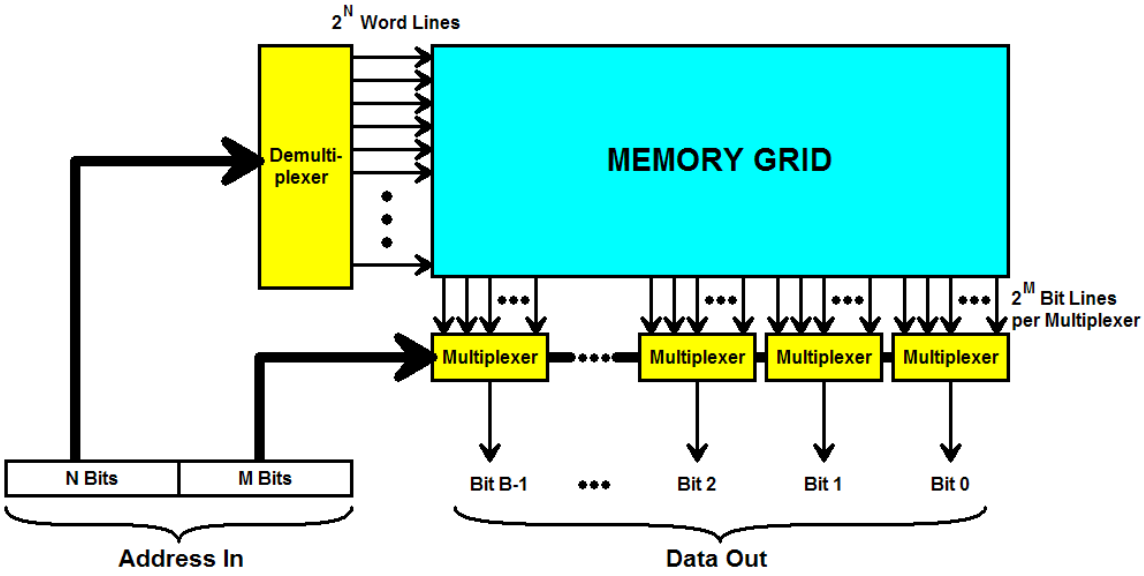
Lecture #26 – April 14, 2004

Memory Minimization and Multiplication

Minimization of Gates in Memory Systems

For any number of address lines and word size in a memory system there is an optimal arrangement of bits in the memory grid which minimizes the overall number of required gates. The general layout of such a memory system is shown below, where some of the address bits select one word line through a demultiplexer and the rest of the address bits extract the desired portion of the selected word through a set of multiplexers.



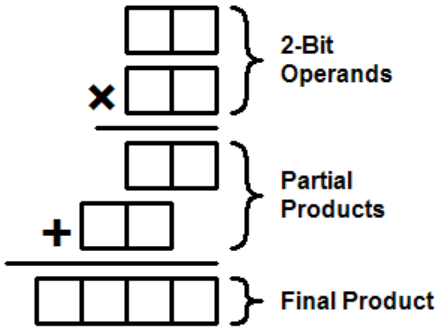
For example, a 64K byte memory system requires 16 address bits and 8 data bits. The grid contains $2^{16}\times2^{3}=2^{19}$ data bits. Arranging the memory as 64K rows (word lines) by 8 columns (bit lines) requires a demultiplexer with 2^{16} AND-gates, each with 16 inputs, and no multiplexers. This is 65536 separate AND-gates.

Arranging the same system with 10 bits to the demultiplexer and 6 bits to a bank of multiplexers gives a grid with $2^{10}=1024$ word lines and $8\times2^{6}=2^{9}=512$ bit lines. The demultiplexer now requires 2^{10} 10-input AND-gates, and each of the eight multiplexers requires 2^{6} AND-gates, each with 7 inputs (6 for the address lines and 1 for the bit line). The total is thus $2^{10} + 8\times2^{6} = 1024 + 512 = 1536$ AND-gates.

