A Fast Constant Coefficient Multiplier for the XC6200

Tom Kean, Bernie New and Bob Slous
Xilinx Inc.

Abstract. We discuss the design of a high performance constant coefficient multiplier on the Xilinx XC6200 FPGA. The dynamic reconfiguration capabilities of the device are used to allow the constant coefficient to be rapidly changed. The design also provides better performance and density than similar multipliers on state of the art conventional FPGA's which require complete reconfiguration to change the coefficient.

Introduction

The Xilinx XC6200 [1] is the first commercial FPGA specifically designed to work in close cooperation with a microprocessor. To the microprocessor the XC6200 appears as a block of RAM, configuration data and registers within the user design appear within this memory space. The data bus width is programmably selectable to 8,16 or 32 bits. Additional features support writing many words of configuration memory with the same data simultaneously (to reduce configuration time) and grouping non-adjacent registers within the user design into a single word for transfer to and from the processor. XC6200 application development and debug is supported by the XACTStep Series 6000 CAD software package and by a PCI bus resident development system [2]. The XC6200 architecure is derived from the Algotronix CAL technology which Xilinx aquired in 1992 [3] [4].

Constant coefficient multiplication is a common function in many Digital Signal Processing (DSP) applications such as Finite Impulse Response (FIR) filtering. In this case the constant coefficients represent filter parameters and are changed much less frequently than the input data values. The multiplier described here multiplies an 8-bit input number by a constant 8-bit coefficient to get a 16-bit result, the numbers are unsigned.

An efficient technique to implement fixed coefficient multipliers of FPGA's using Look Up Table (LUT) based FPGA's was developed by Xilinx [5]. This relies on lookup tables, rather than a network of adders, to perform most of the multiplication. For example, there are 16 possible results when a four bit number is multiplied by an 8 bit fixed number (because there are 16 different four bit numbers). Thus a four bit variable times eight bit constant mu ltiplier can be implemented by a 16 entry lookup table. Each entry must be 12 bits, the width of the largest possible output. An 8-bit by 8-bit constant multiplier can be built using two of these 4-bit by 8-bit constant multipliers in the configuration shown in figure 1. Figure 2 explains the simple mathematics behind this arrangement.

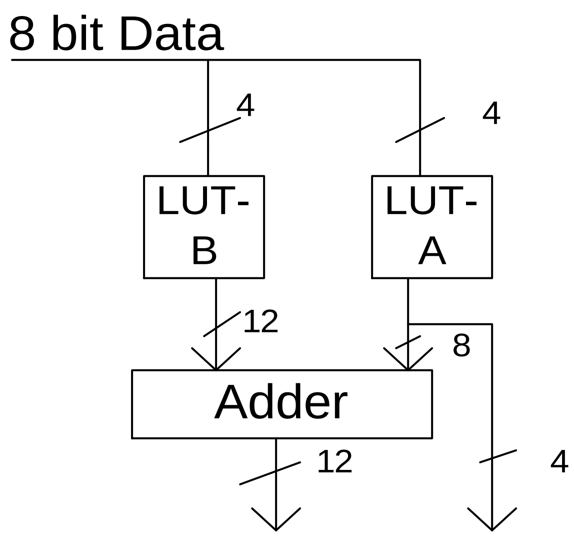


Figure 1: LUT Based Multiplier

Physical Implementation

Figure 3 shows the hardware components required to implement the 8x8-bit multiplier. Pipeline registers have been added to increase the throughput. Note that not only the adder but also the LUT's are pipelined.

Unlike XC4000 FPGA's which use LUT's to implement logic functions XC6200's have a simpler function unit built around a 2:1 multiplexer. This function unit can implement a register and either a 2:1 multiplexer or any one of the 16 logical functions of two input variables. The ability to implement a register and a combinational logic function makes pipelining inexpensive in this technology. Figure 4 shows a naive approach to implementing a 4-input lookup table using 2:1 multiplexers and registers. Figure 5 shows how this can be collapsed by encoding the leaf nodes of the tree into gate functions. It takes four bits of configuration memory to encode the truth table of a 2-input logic gate so what we are doing is moving the 16 bits of memory provided by user registers in figure 4 into bits of configuration memory.

It is possible to obtain a very regular layout for the individual 4-input LUT's as a single row of 8 cells. These slices can be stacked vertically to form the 12-bit output, 4-bit input LUT's required. The fact that the layout is so regular makes it easy to determine which parts of the chip need to be reconfigured at run time. The layout of the multiplier LUT is shown in figure 6. Note that the pipelining scheme used requires that some slices, corresponding to more significant bits, have additional registers.

