You can fit a fast and compact digital multiplier into a field-programmable gate array (FPGA). The technique involves using small look-up tables to find partial products and then adding the partial products. The key factor for making the design compact, fast, and easy to implement is to make the look-up tables for the partial products as small as possible. This multiplier can quickly multiply a number by a constant; changing the constant momentarily takes the multiplier off-line.

The 8×8-bit multiplier in Fig 1 first develops all possible partial products for the constant operand, A_OPERAND, and loads those partial products into two 16×12-bit static RAM look-up tables. Preloading the look-up tables with only those partial products that can be obtained from the present value of A_OPERAND to the look-up table’s address lines, greatly reduces the size of the look-up table. These partial products are:

\[0, 1 \times A_{\text{OPERAND}}, 2 \times A_{\text{OPERAND}}, ..., 15 \times A_{\text{OPERAND}}\]

These 16 partial products must be recalculated and reloaded for each new value of A_OPERAND. The accumulator, address counter, and small state machine (which accesses each location of the memory and stores a new partial product) together perform this modification. Clearly, the multiplier is unavailable from the time it receives a new value of A_OPERAND until look-up-table reloading is complete.

Next, the circuit splits the variable operand, X_OPERAND, into two 4-bit sections, using these sections as addresses into the look-up tables. These look-ups have the effect of multiplying each of the two sections of X_OPERAND by A_OPERAND and form two partial products. The circuit then combines the partial products by addition—and some judicious bit-weighting.

The adder requires 16-bit operands. The 12-bit partial products need to be converted to 16-bit values with the proper weighting. The circuit pads out the partial product from the low-order nibble out with four additional high-end zeros (weighting factor of 1). The partial product from the high-order nibble gets four additional low-end zeros (weighting factor of 16).