L15: Custom and ASIC VLSI Integration

Acknowledgements:

Materials in this lecture are courtesy of the following people and used with permission.


- Curt Schurgers
Follow simple design rules (contract between process and circuit designers)

(Courtesy of Chris Terman. Used with permission.)
Custom Design/Layout

Adder stage 1
Adder stage 2
Adder stage 3
Bit slice 0
Bit slice 1
Bit slice 2
Bit slice 63

Hand crafting the layout to achieve maximum clock rates (> 1Ghz)
Exploits regularity in datapath structure to optimize interconnects
The ASIC Approach

Most Common Design Approach for Designs up to 500Mhz Clock Rates
Standard Cell Example

3-input NAND cell (from ST Microelectronics):
C = Load capacitance
T = input rise/fall time

Each library cell (FF, NAND, NOR, INV, etc.) and the variations on size (strength of the gate) is fully characterized across temperature, loading, etc.
2-level metal technology

- With limited interconnect layers, dedicated routing channels between rows of standard cells are needed
- Width of the cell allowed to vary to accommodate complexity
- Interconnect plays a significant role in speed of a digital circuit

Current Day Technology

Cell-structure hidden under interconnect layers
module adder64 (a, b, sum);
  input [63:0] a, b;
  output [63:0] sum;
  assign sum = a + b;
endmodule

Verilog to ASIC Layout (the push button approach)

After Synthesis

After Routing

After Placement
Wire-to-wire capacitance causes inter-wire delay dependencies

Iterative Removal of Timing Violations (white lines)
Macro Modules

256×32 (or 8192 bit) SRAM Generated by hard-macro module generator

- Generate highly regular structures (entire memories, multipliers, etc.) with a few lines of code
- Verilog models for memories automatically generated based on size
For 1Ghz clock, skew budget is 100ps.

Variations along different paths arise from:

- **Device**: $V_T$, W/L, etc.
- **Environment**: $V_{DD}$, °C
- **Interconnect**: dielectric thickness variation
The IR-drop problem causes internal power supply voltage to be less than the external source

(Courtesy of Prof. David Blaauw. Used with permission.)
Analog Circuits: Clock Frequency Multiplication (Phase Locked Loop)

- **VCO** ➔ produces high frequency square wave
- **Divider** ➔ divides down VCO frequency
- **PFD** ➔ compares phase of ref and div
- **Loop filter** ➔ extracts phase error information

Used widely in digital systems for clock synthesis
(a standard IP block in most ASIC flows)
(Courtesy of Michael Perrott. Used with permission.)
Behavioral Transformations

- There are a large number of implementations of the same functionality
- These implementations present a different point in the area-time-power design space
- Behavioral transformations allow exploring the design space a high-level

Optimization metrics:

1. **Area** of the design
2. **Throughput** or sample time $T_S$
3. **Latency**: clock cycles between the input and associated output change
4. **Power** consumption
5. **Energy** of executing a task
6. …
Conventional Multiplication

\[ Z = X \cdot Y \]

<table>
<thead>
<tr>
<th></th>
<th>(X_3)</th>
<th>(X_2)</th>
<th>(X_1)</th>
<th>(X_0)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Y_3)</td>
<td>(X_3 \cdot Y_0)</td>
<td>(X_2 \cdot Y_0)</td>
<td>(X_1 \cdot Y_0)</td>
<td>(X_0 \cdot Y_0)</td>
</tr>
<tr>
<td>(Y_2)</td>
<td>(X_3 \cdot Y_1)</td>
<td>(X_2 \cdot Y_1)</td>
<td>(X_1 \cdot Y_1)</td>
<td>(X_0 \cdot Y_1)</td>
</tr>
<tr>
<td>(Y_1)</td>
<td>(X_3 \cdot Y_2)</td>
<td>(X_2 \cdot Y_2)</td>
<td>(X_1 \cdot Y_2)</td>
<td>(X_0 \cdot Y_2)</td>
</tr>
<tr>
<td>(Y_0)</td>
<td>(X_3 \cdot Y_3)</td>
<td>(X_2 \cdot Y_3)</td>
<td>(X_1 \cdot Y_3)</td>
<td>(X_0 \cdot Y_3)</td>
</tr>
</tbody>
</table>

| \(Z_7\) | \(Z_6\) | \(Z_5\) | \(Z_4\) | \(Z_3\) | \(Z_2\) | \(Z_1\) | \(Z_0\) |

Constant multiplication (become hardwired shifts and adds)

\[ Z = X \cdot (1001)_2 \]

