REVERSE ENGINEERING
We can do this the easy way or the hard way.
But your protocol is getting documented!
Programmable logic

- Prototyping custom silicon is expensive
  - Need to test before tape-out
  - Simulations are sloooow
- ASIC setup cost is too high for small runs
  - What if we could use something COTS?
Programmable logic

- Can implement any digital logic function
  - Limited only by capacity of device
  - Performance depends on the function
- FPGAs
  - Lookup table (LUT) based. Covered later on.
- CPLDs
So what is a CPLD?

- Several conflicting definitions :(
- For the scope of this class, a CPLD is a programmable logic device built around one or more programmable AND+OR arrays.
- Generally nonvolatile
  - Some vendors refer to LUT-based flash PLDs as CPLDs. We consider them FPGAs.
- Typically small (<5k gate equiv)
- Typically programmed via JTAG
Any Boolean equation can be expressed as a sum of products

\[ AB + CD + EFGH + IJ(!K)(!L) \ldots \]
Programmable Logic Array (PLA)

- Generic circuit for arbitrary SOP expressions
- 2D grid of programmable AND/OR gates

Diagram:
- Input data flows into the AND array.
- Output data flows out of the OR array.
Sense amp AND array

- Ex: Xilinx XC9500* series
- Power-hungry
- Simple
- Dense
CMOS AND array

- Ex: Xilinx CoolRunner-II series
- Power-efficient
- More complex
- Larger
Simple CMOS and array cell

- Real devices usually are tree based
OR array

- Same basic idea as AND array
- Can use CMOS or sense amp designs
Xilinx XC9500XL family

- 350 nm sense amp based CPLD
- Uses flash transistors directly in logic array
  - Dense layout
  - Need to freeze chip to program
- Messy analog architecture
- Less “clean”, has lots of arch limitations
- We won't be studying this family in detail
Xilinx CoolRunner-II family

- 180 nm CMOS CPLD
- Flash-backed SRAM config cells
  - Slightly less dense (need two sets of memory)
  - Can program flash while running from RAM
- Very uniform architecture
- We will be studying the XC2C32A in detail during the rest of the course
Xilinx CoolRunner-II family

- 32, 64, 128, 256, 384, 512 macrocell devices
- 32, 64 are “low end”, others “high end”
  - Some additional features in high-end parts
- Original 32/64 had only one I/O bank
- 32A/64A have two
<table>
<thead>
<tr>
<th>XC2C32A (3.24 mm²)</th>
<th>XC2C64A (4.61 mm²)</th>
<th>XC2C128 (9.59 mm²)</th>
</tr>
</thead>
<tbody>
<tr>
<td>![XC2C32A Image]</td>
<td>![XC2C64A Image]</td>
<td>![XC2C128 Image]</td>
</tr>
<tr>
<td>X8370 (rev A)</td>
<td>X8380 (rev A)</td>
<td>X7440 (rev B)</td>
</tr>
</tbody>
</table>
XC2C32A block diagram
Die floorplan
Boot process

- Flash is structured as 49 rows x 260 bits
  - 9 FB2 macrocell bits + 1 valid bit
  - 112 FB2 PLA bits
  - 16 ZIA bits (interleaved from both FBs)
  - 112 FB1 PLA bits
  - 9 FB1 macrocell bits + 1 valid bit
- Counter loops over flash and copies to SRAM
- 48 logic array rows + 1 system config row
Function block

- 40 inputs from ZIA
- Invert to give 80
- 80x56 AND array
- 56x16 OR array
- 16 macrocells, 16 flipflops
Global interconnect (ZIA/AIM)

- Two names in use:
  - Zero [static] Power Interconnect Array
  - Advanced Interconnect Matrix

- 65 input sources
  - 32 I/O pins (16 per FB * 2 FBs)
  - 32 FFs (16 per FB * 2 FBs)
  - One input-only pin

- 40 independent outputs to each FB
Global interconnect (ZIA/AIM)

