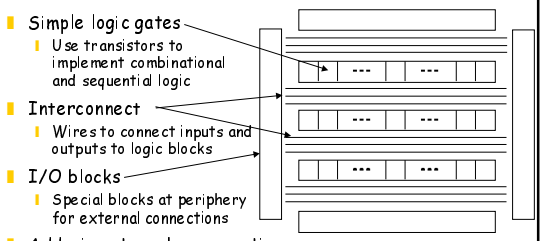
Evolution of Implementation Technologies

- Discrete devices: relays, transistors (1940s-50s)
- Discrete logic gates (1950s-60s)
- Integrated circuits (1960s-70s)
  - e.g. TTL packages: Data Book for 100's of different parts
  - Map your circuit to the Data Book parts
- Gate Arrays (IBM 1970s)
  - "Custom" integrated circuit chips
  - Design using a library (like TTL)
  - Transistors are already on the chip
  - Place and route software puts the chip together automatically
  - + Large circuits on a chip
  - + Automatic design tools (no tedious custom layout)
  - - Only good if you want 1000's of parts

trend toward higher levels of integration

Gate Array Technology (IBM - 1970s)

- Simple logic gates
  - Use transistors to implement combinational and sequential logic
- Interconnect
  - Wires to connect inputs and outputs to logic blocks
- I/O blocks
  - Special blocks at periphery for external connections
- Add wires to make connections
  - Done when chip is fabbed
    - "mask-programmable"
  - Construct any circuit



Programmable Logic

- Disadvantages of the Data Book method
  - Constrained to parts in the Data Book
  - Parts are necessarily small and standard
  - Need to stock many different parts
- Programmable logic
  - Use a single chip (or a small number of chips)
  - Program it for the circuit you want
  - No reason for the circuit to be small

Programmable Logic Technologies

- Fuse and anti-fuse
  - Fuse makes or breaks link between two wires
  - Typical connections are 50-300 ohm
  - One-time programmable (testing before programming?)
  - Very high density
- EPROM and EEPROM
  - High power consumption
  - Typical connections are 2K-4K ohm
  - Fairly high density
- RAM-based
  - Memory bit controls a switch that connects/disconnects two wires
  - Typical connections are .5K-1K ohm
  - Can be programmed and re-programmed *in the circuit*
  - Low density

Programmable Logic

- Program a connection
  - Connect two wires
  - Set a bit to 0 or 1
- Regular structures for two-level logic (1960s-70s)
  - All rely on two-level logic minimization
  - PROM connections - permanent
  - EPROM connections - erase with UV light
  - EEPROM connections - erase electrically
  - PROMs
    - Program connections in the _____ plane
  - PLAs
    - Program the connections in the _____ plane
  - PALs
    - Program the connections in the _____ plane

Making Large Programmable Logic Circuits

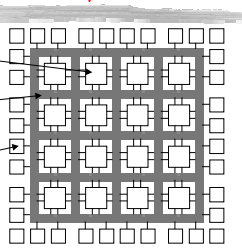
- Alternative 1 : "CPLD"
  - Put a lot of PLDS on a chip
  - Add wires between them whose connections can be programmed
  - Use fuse/EEPROM technology
- Alternative 2: "FPGA"
  - Emulate gate array technology
  - Hence *Field Programmable Gate Array*
  - You need:
    - A way to implement logic gates
    - A way to connect them together

Field-Programmable Gate Arrays

- PALs, PLAs = 10 - 100 Gate Equivalents
- Field Programmable Gate Arrays = FPGAs
  - Altera MAX Family
  - Actel Programmable Gate Array
  - Xilinx Logical Cell Array
- 100 - 1000(s) of Gate Equivalents!

Field-Programmable Gate Arrays

- Logic blocks
  - To implement combinational and sequential logic
- Interconnect
  - Wires to connect inputs and outputs to logic blocks
- I/O blocks
  - Special logic blocks at periphery of device for external connections



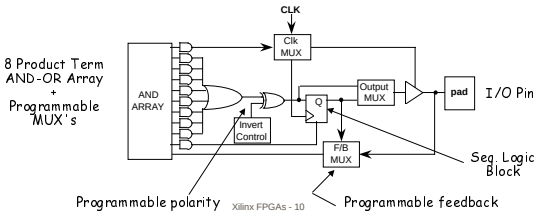
- Key questions:
  - How to make logic blocks programmable?
  - How to connect the wires?
  - *After the chip has been fabbed*

Tradeoffs in FPGAs

- Logic block - how are functions implemented: fixed functions (manipulate inputs) or programmable?
  - Support complex functions, need fewer blocks, but they are bigger so less of them on chip
  - Support simple functions, need more blocks, but they are smaller so more of them on chip
- Interconnect
  - How are logic blocks arranged?
  - How many wires will be needed between them?
  - Are wires evenly distributed across chip?
  - Programmability slows wires down - are some wires specialized to long distances?
  - How many inputs/outputs must be routed to/from each logic block?
  - What utilization are we willing to accept? 50%? 20%? 90%?