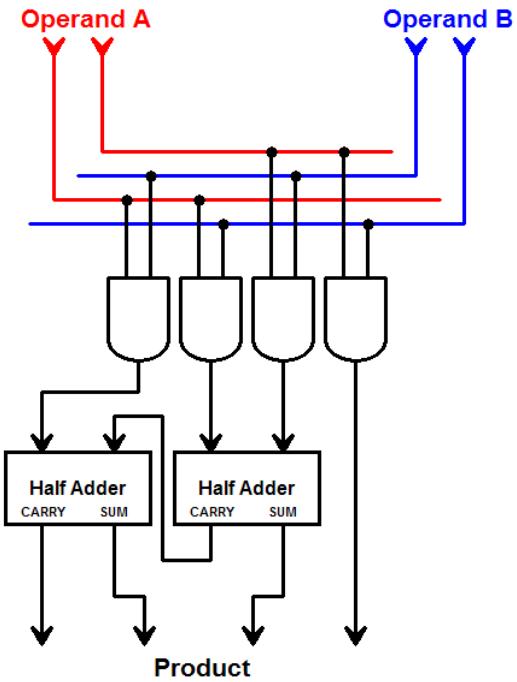
The far end of the spectrum has a single word line, no demultiplexer, and all of the hardware load in the demultiplexers. Each of the eight demultiplexers now contains 65536 AND-gates; more than half a million gates in total. As you can see, somewhere between the two extremes is the "sweet spot" that minimizes the total number of gates (a separate problem is to minimize the number of gate inputs, which may return a slightly different answer).

Multiplication

In binary the process of multiplication is very similar to the process in decimal. Each digit of one operand is multiplied by the other operand to form a partial product. The partial products are added together to generate the final answer. The process of multiplying two 2-digit (or 2-bit) numbers together is shown below:

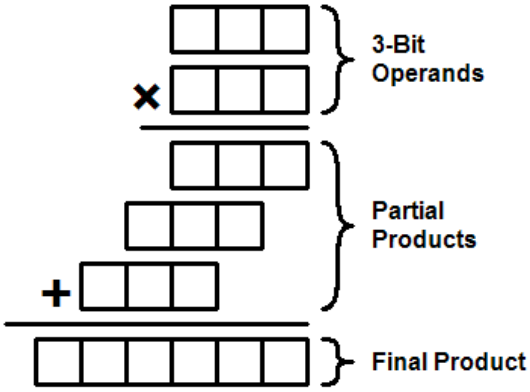


In binary each partial product is generated by multiplying an operand by either 0 or 1, resulting in either zero or a copy of the operand. This is the same behavior as a group of AND-gates. One 2-input AND-gate is used for each pair of bits, then the bits are added together with half-adders and full-adders to generate the final product. The hardware for multiplying two 2-bit numbers together is shown below:

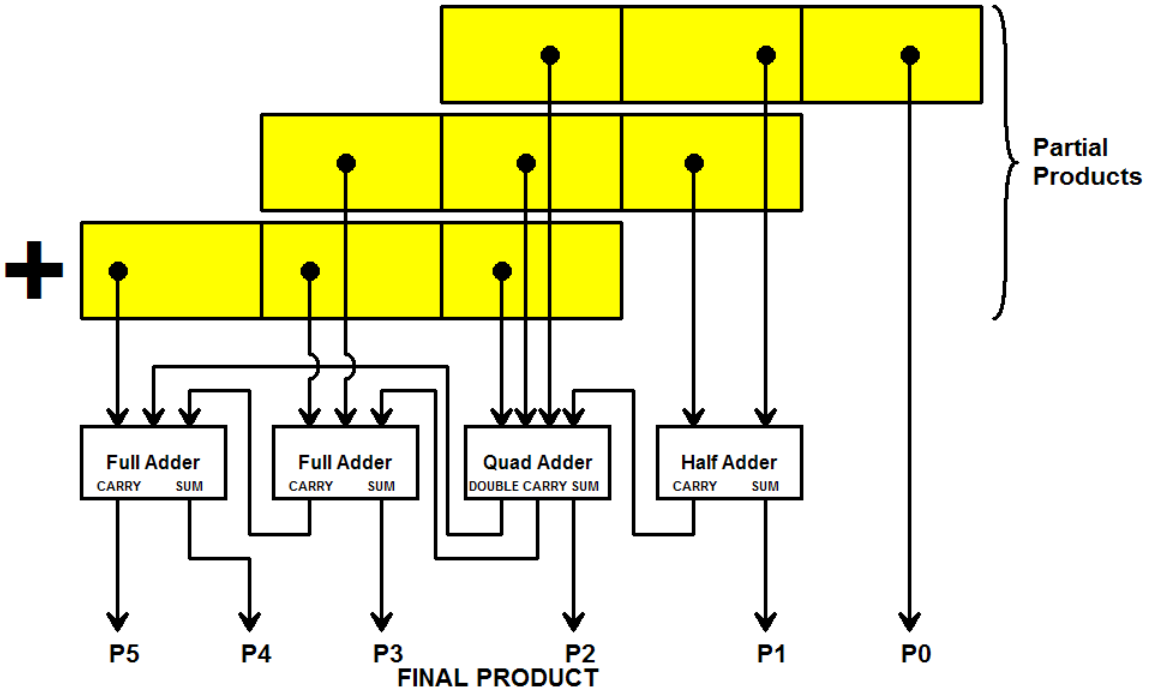


To add the partial products the circuit requires only a pair of half-adders, the leftmost accepting a carry out of the rightmost. Notice that the rightmost partial product bit appears in the answer directly; this will be the case for the product of two numbers of any length.

Similarly, multiplying two 3-bit numbers together requires the addition of three partial products. In this case, however, the addition circuitry for the partial products is significantly more complex than in the case of 2-bit numbers.

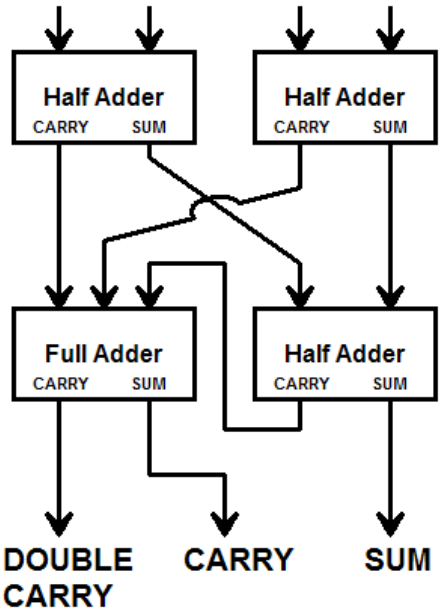


The diagram below shows the addition block of the partial products. As before, the rightmost partial product bit drops down to the answer directly, and the next bits are added with a half adder. The carry out of that half adder goes into the column with three partial product bits, requiring addition of four bits in a special "quad adder" circuit.

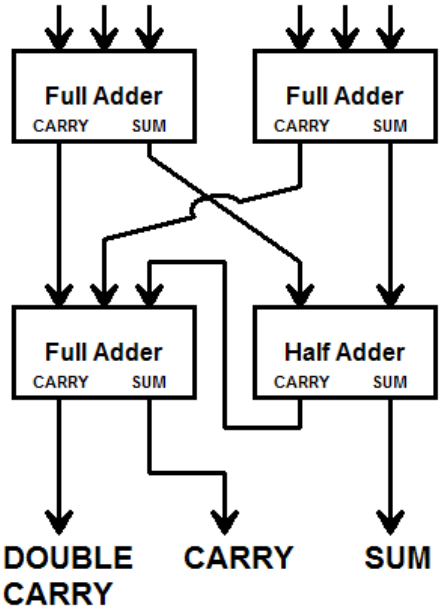


Notice that the quad adder circuit returns a 3-bit sum, emitting the sum bit to the answer, a carry bit to the next stage of partial products, and a *double-carry* to the stage beyond that.

The quad adder circuit is composed of half adders and full adders, as shown in the circuit below:



By extending this process even further, we will find that the adders for the partial product columns get more and more complicated. We can extend the quad adder to a "quint adder" by replacing one of the input half adders with a full adder, and then extend it again to a "hex adder" by replacing both of them, as shown below:



Increasing the number of bits being multiplied, and being forced thereby to deal with the corresponding complexity of the partial product adders, indicates that better methods for multiplication must be developed.