EECS150 - Digital Design
Lecture 12 - Combinational Logic Circuits
Part 3

March 4, 2002
John Wawrzynek
Xilinx OKs deal to get IBM’s aid in manufacturing

By Matthew Yi
Chronicle Staff Writer

San Jose chipmaker Xilinx Inc. has signed a two-year, $100 million agreement to use IBM’s latest technology manufacturing facilities.

The deal, to be announced today, means Xilinx’s new Virtex-II chips, which will be used in cellular telephone base stations, will be made utilizing IBM’s 0.13 micron manufacturing process.

While sample products have been shipped to some customers, volume production is scheduled to begin in six months, said Wim Roelandts, Xilinx president and chief executive officer.

Xilinx will also have access to IBM’s chip plants when it moves to 0.10 micron line width process on larger 300mm silicon wafers, he said.

The combination would enable the San Jose firm to make smaller and better-performing chips and squeeze more of them on each wafer, translating to cost savings.

“This is a big deal for us,” said Roelandts. “It gives us earlier access to new manufacturing technologies.”

Xilinx also will be able to bring new products to market faster by working with IBM engineers early in the chip design process. That alone can account for trimming roughly six months off the entire process, Roelandts said.

From Akiki, manager of semiconductor contract manufacturing at IBM, said the partnership with Xilinx will be a model for his company to create similar relationships with other chipmakers who rely on foundry services.

“It is a key deal for the micro-electronic division of IBM,” he said.

Analysts say Xilinx’s efforts to produce smaller chips on bigger wafers is especially significant because Virtex-II chips are big in size.

“In this recovery-year market, any little extra push you can get in the market in terms of cutting your costs or improving your yields certainly will be welcome,” said Will Strauss, an analyst at Forward Concepts.

Mark Edelestone, an analyst at Morgan Stanley, says partnering with a notable player like IBM is not a bad move.

“At this point, Xilinx is the leading (programmable logic devices) company and as it effectively locks up very state-of-the-art manufacturing, and if they execute, that’ll further enhance their already strong position,” Edelestone said.

E-mail Matthew Yi at my@sfchronicle.com.
Multiplication

\[
\begin{array}{cccc}
  a_3 & a_2 & a_1 & a_0 \\
  b_3 & b_2 & b_1 & b_0 \\
\end{array}
\]

Multiplcand

\[
\begin{array}{cccc}
  a_3b_0 & a_2b_0 & a_1b_0 & a_0b_0 \\
  a_3b_1 & a_2b_1 & a_1b_1 & a_0b_1 \\
  a_3b_2 & a_2b_2 & a_1b_2 & a_0b_2 \\
  a_3b_3 & a_2b_3 & a_1b_3 & a_0b_3 \\
\end{array}
\]

Multiplier

\[
\begin{array}{cccc}
  a_3b_0 & a_2b_0 & a_1b_0 & a_0b_0 \\
  a_3b_1 & a_2b_1 & a_1b_1 & a_0b_1 \\
  a_3b_2 & a_2b_2 & a_1b_2 & a_0b_2 \\
  a_3b_3 & a_2b_3 & a_1b_3 & a_0b_3 \\
\end{array}
\]

Partial products

\[
\begin{array}{cccc}
  a_3b_0 & a_2b_0 & a_1b_0 & a_0b_0 \\
  a_3b_1 & a_2b_1 & a_1b_1 & a_0b_1 \\
  a_3b_2 & a_2b_2 & a_1b_2 & a_0b_2 \\
  a_3b_3 & a_2b_3 & a_1b_3 & a_0b_3 \\
\end{array}
\]

Product

Many different circuits exist for multiplication. Each one has a different balance between speed (performance) and amount of logic (cost).
“Shift and Add” Multiplier

- Sums each partial product, one at a time.
- In binary, each partial product is shifted versions of A or 0.

Control Algorithm:
1. \( P \leftarrow 0, A \leftarrow \text{multiplicand}, \)  
   \( B \leftarrow \text{multiplier} \)
2. If LSB of \( B \) == 1 then add \( A \) to \( P \)  
   else add 0
3. Shift \([P][B]\) right 1
4. Repeat steps 2 and 3 \( n-1 \) times.
5. \([P][B]\) has product.

- Cost \( \alpha n, T = n \) clock cycles.
- What is the critical path for determining the min clock period?
“Shift and Add” Multiplier

Signed Multiplication:

Remember for 2’s complement numbers MSB has negative weight:

\[ X = \sum_{i=0}^{N-2} x_i 2^i - x_{n-1} 2^{n-1} \]

ex: \(-6 = 11010_2 = 0 \cdot 2^0 + 1 \cdot 2^1 + 0 \cdot 2^2 + 1 \cdot 2^3 - 1 \cdot 2^4\)

\[ = 0 + 2 + 0 + 8 - 16 = -6 \]

• Therefore for multiplication:
  a) subtract final partial product
  b) sign-extend partial products

• Modifications to shift & add circuit:
  a) adder/subtractor
  b) sign-extender on P shifter register
Array Multiplier

Generates all partial products simultaneously.

