Dang Tran
Riley Guidry
Alberto Rosas
CMPE 480
6 March 2024

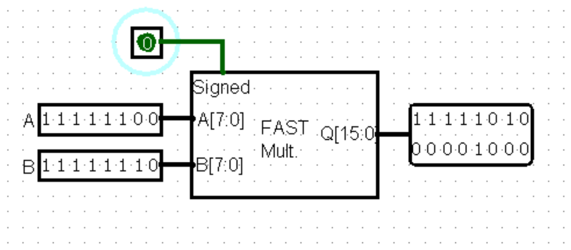
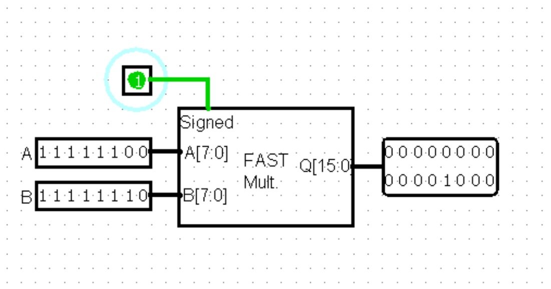
<u>**Fast Multiplier**</u>

Introduction

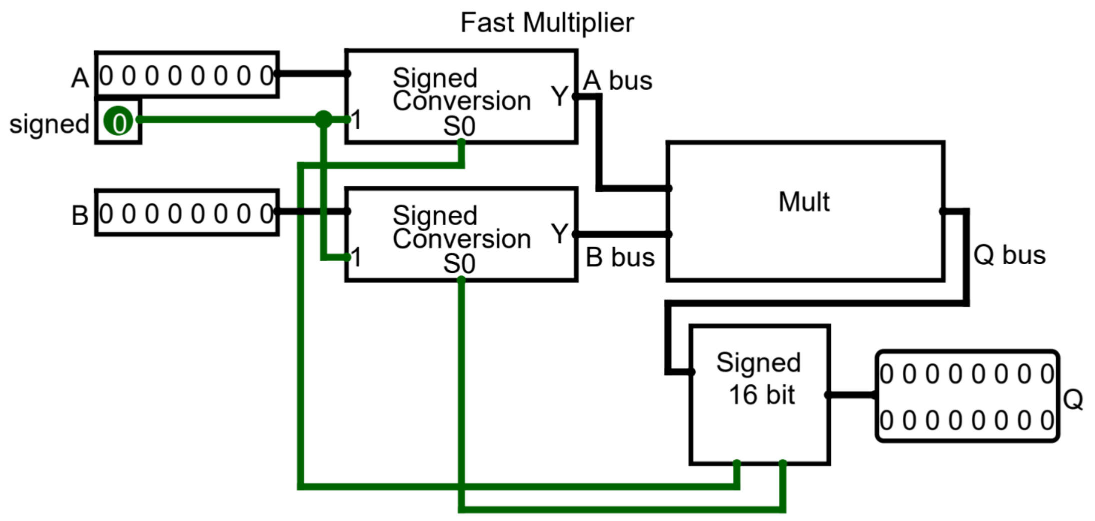
Computer chips were made to perform mathematical calculations for the ease of humans, and they were built to do it fast. So it is an area of intrigue for Computer Engineers to consider how best to do an operation like multiplication. Because multiplication is repeated addition, it can be considerably slower than any of the other operations a CPU may perform, up to 32 times for a standard MIPS processor if one addition is performed every clock cycle. A much faster method of multiplication is the Wallace tree, or a flow-through multiplier, which completes the whole process in one clock cycle, "flowing through" the whole system like an unmitigated waterfall.