<table>
<thead>
<tr>
<th></th>
<th>(X_3)</th>
<th>(X_2)</th>
<th>(X_1)</th>
<th>(X_0)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1)</td>
<td>(1)</td>
<td>(0)</td>
<td>(0)</td>
<td>(1)</td>
</tr>
</tbody>
</table>

| \(Z_7\) | \(Z_6\) | \(Z_5\) | \(Z_4\) | \(Z_3\) | \(Z_2\) | \(Z_1\) | \(Z_0\) |

\[ Y = (1001)_2 = 2^3 + 2^0 \]

X \[\rightarrow\] \(<< 3\) \[\rightarrow\] \(+\) \[\rightarrow\] Z

shifts using wiring
Canonical signed digit representation is used to increase the number of zeros. It uses digits \{-1, 0, 1\} instead of only \{0, 1\}.

**Iterative encoding:** replace string of consecutive 1’s

\[
0 \ 1 \ 1 \ \ldots \ 1 \ 1 \quad \Rightarrow \quad 1 \ 0 \ 0 \ \ldots \ 0 \ -1
\]

\[2^{N-2} + \ldots + 2^1 + 2^0\]

\[2^{N-1} - 2^0\]

**Worst case CSD has 50% non zero bits**

\[01101111\]

\[\begin{array}{cccccccc}
0 & 1 & 1 & 0 & 1 & 1 & 1 & 1 \\
\| & & & & & & & \\
10010001 & -1
\end{array}\]

\[\begin{array}{cccccccc}
0 & 1 & 1 & 1 & 0 & 0 & 0 & -1 \\
\downarrow & & & & & & & \\
1 & 0 & 0 & -1 & 0 & 0 & 0 & -1
\end{array}\]

**X**

\(\ll 7\)

\(\ll 4\)

\(\ll 7\) +

\(\ll 4\) -

\(\downarrow\)

\(\downarrow\)

\(\ll 7\)

\(\ll 4\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)

\(\downarrow\)
Algebraic Transformations

Commutativity

\[ A + B = B + A \]

Distributivity

\[ (A + B) C = AB + BC \]

Associativity

\[ (A + B) + C = A + (B+C) \]

Common sub-expressions
Transforms for Efficient Resource Utilization

Time multiplexing: mapped to 3 multipliers and 3 adders

Reduce number of operators to 2 multipliers and 2 adders

distributivity
Retiming is the action of moving delay around in the systems

- Delays have to be moved from ALL inputs to ALL outputs or vice versa.

**Cutset retiming:** A cutset intersects the edges, such that this would result in two disjoint partitions of these edges being cut. To retime, delays are moved from the ingoing to the outgoing edges or vice versa.

**Benefits of retiming:**

- Modify critical path delay
- Reduce total number of registers

(Courtesy of Prof. Charles E. Leiserson. Used with permission.)
Retiming Example: FIR Filter

Direct form

\[ y(n) = h(n) \otimes x(n) = \sum_{i=0}^{K} x(n-i) \cdot h(i) \]

Transposed form

Note: here we use a first cut analysis that assumes the delay of a chain of operators is the sum of their individual delays. This is not accurate.
Pipelining, Just Another Transformation (Pipelining = Adding Delays + Retiming)

Contrary to retiming, pipelining adds extra registers to the system.

How to pipeline:
1. Add extra registers at *all* inputs
2. Retime
The Power of Transforms: Lookahead

\[
y(n) = x(n) + A y(n-1)
\]

Try pipelining this structure

\[
y(n) = x(n) + A[x(n-1) + A y(n-2)]
\]

How about pipelining this structure!

\[
x(n)
\]

\[
A
\]

\[
A^2
\]

\[
D
\]

\[
2D
\]
Scan Testing

Idea: have a mode in which all registers are chained into one giant shift register which can be loaded/read-out bit serially. Test remaining (combinational) logic by

1. in “test” mode, shift in new values for all register bits thus setting up the inputs to the combinational logic
2. clock the circuit once in “normal” mode, latching the outputs of the combinational logic back into the registers
3. in “test” mode, shift out the values of all register bits and compare against expected results.

Adapted from C. Terman and IEEE Press.
Map algorithms directly to silicon - bypass writing Verilog!
(Courtesy of R. Brodersen. Used with permission.)
Fingerprinting is a technique to deter people from illegally redistributing legally obtained IP by enabling the author of the IP to uniquely identify the original buyer of the resold copy.

The essence of the watermarking approach is to encode the author's signature. The selection, encoding, and embedding of the signature must result in minimal performance and storage overhead.

(Images removed due to copyright considerations.)