Announcements

• We have a new room!
  - Starting this Thursday, lectures will be held in 9 Lewis

• Fri. discussion session shifted to 3-4pm
  - Mon. discussion still 11am-12pm
  - First discussion this Friday

• Labs start today

• Homework #1 will be handed out later this week
  - Due next Thurs.
Digital Logic Review Outline

• Topics in the review, you have already seen in CS61C, and possibly EE40:
  1. Digital Signals.
  2. General model for synchronous systems.
  3. Combinational logic circuits
  4. Flip-flops, clocking [Next lecture]

Clock Signal

A source of regularly occurring pulses used to measure the passage of time.

• Waveform diagram shows evolution of signal value (in voltage) over time.
• Usually comes from an off-chip crystal-controlled oscillator.
• One main clock per chip/system.
• Distributed throughout the chip/system.
• “Heartbeat” of the system. Controls the rate of computation by directly controlling all data transfers.
Data Signals

Random adder circuit at a random point in time:

Observations:
- Most of the time, signals are in either low- or high-voltage position.
- High- or low-voltage positions aren’t necessarily at the rails; signals can even overshoot.
- Changes in the signals correspond to changes in the clock signal (but don’t change every cycle).

The facts:
1. Low-voltage represents binary 0 and high-voltage, binary 1.
2. Circuits are designed and built to be “restoring”. Deviations from ideal voltages are ignored. Outputs close to ideal.
3. In synchronous systems, all changes first initiated by clock edges.

Circuit Delay

Outputs cannot be produced instantaneously.
- In general, the delay through a circuit is called the propagation delay. It measures the time from when inputs arrive until the outputs change.
- The delay is a function of many things. Some out of the control of the circuit designer:
  - Processing technology, the particular input values.
- And others under their control:
  - Circuit structure, physical layout parameters.
Bus Signals

Signal wires grouped together often called a bus.

- \( X_0 \) is called the least significant bit (LSB)
- \( X_3 \) is called the most significant bit (MSB)
- Capital X represents the entire bus.
  - Here, hexadecimal digits are used to represent the values of all four wires.
  - The waveform for the bus depicts it as being simultaneously high and low. (The hex digits give the bit values). The waveform just shows the timing.

Combinational Logic Blocks

- Example four-input function:
  
<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>F(0,0,0)</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>F(0,0,1)</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>F(0,1,0)</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>F(0,1,1)</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>F(1,0,0)</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>F(1,0,1)</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>F(1,1,0)</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>F(1,1,1)</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>F(1,1,0)</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>F(1,1,1)</td>
</tr>
</tbody>
</table>

- Truth-table representation of a function. Output is explicitly specified for each input combination.
- In general, CL blocks have more than one output signal, in which case, the truth-table will have multiple output columns.
Example CL Block

- 2-bit adder. Takes two 2-bit integers and produces 3-bit result.

<table>
<thead>
<tr>
<th>a1</th>
<th>a0</th>
<th>b1</th>
<th>b0</th>
<th>c2</th>
<th>c1</th>
<th>c0</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
</tr>
<tr>
<td>00</td>
<td>01</td>
<td>00</td>
<td>01</td>
<td>01</td>
<td>01</td>
<td>01</td>
</tr>
<tr>
<td>00</td>
<td>10</td>
<td>01</td>
<td>10</td>
<td>01</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>00</td>
<td>11</td>
<td>01</td>
<td>11</td>
<td>01</td>
<td>11</td>
<td>11</td>
</tr>
<tr>
<td>01</td>
<td>00</td>
<td>01</td>
<td>00</td>
<td>01</td>
<td>01</td>
<td>01</td>
</tr>
<tr>
<td>01</td>
<td>01</td>
<td>01</td>
<td>01</td>
<td>01</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>01</td>
<td>10</td>
<td>01</td>
<td>10</td>
<td>01</td>
<td>11</td>
<td>11</td>
</tr>
<tr>
<td>01</td>
<td>11</td>
<td>01</td>
<td>11</td>
<td>01</td>
<td>11</td>
<td>11</td>
</tr>
<tr>
<td>10</td>
<td>00</td>
<td>10</td>
<td>00</td>
<td>10</td>
<td>01</td>
<td>11</td>
</tr>
<tr>
<td>10</td>
<td>01</td>
<td>10</td>
<td>01</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>11</td>
<td>11</td>
</tr>
<tr>
<td>10</td>
<td>11</td>
<td>10</td>
<td>11</td>
<td>10</td>
<td>11</td>
<td>11</td>
</tr>
<tr>
<td>11</td>
<td>00</td>
<td>11</td>
<td>00</td>
<td>11</td>
<td>01</td>
<td>11</td>
</tr>
<tr>
<td>11</td>
<td>01</td>
<td>11</td>
<td>01</td>
<td>11</td>
<td>10</td>
<td>11</td>
</tr>
<tr>
<td>11</td>
<td>10</td>
<td>11</td>
<td>10</td>
<td>11</td>
<td>11</td>
<td>11</td>
</tr>
</tbody>
</table>

- Think about truth table for a 32-bit adder. It’s possible to write out, but it might take a while!