Specification

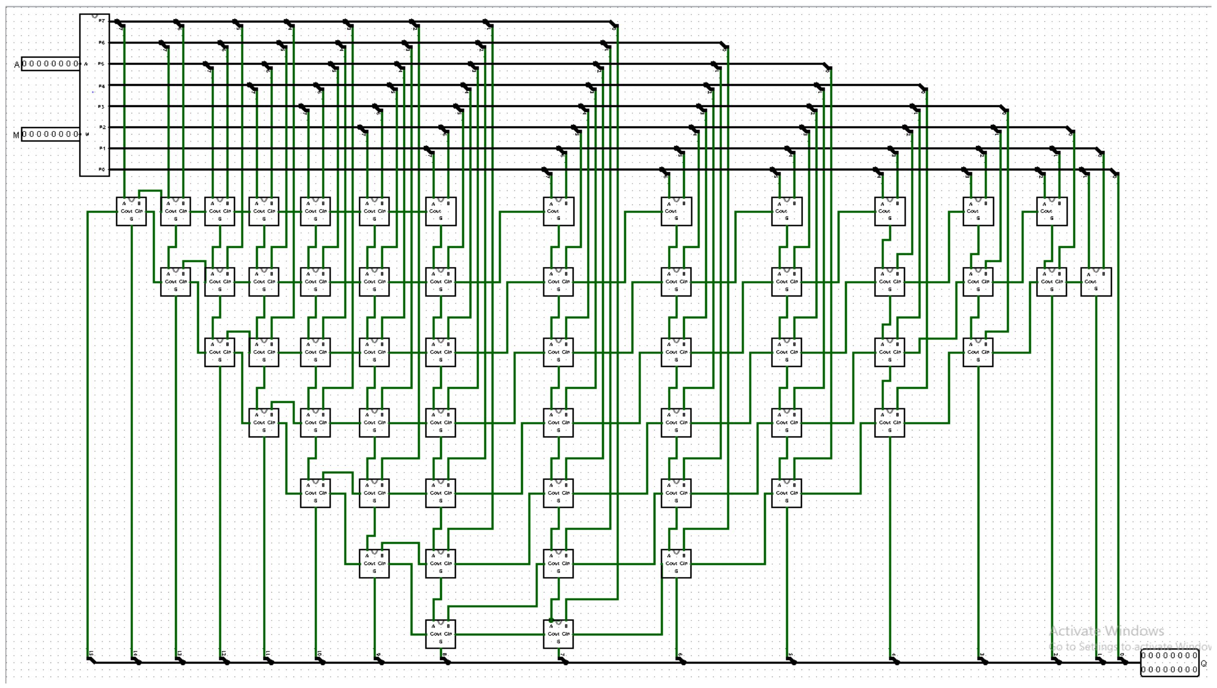
Our goal was to implement a Wallace tree multiplier in Magic, the VLSI software. For simplicity, we wanted our multiplier to multiply two eight-bit numbers together, to have an output of a maximum of 16 bits. We also wanted to control whether the multiplier understood the sign of an input, in two's complement form, or if it treated the two inputs as unsigned integers. Because a Wallace tree performs solely unsigned multiplication, we need to manipulate the data before it enters the tree, and then manipulate the data that comes out of it depending on the state of the inputs and if we are doing signed/unsigned multiplication.

Top-Level Block
 0 Signed A 1 1 1 1 1 1 0 0 B 1 1 1 1 1 1 1 0 A[7:0] B[7:0] FAST Mult. Q[15:0] 1 1 1 1 1 0 1 0 0 0 0 0 1 0 0 0  1 Signed A 1 1 1 1 1 1 0 0 B 1 1 1 1 1 1 1 0 A[7:0] B[7:0] FAST Mult. Q[15:0] 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0
A: 8-bit input (Multiplicand) *B: 8-bit input (Multiplier)* *Signed: 1-bit input where 0 represents unsigned multiplication and 1 is signed multiplication* *Q: 16-bit Product*

