Outline

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

Integrated Circuit Example

• PowerPC microprocessor microphotograph
  - Superscalar (3 instructions/cycle)
  - 6 execution units (2 integer and 1 double precision IEEE floating point)
  - 32 KByte Instruction and Data L1 caches
  - Dual Memory Management Units (MMU)
  - External L2 Cache interface with integrated controller and cache tags.

*Comprises only transistors and wires.*

Connections to outside world (ex. motherboard):
• Memory interface
• Power (Vdd, GND)
• Clock input

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.
• Usually one per chip/system.
• Distributed throughout the chip.
• “Heartbeat” of the system. Controls the rate of computation by directly controlling all data transfers.

T represents the time of one clock cycle.
Data Signals

Random adder circuit at a random point in time:

Observations:
1. Most of the time, signals are in either low- or high-voltage position.
2. When the signals are at the high- or low-voltage positions, they are not all the way to the voltage extremes (or they are past).
3. Changes in the signals correspond to changes in clock signal (but don’t change every cycle).

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

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.

Circuit Delay

Digital circuits cannot produce outputs 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 amount is a function of many things. Some out of the control of the circuit designer:
  - Processing technology, the particular input values.
- And others under her control:
  - Circuit structure, physical layout parameters.

Registers

- The adder circuit discussed thus far is an example of a combinational logic circuit.
  - Its output changes as soon as a new input is presented (after a small delay).
  - Its output is a function of only the current inputs (has no memory of past inputs).
- Combinational logic circuits do the real work of computations, ex: arithmetic functions. They are combined with circuits that remember their inputs - registers.
  - Under the control of the "LOAD" signal the register captures its input. After a short delay it appears at the output. The output doesn’t change again until another load signal.
  - Sometimes the clock signal is used as LOAD, in which case the register loads a new value every clock cycle.
Accumulator Circuit Example

Assume X is a vector of N integers, presented to the input of our accumulator circuit one at a time (one per clock cycle), so that after N clock cycles, S hold the sum of all N numbers.

\[ S = 0; \text{ Repeat N times}\]
\[ S = S + X; \]

- We need something like this:
- But not quite.
- Need to use the clock signal to hold up the feedback to match up with the input signal.

Register Details

- A \( n \)-bit wide register is nothing but a set of flip-flops (1-bit wide registers) with a common load/\( clk \) signal.

- A flip-flop captures its input on the edge of the clock (rising edge in this case - positive edge flip-flop). The new input appears at the output after a short delay.

Accumulator Circuit

- Put register, with clock signal controlling its load, in feedback path.
- On each clock cycle the register prevents the new value from reaching the input to the adder prematurely. (The new value just waits at the input of the register.)

Break

- Announcements:
  - Lab lecture will be held as listed by Telebears - Fridays 1-2 in 125 Cory.
  - Quizzes will be in class on Thursday, starting next week.
  - Reading for today’s lecture are linked to Tuesday’s lecture on the web.
  - Homework due next week will be posted later today or early tomorrow.
  - New reading assignments will be posted later today or early tomorrow.

- Questions from Tuesday about class information or content?
Flip-Flop Timing Details

Three important times associated with flip-flops:
- setup time
- hold time
- clock-to-q delay.

Accumulator Revisited

- Note:
  - Reset signal (synchronous)
  - Timing of X signal is not known without investigating the circuit that supplies X. Here we assume it comes just after $S_{i-1}$.
  - Observe transient behavior of $S_i$.

General Model for Synchronous Systems

- All synchronous digital systems fit this model:
  - Collections of combinational logic blocks and state elements connected by signal wires. These form a directed graph with only two types of nodes (although the graph need not be bi-partite.)
  - Instead of simple registers, sometimes the state elements are large memory blocks.

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(0,0,0,0)</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>F(0,0,0,1)</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>F(0,0,1,0)</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>F(0,0,1,1)</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>F(0,1,0,0)</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>F(0,1,0,1)</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>F(0,1,1,0)</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>F(0,1,1,1)</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>F(1,0,0,0)</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>F(1,0,1,1)</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>F(1,1,1,0)</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>F(1,1,1,1)</td>
</tr>
</tbody>
</table>

- True-table representation of 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>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>e</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

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

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).
- How about these gates with more than 2 inputs?
- Do we need all these types?

Example Circuit with Gates

- How do we know that these two representations are equivalent?

Boolean Algebra

- Boolean equation:
  \[ y = ab + bc + ac \]

- Same circuit, this time Boolean equation representation.
- How do we know that these two representations are equivalent?
- Are there other equivalent equations for y?
- Why do we need three different representations for combinational logic circuits?
Boolean Algebra Manipulation

- Boolean equation representations allow circuits to be manipulated algebraically.
- Example:

\[
y = ab + a + c = a(b + 1) + c = a(1) + c = a + c
\]

Representational Trilogy for CL Circuits

- Every combinational logic circuit can be represented in three forms:
  - Truth table:
    - unique representation
    - Sometimes (almost always) too large to be practical.
  - Boolean expression:
    - Good for algebraic manipulation.
    - Always compact representation of complex circuits, when factoring is possible.
  - Gate diagram:
    - Close to the implementation.
    - Sometimes more intuitive.

Are there other representations forms for combinational logic circuits?

Yes, one used commonly for manipulation by computer is Binary Decision Diagrams (BDDs).

Another one, used for manipulation by hand, is Karnough Maps.