*Any* combinational logic function can be implemented as a network of logic gates.

Logic “Gates”

- Logic gates are often the primitive elements out of which combinational logic circuits are constructed.
  - In some technologies, there is a one-to-one correspondence between logic gate representations and actual circuits.
  - Other times, we use them just as another abstraction layer (FPGAs have no real logic gates).
What Makes a Circuit Digital?

Restoration

- A necessary property of any successful technology for logic circuits is "Restoration".
- **Digital** circuits need:
  - to ignore noise and other non-idealities at their inputs, and
  - generate "cleaned-up" signals at their output.
- Otherwise, each stage would propagate input noise to their output and eventually noise and other non-idealities would accumulate and signal content would be lost.
Inverter Example Restoration

- Look at 1-input gate for simplicity:

  ![Idealized Inverter](image1)

  ![Actual Inverter](image2)

- The inverter acts like a non-linear amplifier
- Non-linearity is critical to restoration
- Other gates behave similarly with respect to input/output relationship

Field Programmable Gate Array (FPGA)
Introduction
**FPGA Overview**

- Basic idea: two-dimensional array of logic blocks and flip-flops with a means for the user to configure (program):
  1. the interconnection between the logic blocks,
  2. the function of each block.

**Simplified version of FPGA internal architecture:**

---

**Why are FPGAs Interesting?**

- Main advantage: **design cost/time**
  - In comparison to a custom chip (ASIC)

- ASIC – can’t modify the design after the fact easily
  - Generally gave to re-fabricate the chip from scratch in order to change the hardware

- FPGA – hardware can be “changed” simply be reconfiguring the logic blocks/interconnects
  - Can re-spin within a few minutes/hours
  - This often out-weights the low-level overhead of supporting this reconfiguration vs. a custom, fixed-function ASIC
    - FPGA vs. ASIC: 40X area, 3-4x delay, 12x power
      (Kuon and Rose, FPGA ‘06)
Why are FPGAs Interesting?

- Staggering logic capacity growth
  - Tracked Moore’s law extremely well

<table>
<thead>
<tr>
<th>Year Introduced</th>
<th>Device</th>
<th>Logic Cells</th>
<th>“logic gate equivalents”</th>
</tr>
</thead>
<tbody>
<tr>
<td>1985</td>
<td>XC2064</td>
<td>128</td>
<td>1024</td>
</tr>
<tr>
<td>2011</td>
<td>XC7V2000T</td>
<td>1,954,560</td>
<td>15,636,480</td>
</tr>
</tbody>
</table>

Why Are FPGAs Interesting?

- Logic now only a part of the story: on-chip RAMs, high-speed I/O, “hard” function blocks...
- Modern FPGAs are “reconfigurable systems”
Background (review) for upcoming

- A **MUX** or multiplexor is a combinational logic circuit that chooses between $2^N$ inputs under the control of N control signals.

- A **latch** is a 1-bit memory [similar to a flip-flop].

FPGA Variations

- Families of FPGA’s differ in:
  - physical means of implementing user programmability,
  - arrangement of interconnection wires, and
  - the basic functionality of the logic blocks.

- Most significant difference is in the method for providing flexible blocks and connections:
  - Anti-fuse based (ex: Actel)
    - Non-volatile, relatively small
    - fixed (non-reprogrammable)

- Several “floating gate” or eprom style approaches have been used. One now by Actel.
User Programmability

- Latch-based (Xilinx, Altera, ...)

Latches are used to:
1. control a switch to make or break cross-point connections in the interconnect
2. define the function of the logic blocks
3. set user options:
   - within the logic blocks
   - in the input/output blocks
   - global reset/clock
- “Configuration bit stream” is loaded under user control

+ reconfigurable
- volatile
  - relatively large.

Idealized FPGA Logic Block

- 4-input look up table (LUT)
  - implements combinational logic functions
- Register
  - optionally stores output of LUT
4-LUT Implementation

- n-bit LUT is a direct implementation of a logic truth-table.
- n-bit LUT is implemented as a $2^n \times 1$ memory:
  - inputs choose one of $2^n$ memory locations.
  - memory locations (latches) are normally loaded with values from user's configuration bit stream.
  - Inputs to mux control are the CLB inputs.
- Result is a general purpose "logic gate".
  - n-LUT can implement any function of n inputs!

FPGA Generic Design Flow

- **Design Entry:**
  - Create your design files using:
    - schematic editor or
    - HDL (hardware description languages: Verilog, VHDL)
- **Design Implementation:**
  - Logic synthesis (in case of using HDL entry) followed by,
  - Partition, place, and route to create configuration bit-stream file
- **Design verification:**
  - Optionally use simulator to check function,
  - Load design onto FPGA device (cable connects PC to development board), optional "logic scope" on FPGA
  - check operation at full speed in real environment.
Example Placement, Partition, and Route

- Idealized FPGA structure:

- Example Circuit:
  - collection of gates and flip-flops

Circuit combinational logic must be "covered" by 4-input 1-output LUTs.

