
<td>4</td>
<td>4.6</td>
<td>4.5</td>
<td>5</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>f_{MAX} (MHz)</td>
<td>151.5</td>
<td>151.5</td>
<td>125.0</td>
<td>125.0</td>
<td>100.0</td>
<td>90.9</td>
<td>90.9</td>
</tr>
</tbody>
</table>

Source: Altera Data sheets

---

Xilinx XC9500 Crossbar (Fast CONNECT)
**MAX7000 Macrocell**

![MAX7000 Macrocell Diagram]

Source: Altera Data sheets

Kuruvilla Varghese

**Clock Enable**

![Clock Enable Diagram]

Kuruvilla Varghese
Logic Block Features

- 5 PT’s per Macrocell
- D/T Flip-flop. Flip flop can be bypassed for combinational output.
- XOR Gate: Polarity control, PT optimization, Comparator, Arithmetic circuits, Parity
- PT set, PT Reset,
- Global Clock, PT Clock
- PT Clock enable

MAX7000S Macrocell Fast Input

Source: Altera Data sheets
Fast Input

- Fast input architecture allows direct connection of input to the Macrocell flip-flop through 2 to 1 Mux at the input of flip-flop
MAX7000S Macrocell Fast Input

Fitting: VHDL Code

```vhdl
process (clk, rst)
begin
if (rst = '1') then q <= '0';
elsesif (clk'event and clk = '1') then
    if (en = '1') then
        q <= a xor b;
    end if;
end if;
end if;
end process;
```
Fitting: Synthesized Circuit

\[ \text{clk} \rightarrow q \rightarrow q \]

\[ \text{a, b, en, clk, rst} \]

Fitting in CPLD

\[ \text{rst, clk} \rightarrow \text{q} \rightarrow \text{to I/O Control Blisk} \]

\[ \text{b, a, en} \]

Source: Altera Data sheets
Product Term Allocator

- Demultiplexers let the PT’s to be allocated for various functions like XOR input, PT clock, PT clock enable, PT Set, PT Reset and PT Steering.
- 5 input OR gate with associated steering circuit allows the unused PT’s in one section (combined with ones from section above/below) to be steered to OR gate of the section above or below.
MAX7000S I/O Control Block

Source: Altera Data sheets

MAX7000 Timing Model

Source: Altera Data sheets
**CPLD vs FPGA**

<table>
<thead>
<tr>
<th>Features</th>
<th>PLD</th>
<th>FPGA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logic</td>
<td>AND-OR</td>
<td>Mux / LUT / Gates</td>
</tr>
<tr>
<td>Register to Logic ratio</td>
<td>Small</td>
<td>Large</td>
</tr>
<tr>
<td>Timing</td>
<td>Simple</td>
<td>Complex</td>
</tr>
<tr>
<td>Architecture Variation</td>
<td>Small</td>
<td>Large</td>
</tr>
<tr>
<td>Programming Technology</td>
<td>Flash</td>
<td>Anti-Fuse, SRAM</td>
</tr>
<tr>
<td>Capacity</td>
<td>10 K</td>
<td>Few Million Logic Cells + Few MB RAM</td>
</tr>
</tbody>
</table>

**CPLD Applications**

- Small design comprising of counters, FSM, Small logic
- Examples
  - Memory controllers (DRAM controller)
  - Bus protocol translation (CPU Bus -> PCI Bus)
  - Optical encoders.
  - Small control circuit in Instrumentation, Power Electronics for Data Acquisition control, Small digital control circuits.
CPLD Applications

- Design which requires lot of registers and memory and complex designs cannot be implemented.
  - Signal processing architectures (Filters)
  - Complex arithmetic circuits
  - Communication circuits (Packet processing, Modems)
  - CODEC’s
  - Cryptographic circuits