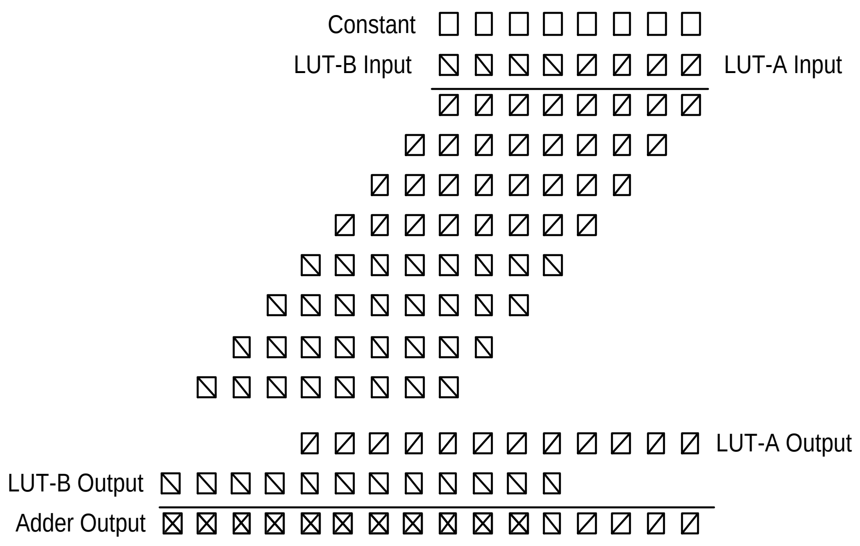


Figure 2: Maths of LUT Based Multiplier

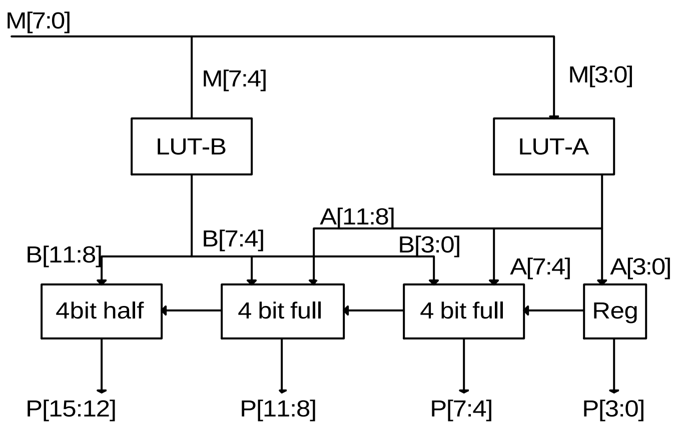


Figure 3. Multiplier Architecture

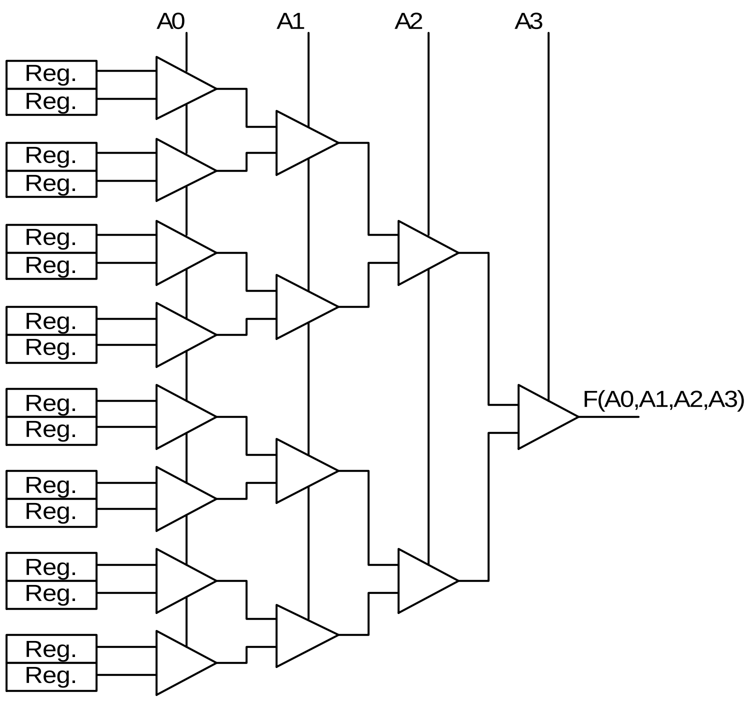
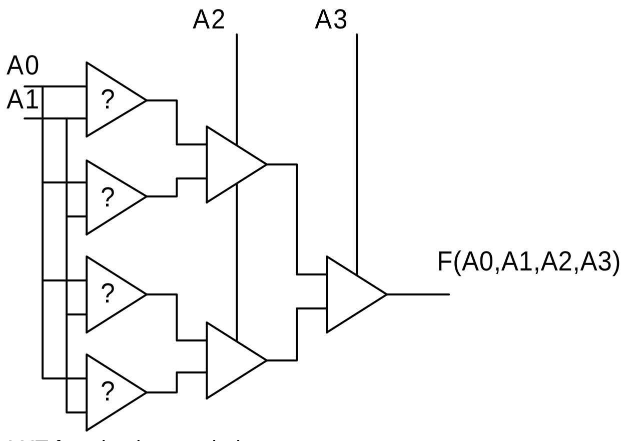


