sends that around the loop to the partial-sum register again. In this way, the multiplication proceeds by successive addition of selected shifted multiples, and a breakdown of the problem will then appear as follows:

Multiplication	Partial Sum at Cycle
1101	
× 1011	
1101	2
+ 1101	
= 100111	3
+ 0000	
= 100111	4
+ 1101	
= 10001111	5

We must now get the final answer 10001111 out of the circuit into some other part of the machine where we can make use of it. We use FLIP-FLOP 3, operating it with a "readout timing pulse from the time selector" on the left side as follows:

Cycle	Number
1	00000000
2	00000000
3	00000000
4	00000000
5	00000001
6	00000000

together with a reset pulse on the right side of the flip-flop will then be as follows:

Cycle	Number
1	11111111
2	00000000
3	00000000
4	00000000
5	00000000
6	11111111

Here is where—and why—we need a cycle 6 in the multiplier circuit: to be certain that no stray pulses come out on the output line after we have received the product. The output of FLIP-FLOP 3 accordingly will then appear as follows:

Cycle	Number
1	00000000
2	00000000
3	00000000
4	00000000
5	11111111
6	00000000

This is the input on one side of the AND circuit 3. The other input coming from the adder is:

Cycle	Number
1	00000000
2	00001101
3	00100111
4	00100111
5	10001111
6	00000000

and as a result it may be seen the output is:

Cycle	Number
1	00000000
2	00000000
3	00000000
4	00000000
5	10001111
6	00000000

In order to clear the multiplicand, the multiplier, and the partial-sum registers, the EXCEPT circuits 3, 4, and 5 are used. The input to all of them comes in on line T5, and is in the usual listing, as follows, and after this the multiplicand, the multiplier and partial-sum registers are ready for new signals.

Cycle	Number
1	11111111
2	00000000
3	00000000
4	00000000
5	11111111
6	11111111

In the same way, to reset the flip-flops, the input of standard pulses from the time-pulse selector is as follows:

Cycle	Flip-Flop 1 and 2	Flip-Flop 3
1	10000000	11111111
2	10000000	00000000
3	10000000	00000000
4	10000000	00000000
5	10000000	00000000
6	10000000	11111111

The circuit is, of course, a block diagram. To convert it into working hardware, a good deal of attention will have to be given to the shape of pulses, the precise detailed timing of them, microsecond delays due to the length of cables along which they are traveling, etc. Descriptions or circuits of the components inside the blocks were given last month.

It should be emphasized once more that there are many ways in which the circuit can be improved, or otherwise designed. Some improvements will doubtless occur to readers of this article.

For example, F-F1, E1, and E2 can be avoided if we give up optional storage of the multiplicand unchanged in register D1; that is, if we make sure that the multiplicand and the multiplier do come in simultaneously to the multiplying circuit. Actually, in a serial computer it will be more convenient to have the multiplicand come in at one time and wait in the multiplicand register until the multiplier is ready and comes in at a later time. The circuit in Fig. 1 has provided for this second more convenient scheme.

Acknowledgment is made to Henry W. Schrimpf for many of the features of the multiplying circuit in this article.

Division

If we replace the adder by a subtractor, and put in some kind of circuit for comparing the divisor with the partial remainder from successive subtraction, and in still other ways modify the circuit of Fig. 1 so that we can use the method of dividing that was illustrated in Part IV (January, 1951) of this series for a relay computer, then we can design a division circuit for binary numbers.

But divisions do not occur as often as multiplications. Studies made some time ago in the Harvard Computation Laboratory indicated that a division occurred once for about every fourteen multiplications. So why spend money on equipment for division, which you may use less than 8% of the time, if we can get division some other way?

You can get the result of dividing the number M by the number K if you multiply M by the *reciprocal* of K, which is 1/K. And you can get the reciprocal 1/K by a process using multiplication. First, find some kind of reasonable first approximation X_1; second, use the following formula over and over:

$$X_n = X_{n-1} \times (2 - KX_{n-1}).$$

It can be shown that a reasonable first approximation X_1 is any number between 0 and 2/K.

For example, suppose that we want to find 1/3, which we know is equal to .33333 to five decimals, and suppose that we start off with a guessed first approximation .5. Then:

$X_1 = .5$ (guess)
$X_2 = .5\ [2 - 3(.5)\ ] = .25$
$X_3 = .25[2 - 3(.25)\ ] = .3125$
$X_4 = .3125[2 - 3(.3125)\ ] = .33203$
$X_5 = .33203[2 - 3(.33203)\ ] = .33333$

Thus we can get a fine result with just a few steps in multiplication. We can have our computer carry out division by programming our computer. We yield time; we save equipment. Over and over again this kind of compromise occurs in computer design. Division by successive multiplication, with special circuits for sensing good first approximations (which saved on the number of multiplications necessary), was the process built into the Harvard Mark II computer, now at the Naval Proving Ground, Dahlgren, Va.

This kind of scheme for use in automatic computers was first proposed, we believe, by Prof. Howard H. Aiken of Harvard.

(continued next month)