Flip-flops from circuit must map to FPGA flip-flops.
(Best to preserve "closeness" to CL to minimize wiring.)

Best placement in general attempts to minimize wiring.
Vdd, GND, clock, and global resets are all "prewired".

Example Partition, Placement, and Route

Example Circuit:
- collection of gates and flip-flops

Two partitions. Each has single output, no more than 4 inputs, and no more than 1 flip-flop. In this case, inverter goes in both partitions.

Note: the partition can be arbitrarily large as long as it has not more than 4 inputs and 1 output, and no more than 1 flip-flop.
Xilinx Virtex-5 XC5VLX110T

Colors represent different types of resources:

- Logic
- Block RAM
- DSP (ALUs)
- Clocking
- I/O
- Serial I/O + PCI

A routing fabric runs throughout the chip to wire everything together.
Configurable Logic Blocks (CLBs)

Slices define regular connections to the switching fabric, and to slices in CLBs above and below it on the die.

The LX110T has 17,280 slices.

X-Y naming convention for slices

- X0, X2, ... are lower CLB slices.
- X1, X3, ... are upper CLB slices.
- Y0, Y1, ... are CLB column positions.
Atoms: 5-input Look Up Tables (LUTs)

Computes any 5-input logic function.

Timing is independent of function.

Latches set during configuration.

Virtex-5 6-LUTs: Composition of 5-LUTs

May be used as one 6-input LUT (D6 out) ...

... or as two 5-input LUTS (D6 and D5)

The LX110T has 69,120 6-LUTs
6-LUT delay is 0.9 ns

Combinational logic (post configuration)
The Simplest View of a Slice

Four 6-LUTs

Four Flip-Flops

Switching fabric may see combinational and registered outputs.

An actual Virtex-5 slice adds many small features to this simplified diagram. We show them one by one ...

Two 7-LUTs per Slice

Extra multiplexers (F7AMUX, F7BMUX)

Extra inputs (AX and CX)
Or One 8-LUT per Slice

Third multiplexer (F8MUX)

---

Third input (BX)

---

Configuring the “n” of an n-LUT ...

Extra muxes to chose LUT option ...

From eight 5-LUTs ... to one 8-LUT.

---

Combinational or registered outs.

---

Flip-flops unused by LUTs can be used standalone.

---

Flip-flops ...
Slice Flip-Flop Properties

Each state element may be edge-triggered FF or latch.

Clock enable, clock polarity, and set/reset lines in a slice are shared.

Each state element may respond differently to set/reset signal.

Next: The vertical dimension ...

Vertex-5 Vertical Logic

We can map ripple-carry addition onto carry-chain block.

The carry-chain block also useful for speeding up other adder structures and counters.
Putting it all together ... a SLICEL.

The previous slides explain all SLICEL features.

About 50% of the 17,280 slices in an LX110T are SLICELs.

The other slices are SLICEMs, and have extra features.

Recall: 5-LUT architecture ...  

- 32 Latches.
- Configured to 1 or 0.
- Some parts of a logic design need many state elements.

SLICEMs replace normal 5-LUTs with circuits that can act like 5-LUTs, but can alternatively use the 32 latches as RAM, ROM, shift registers.
A SLICEM 6-LUT

Memory data input

Normal 6-LUT inputs.

Normal 5/6-LUT outputs.

Memory data input.

Control output for chaining LUTs to make larger memories.

A 1.1 Mb distributed RAM can be made if all SLICEMs of an LX110T are used as RAM.

Many RAM Configurations Possible ...

Example configuration: Single-port 256b x 1, registered output.

A complete list:

- Single-Port 32 x 1-bit RAM
- Dual-Port 32 x 1-bit RAM
- Quad-Port 32 x 1-bit RAM
- Simple Dual-Port 32 x 2-bit RAM
- Single-Port 64 x 1-bit RAM
- Dual-Port 64 x 1-bit RAM
- Quad-Port 64 x 1-bit RAM
- Simple Dual-Port 64 x 2-bit RAM
- Single-Port 128 x 1-bit RAM
- Dual-Port 128 x 1-bit RAM
- Single-Port 256 x 1-bit RAM

A 128 x 32b LUT RAM has a 1.1ns access time.
SLICEM shift register (one of many).

See Virtex-5 User Guide for a complete list of shift-register types.

SLICEL vs SLICEM...

SLICEM adds memory features to LUTs, + muxes.
Virtex-5 DSP48E Slice

Efficient implementation of multiply, add, bit-wise logical. LX110T has 64 in a single column.
Throughout the semester, we will look at different Virtex-5 features in-depth.

Switch fabric
Block RAM
DSP48 (ALUs)
Clocking
I/O
Serial I/O + PCI