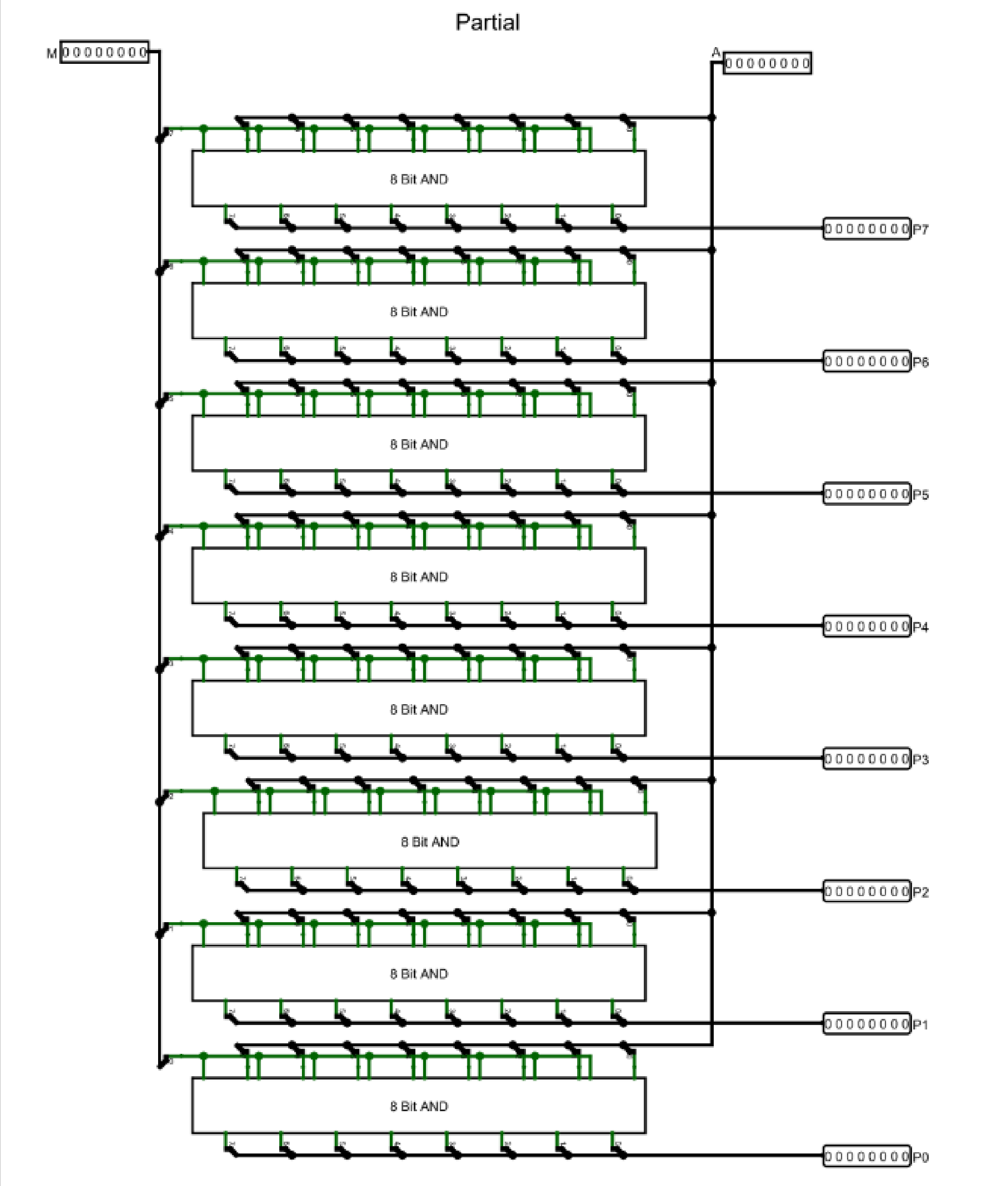
This is the final top-level schematic of our multiplier. The two 8-bit inputs A and B are multiplied together to produce Q, and a “signed” input determines if the block reads those inputs assigned or not. Inside the block is shown below.

Logisim Schematic
 Fast Multiplier A 0 0 0 0 0 0 0 0 signed 0 B 0 0 0 0 0 0 0 0 Signed Conversion Y 1 S0 A bus Signed Conversion Y 1 S0 B bus Mult Q bus Signed 16 bit 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 Q
Block Descriptions
This schematic above controls the signed/unsigned output of the multiplier, and ensures that no mathematical errors are made. The bottom schematic is the Wallace tree multiplier itself. Two 8-bit numbers are fed into an ANDer, where every bit of one input is ANDed with every bit of the other, to produce 64 results, separated into the 8 black busses above the tree of half and full adders. In multiplication, once all the multiplying of digits is finished, some addition needs to be done to the equally positioned results of each multiplication. Thus, the results of our ANDer are added according to their “weights.” The weight of a result is the addition of the position of the two bits that produced the result. For example, the weight of the result of (A[0] AND B[7]) is 0 + 7 = 7, and the weight of (A[2] AND B[3]) is 2 + 3 = 5. When two equally weighted results are added, their sum remains the same weight and propagates down the tree, while carry signals are weighted one higher than the signals that produced it. Once every signal of the same weight has been added, along with the carry signals from the weight below, the sums can be grouped into a final, unsigned, 16-bit output.