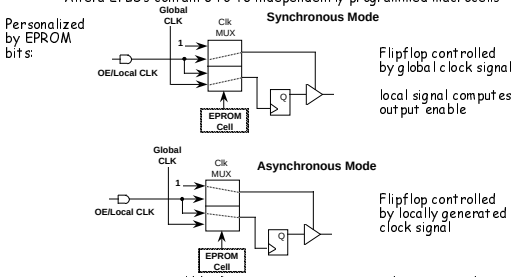
Altera EPLD (Erasable Programmable Logic Devices)

- Historical Perspective
  - PALs: same technology as programmed once bipolar PROM
  - EPLDs: CMOS erasable programmable ROM (EPROM) erased by UV light
- Altera building block = MACROCELL



Altera EPLD

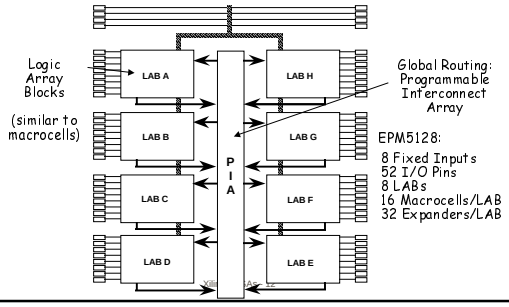
Altera EPLDs contain 8 to 48 independently programmed macrocells



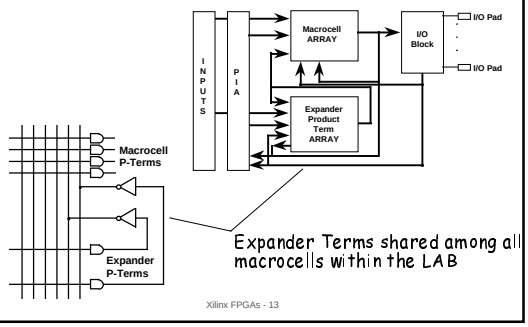
+ Seq Logic: could be D, T positive or negative edge triggered
+ product term to implement clear function

Altera Multiple Array Matrix (MAX)

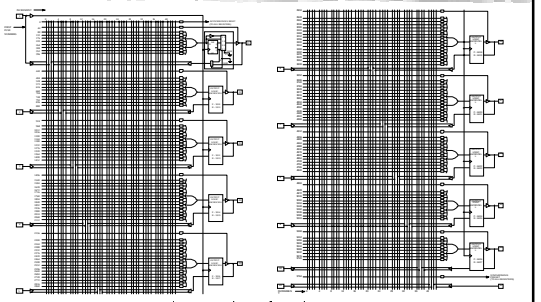
AND-OR structures are relatively limited
Cannot share signals/product terms among macrocells



LAB Architecture



P22V10 PAL



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

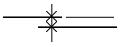
Actel Programmable Gate Arrays

Rows of programmable logic building blocks

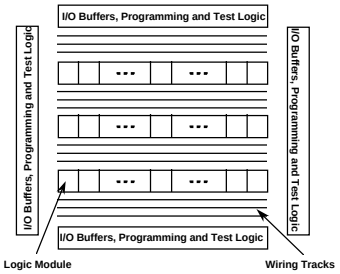
+

rows of interconnect

Anti-fuse Technology: Program Once



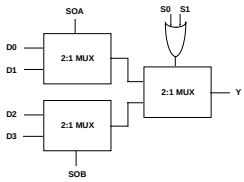
Use Anti-fuses to build up long wiring runs from short segments



8 input, single output combinational logic blocks

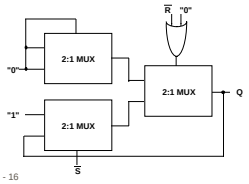
FFs constructed from discrete cross coupled gates