Each row: n-bit adder with AND gates.

What is the critical path? Delay $\alpha$?, cost $\alpha$?
Carry-save Addition

- Speeding up multiplication is a matter of speeding up the summing of the partial products.
- “Carry-save” addition can help.
- Carry-save addition passes (saves) the carries to the output, rather than propagating them.

Example: sum three numbers, $3_{10} = 0011$, $2_{10} = 0010$, $3_{10} = 0011$

\[
\begin{align*}
3_{10} & \quad 0011 \\
+ \quad 2_{10} & \quad 0010 \\
\text{c} & \quad 0100 = 4_{10} \\
\text{s} & \quad 0001 = 1_{10}
\end{align*}
\]

In general, carry-save addition takes in 3 numbers and produces 2.
Whereas, carry-propagate takes 2 and produces 1.
With this technique, we can avoid carry propagation until final addition.

\[
\begin{align*}
3_{10} & \quad 0011 \\
\text{c} & \quad 0010 = 2_{10} \\
\text{s} & \quad 0110 = 6_{10} \\
1000 & \quad = 8_{10}
\end{align*}
\]
Carry Save Circuits

For adding sets of numbers carry-save can be used on all but the final sum:

standard adder (carry propagate)
Array Multiplier with Carry-Save
Carry Save Addition on sets of numbers is associative and commutative.

Ex:

\[
(((x_0 + x_1) + x_2) + x_3) = \\
((x_0 + x_1) + (x_2 + x_3))
\]
Wallace Tree Multiplier

\[ a_B, a_B, a_B, a_B, a_B, a_B \]

\[ \text{partial products} \]

\[ \text{fast adder} \]

\[ \text{delay} \alpha \log_2 n \]

\[ \text{CPA} \]
CL Circuits from Mano

- Magnitude Comparator
- Multiplexors (revisited)
- Decoders
  - basic
  - hierarchical
- Encoders
  - standard
  - Priority Encoder
Magnitude Comparator

- We studied magnitude comparators in CS61c as part of the MIPS processor design.
- What was that method? Why that then and not now?

\[(A > B) = A_3B'_3 + x_3A_2B'_2 + x_3x_2A_1B'_1 + x_3x_2x_1A_0B'_0\]

\[(A > B) = A'_3B_3 + x_3A'_2B_2 + x_3x_2A'_1B_1 + x_3x_2x_1A'_0B_0\]

\[(A = B) = x_3x_2x_1x_0\]
Multiplexors Revisited

- Basic AND/OR form
- NAND/NAND
- tristate buffer based
- transmission gate based
- hierarchical
- decoder based
  - delay analysis
Decoders

\[ D_0 = x'y'z' \]
\[ D_1 = x'y'z \]
\[ D_2 = x'yz' \]
\[ D_3 = x'yz \]
\[ D_4 = xy'z' \]
\[ D_5 = xy'z \]
\[ D_6 = xyz' \]
\[ D_7 = xyz \]

Fig. 4-18 3-to-8-Line Decoder
Hierarchical Decoders

(a) Logic diagram

(b) Truth table

Fig. 4-19 2-to-4-Line Decoder with Enable Input

Fig. 4-20 4 × 16 Decoder Constructed with Two 3 × 8 Decoders
Encoders

- Generates *binary code* at output corresponding to input code.
- Example: one-hot to binary encoder. (Opposite of decoder)

<table>
<thead>
<tr>
<th>a b c d</th>
<th>x y</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 0 0 0</td>
<td>0 0</td>
</tr>
<tr>
<td>0 1 0 0</td>
<td>0 1</td>
</tr>
<tr>
<td>0 0 1 0</td>
<td>1 0</td>
</tr>
<tr>
<td>0 0 0 1</td>
<td>1 1</td>
</tr>
</tbody>
</table>

- Priority Encoder:
  - If two or more inputs are equal to 1 at the same time the input with the highest “priority” will take precedence.

<table>
<thead>
<tr>
<th>a b c d</th>
<th>x y</th>
<th>V</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0 0</td>
<td>- -</td>
<td>0</td>
</tr>
<tr>
<td>1 0 0 0</td>
<td>0 0 0</td>
<td></td>
</tr>
<tr>
<td>- 1 0 0</td>
<td>0 1 1</td>
<td></td>
</tr>
<tr>
<td>- - 1 0</td>
<td>1 0 1</td>
<td></td>
</tr>
<tr>
<td>- - - 1</td>
<td>1 1 1</td>
<td></td>
</tr>
</tbody>
</table>

- “V” is the valid signal, “d” has highest priority.