- We need a subset of 40 signals out of 65
- What about a 40x65 crossbar?
- Needs 40 65:1 muxes
  - We need 64 2:1 per output!
  - 2560 2:1 muxes for entire array
  - This is a lot of transistors and space!
  - 500 nm XC9500 parts did this
- Can we do better?
Global interconnect (ZIA/AIM)

- Yes, we can!
- The full crossbar design produces every \textit{permutation} of 40 items chosen from the 65
- We only need every \textit{combination}, ordering is unimportant
  - AND operation is commutative and associative
- Only route each input to a few outputs
Global interconnect (ZIA/AIM)

- Each row only connects to 6 out of the 65
  - Need 40 6:1 muxes instead of 40 65:1
  - 5 2:1 muxes per row instead of 64
  - >10x die area savings!

- The implicit 65:6 mux is hard-wired with a different selector for each row

- Some combinations may be unroutable :(
ZIA metal 4: Global data bus

<table>
<thead>
<tr>
<th>GND</th>
<th>VCCINT</th>
<th>0xA0</th>
<th>0x90</th>
<th>0x88</th>
<th>0x84</th>
<th>0x82</th>
<th>0x81</th>
</tr>
</thead>
</table>

[Diagram showing data bus connections and values]
ZIA metal 3: Muxes, WL, X routing
ZIA M3+M4 stack
ZIA M2
ZIA M1
ZIA poly/active
ZIA gate-level floorplan
ZIA schematic: Pulls
ZIA schematic: Tristate mux
ZIA schematic: output buffer
PLA AND array

- 40 inputs J[39:0] from ZIA
- Each AND input is J, !J, or 1
- Seems to be some kind of tree
- Not fully vectorized/analyzed yet
  - Final project anyone?
PLA OR array

- 56 inputs $P[55:0]$ from AND array
- Each input is $P$ or $0$
- Probably a tree too
- No analysis done yet
  - Final project anyone?
Several 6T SRAM structures in use

- PLA AND array (pairs)
- OR array is unpaired
- ZIA (single cells)
- Macrocells (pairs)
Macrocell product terms

- **Product term A**
  - FF set/reset
- **Product term B**
  - IOB output enable
- **Product term C (10+3N for macrocell N)**
  - FF clock enable
  - FF clock
Function block product terms

- Control term clock
  - FF clock
- Control term set, control term reset
  - FF set/reset
- Control term output enable
  - IOB output enable
Macrocells

- XOR2 gate
  - Input 1 from OR array
  - Input 2 is one of \{PTC, !PTC, 1, 0\}
  - Allows for slightly denser packing than pure PLA
- Flipflop fed by XOR or IBUF
  - Can be clocked by PTC, CTC, GCK*
  - Can be DDR/SDR, DFF/TFF/latch
  - Set/reset can be PTA, CTS/R, GSR, or unused
Macrocells

- Macrocell also stores SRAM cells for IOB cfg
- Output of XOR or FF drives OBUF
- Tristate mode can be CTE, PTB, GTS*, or off
I/O

- Your homework 1
- Configurable I/O driver with programmable tristate, drive strength, Schmitt trigger, etc
Banked I/O

- I/O pins need a power rail
- Normally higher voltage than core supply
  - CR-II runs VCCINT at 1.8V
  - VCCIO can be 1.5 to 3.3
- Divide pins into groups with separate VCCIO
  - Can work with lots of different chips without needing external level shifters
I/O bank kludges

- XC2C32 and XC2C64 had only one IO bank
  - One bit each for input Vth and output drive
- 32A and 64A have two
  - Need config per bank
- How to maintain backward compatibility?
  - Keep global config
  - One in/out level pair per bank
  - AND per-bank pair with global
System configuration row

- 49th row of config flash
- 258 data bits wide + 2 valid bits
- Contents:
  - User ID (32 bits)
  - Config-done bits (2 bits)
  - Security/lock bits (7 bits)
  - What about the rest? May be unused/free
Questions?

- TA: Andrew Zonenberg <azonenberg@drawersteak.com>
- Image credit: Some images CC-BY from:
  - John McMaster <JohnDMcMaster@gmail.com>