The multiplier cannot multiply a “negative” number, so if the leading bit is one, signaling it is negative is two’s complement, and we are doing a signed multiplication, then the input is inverted in two’s complement before entering the multiplier. In a signed multiplication, neither input will be negative entering the tree. The result is then kept the same or negated depending on the state of the two inputs: if one of them is negative, then the result will be re-inverted so that a negative time a positive turns out negative. But if neither or both of them are negative, the result remains positive.

Multiplier

Block Descriptions
The schematic below is the circuit that produces the 64 output results of ANDing every bit of the M input with every bit of the A input. So, the bit M[0] is ANDed with A[0], A[1], A[2], …, and A[7], and the bit M[1] is ANDed with A[0], A[1], A[2], …, and A[7], etc.

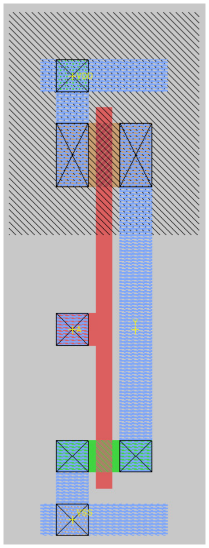
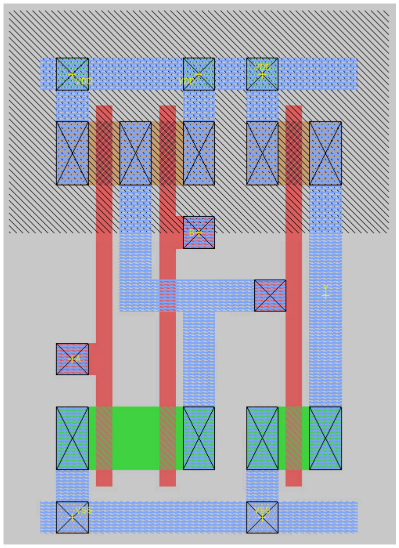
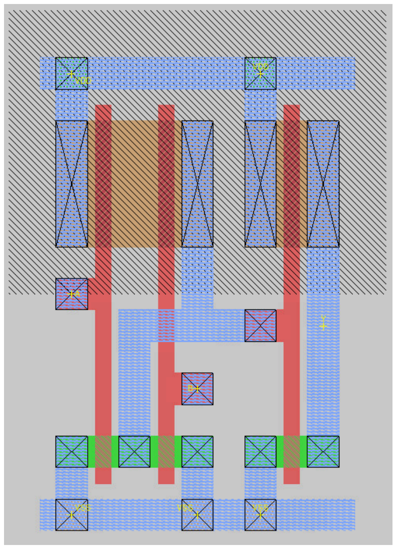
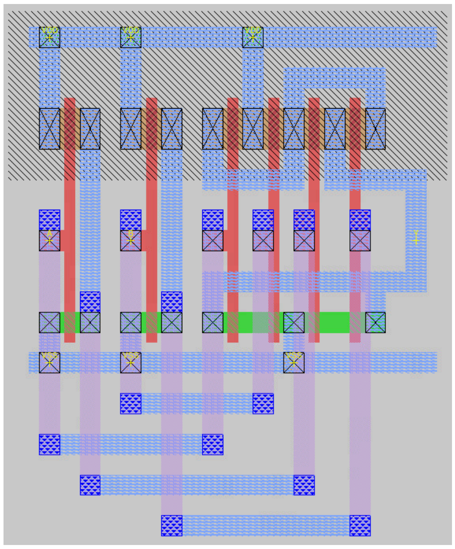
Above is the schematic for the Wallace tree. A Wallace tree takes advantage of the nature of binary multiplication, in that a multiplication between two bits is merely an ANDing of them, and the results of those ANDings need to be added according to their "weight." The weight of a product is the addition of the position of the bits that produced it. For example, P0_1 is the product of (A0 AND B1) and its weight is (1 + 0 = 1). And P3_2 is the product of (A3 and B2) and its weight is (3 + 2 = 5). Products of the same weight are added to produce a sum bit, which is considered the same weight as its parents, and a carry bit, which is one weight higher than its parents. The reason products of the same weight need to be added is made more obvious in the image below.