Actel Logic Module

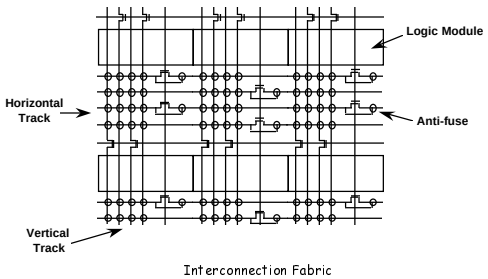


Basic Module is a Modified 4:1 Multiplexer

Example: Implementation of S-R Latch

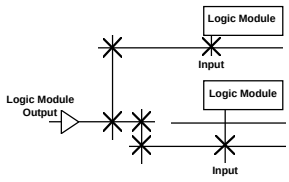


Actel Interconnect



Interconnection Fabric

Actel Routing Example



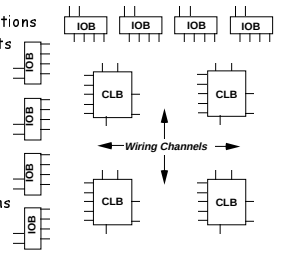
Jogs cross an anti-fuse

minimize the # of jobs for speed critical circuits

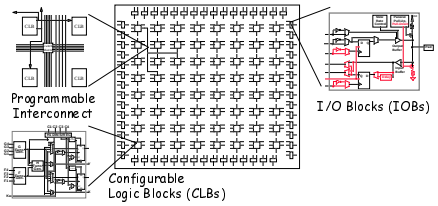
2 - 3 hops for most interconnections

Xilinx Programmable Gate Arrays

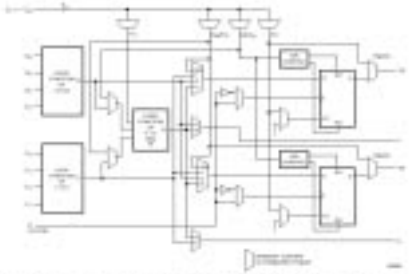
- CLB - Configurable Logic Block
  - 5-input, 1 output function
  - or 2 4-input, 1 output functions
  - optional register on outputs
- Built-in fast carry logic
- Can be used as memory
- Three types of routing
  - direct
  - general-purpose
  - long lines of various lengths
- RAM-programmable
  - can be reconfigured



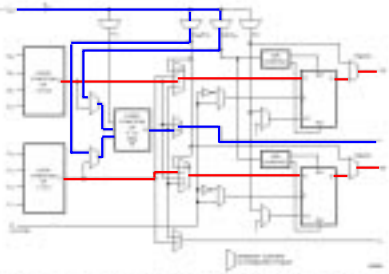
XILINX®



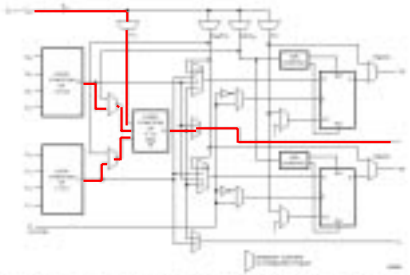
The Xilinx 4000 CLB



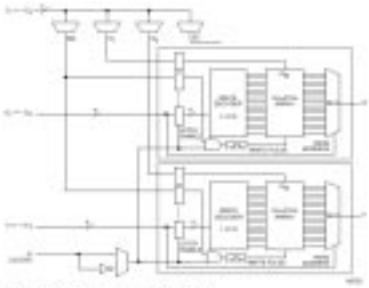
Two 4-input functions, registered output



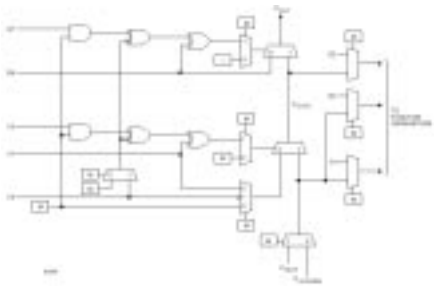
5-input function, combinational output



CLB Used as RAM



Fast Carry Logic



Xilinx 4000 Interconnect

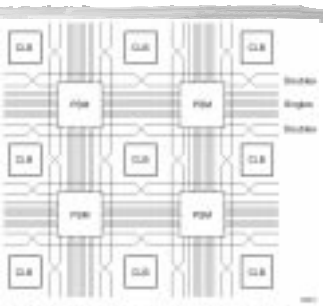


Figure 25: Single- and Double-Length Lines, with Programmable Switch Matrices (PSMs)

Switch Matrix

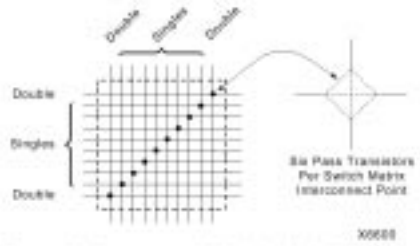
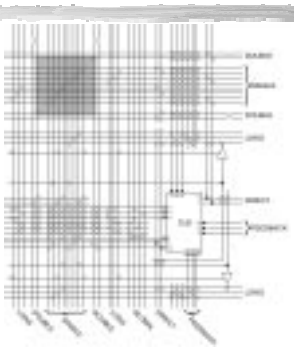
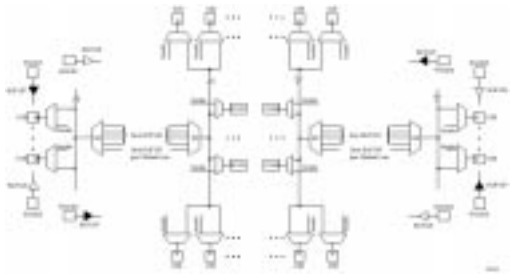