Partial Product
 Partial M 00000000 A 00000000 8 Bit AND 00000000 P7 00000000 P6 00000000 P5 00000000 P4 00000000 P3 00000000 P2 00000000 P1 00000000 P0

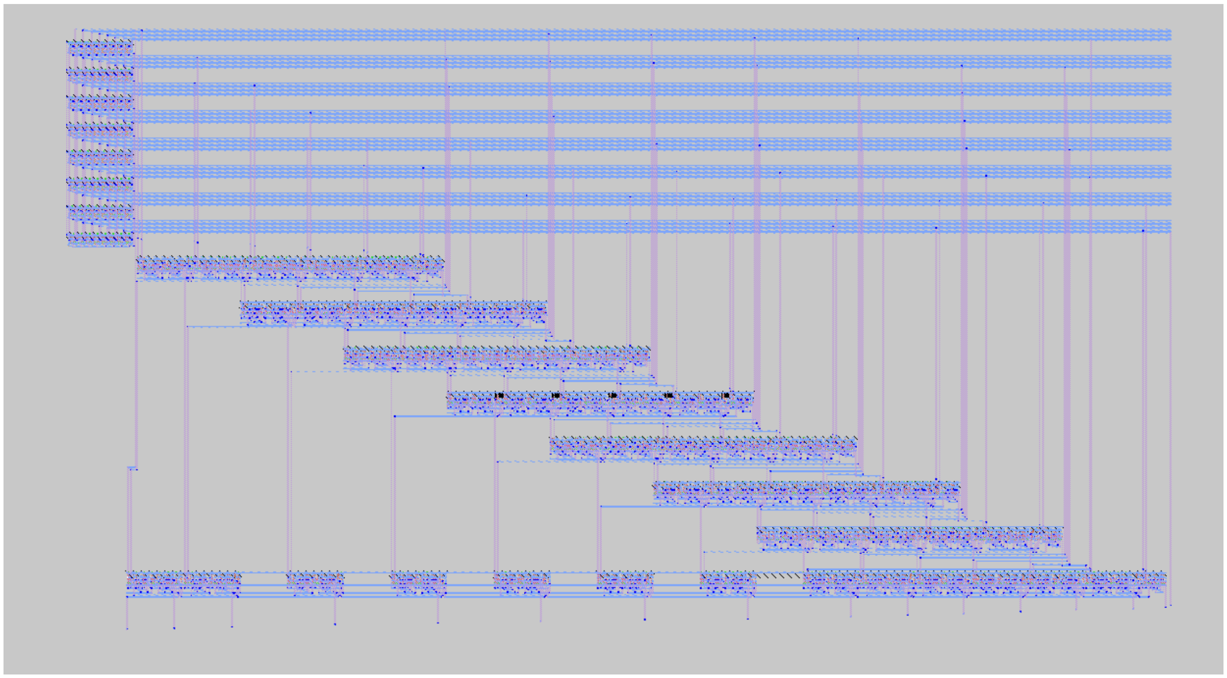
This is the array of AND gates mentioned previously.

Implementation (magic schematics go under here)

To realize this idea fully, we used Magic, the VLSI simulation software, to lay out the chemical makeup of a microchip that operated this way. We start by making the basic gates that constitute the fundamental building blocks of the massive schematic this will eventually turn into.

Basic Logic Gates in Magic	
 INV	 AND-2
 OR-2	 XOR-2

Multiplier



This is what it looks like when the AND signals are pulled out into the ADDers according to their weights. If you look above at Logisim schematics, the similarities in structure are obvious.