Figure 26: Programmable Switch Matrix (PSM)

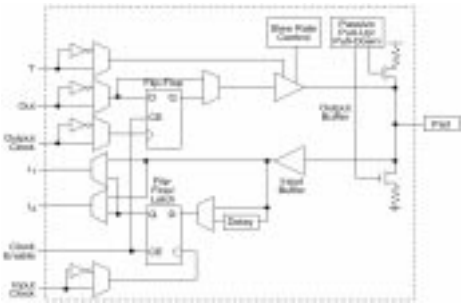
Xilinx 4000 Interconnect Details



Global Signals - Clock, Reset, Control



Xilinx 4000 IOB

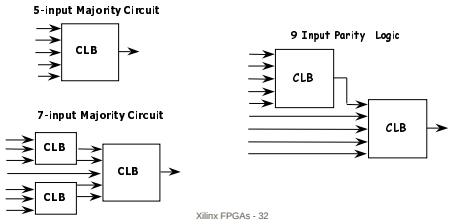


Xilinx FPGA Combinational Logic Examples

- Key: General functions are limited to 5 inputs
  - (4 even better - 1/2 CLB)
  - *No limitation on function complexity*
- Example
  - 2-bit comparator:
    A B = C D and A B > C D implemented with 1 CLB
    (GT) F = A C' + A B D' + B C' D'
    (EQ) G = A'B'C'D'+ A'B C'D + A B'C D'+ A B C D
- Can implement some functions of > 5 input

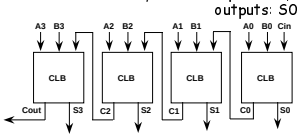
Xilinx FPGA Combinational Logic

- Examples
  - N-input majority function: 1 whenever n/2 or more inputs are 1
  - N-input parity functions: 5 input/1 CLB; 2 levels yield 25 inputs!

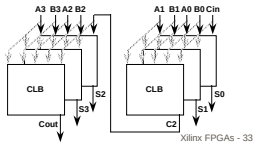


Xilinx FPGA Adder Example

- Example
  - 2-bit binary adder - inputs: A1, A0, B1, B0, CIN
    outputs: S0, S1, Cout



Full Adder, 4 CLB delays to final carry out



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

Computer-Aided Design

- Can't design FPGAs by hand
  - Way too much logic to manage, hard to make changes
- Hardware description languages
  - Specify functionality of logic at a high level
- Validation: high-level simulation to catch specification errors
  - Verify pin-outs and connections to other system components
  - Low-level to verify mapping and check performance
- Logic synthesis
  - Process of compiling HDL program into logic gates and flip-flops
- Technology mapping
  - Map the logic onto elements available in the implementation technology (LUTs for Xilinx FPGAs)

CAD Tool Path (cont'd)

- Placement and routing
  - Assign logic blocks to functions
  - Make wiring connections
- Timing analysis - verify paths
  - Determine delays as routed
  - Look at critical paths and ways to improve
- Partitioning and constraining
  - If design does not fit or is unroutable as placed split into multiple chips
  - If design it too slow prioritize critical paths, fix placement of cells, etc.
  - Few tools to help with these tasks exist today
- Generate programming files - bits to be loaded into chip for configuration

Xilinx CAD Tools

- Verilog (or VHDL) use to specify logic at a high-level
  - Combine with schematics, library components
- Synopsys
  - Compiles Verilog to logic
  - Maps logic to the FPGA cells
  - Optimizes logic
- Xilinx APR - automatic place and route (simulated annealing)
  - Provides controllability through constraints
  - Handles global signals
- Xilinx Xdelay - measure delay properties of mapping and aid in iteration
- Xilinx XACT - design editor to view final mapping results

Applications of FPGAs

- Implementation of random logic
  - Easier changes at system-level (one device is modified)
  - Can eliminate need for full-custom chips
- Prototyping
  - Ensemble of gate arrays used to emulate a circuit to be manufactured
  - Get more/better/faster debugging done than with simulation
- Reconfigurable hardware
  - One hardware block used to implement more than one function
  - Functions must be mutually-exclusive in time
  - Can greatly reduce cost while enhancing flexibility
  - RAM-based only option
- Special-purpose computation engines
  - Hardware dedicated to solving one problem (or class of problems)
  - Accelerators attached to general-purpose computers