Figure 4: Building LUT's from Muxes



The LUT function is encoded into the functions implemented by these gates

Figure 5: Optimised LUT Construction

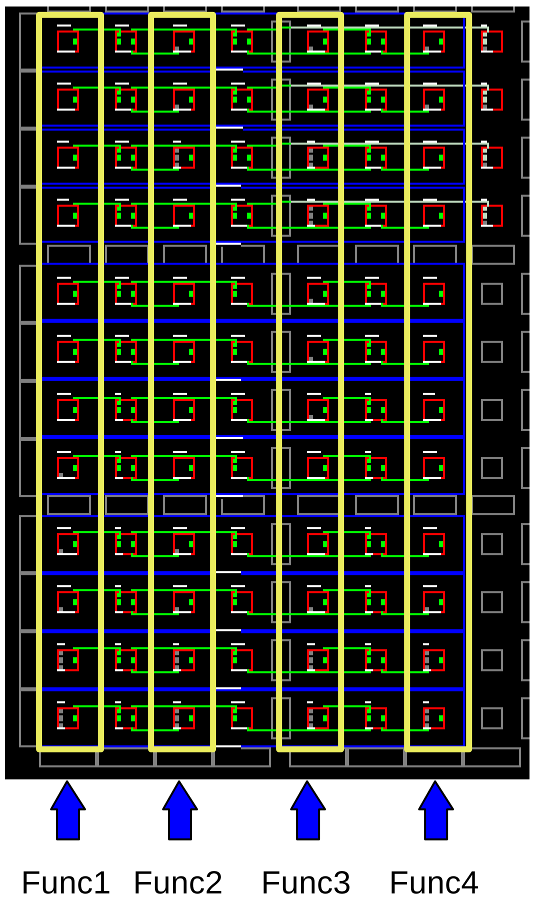


Figure 6: LUT layout on XC6200

Changing Coefficients.

The XC6216 memory map is designed so that the control memory for functionally related resources on the chip occur in logically adjacent bits and words of control memory. The bits which control the gate function of a cell occur in the same byte of memory. With a 32-bit data bus this byte in four vertically adjacent cells can be written simultaneously. In figure 6 the vertical columns labelled Func1 through Func 4 contain the gates whose functions are changed according to the lokup table - each horizontal slice implements one 4-input LUT.

There are 12x4=48 cells in each multiplier LUT which may need reconfigured if the coefficient changes and 2 LUT's for a total of 96 cells. Thus 24 write cycles are required to completely reprogram the LUT's or 960ns with a 50MHz clock.

Multiplier Performance

The multiplier has been simulated at 75MHz using the -2 speed grade. The equivalent design on the XC4000E -3 grade runs at 65MHz (note that the speed grade numbering system is different for the two families so this is a fair comparison). The cost is around 25CLB's in the XC4000E and 280 used cells in the XC6200. Normalising for silicon area approximately 10.6 XC6200 cells take the same area on the same process as 1 XC4000E CLB: so the area is almost equivalent. The big difference between the designs is that the XC6200 allows the coefficients to be changed in less than 1us where the XC4000E would require that the entire chip be reconfigured which takes several ms.

XC4000 devices can support changing of LUT configuration data without reconfiguring the chip by using the feature which allows user logic to address LUT RAM. If this technique is used, however, significant additional logic must be added to the user design to control the LUT RAM and to interface with the external source of data (e.g. a microprocessor). Routing resources will be tied up routing between IOB's and the LUT RAM control logic. This circuitry is supplied 'for free' by the dedicated processor interface on the XC6200.

Increasing Density

The design presented above is laid out to preserve regularity in order to make dynamic reconfiguration to change coefficients fast and efficient. If longer reconfiguration times can be tolerated it is possible to substantially reduce area by co-optimising the logic of the groups of 12 LUT's which share the same 4 inputs together. A simple way of doing this is to notice that only 5 unique functions of the first two input variables must be generated. There are 16 possible functions but the second 8 are the logical inverses of the first and the inverse can be calculated by inverting the corresponding multiplexer input in the second level of the tree. Of the remaining 8 functions 3 are trivial (ZERO,A0 and A1) and do not require a logic gate. Thus we need only calculate the 5 non trivial functions and route these 5 functions and the two input variables to the 12 sets of muxes to build the LUT. This requires 3*12+5=41 cells rather than the 7*12=84 cells of the previous design, at the expense of having a different routing pattern for each possible LUT. This will almost half the size of the XC6216 design while still allowing coefficients to be changed much faster than the XC4000E. Further reduction in gate count to 30 gates for the worst case coefficient is possible with more complex logic optimisation algorithms.

Summary

This paper has shown that the XC6200 fine grained multiplexer based FPGA's can implement LUT based functions almost as efficiently as state of the art LUT based FPGA's. In fact the fine grained FPGA can be more efficient when several lookup tables have the same set of input variables, because it allows common logic used by all LUT's to be implemented only once.

XC6200 devices implement the adder more efficiently than coarse grain devices which allows the fine grain implementation of the multiplier to take almost identical area to the coarse grain implementation even when common LUT logic is not optimised. The LUT implementation techniques described here, and the results reported are specific to XC6200. The available fine grain FPGA's have significantly different functional and routing resources and each may require a different approach to LUT implementation.

In addition the ability of the XC6200 architecure to dynamically reconfigure small portions of the device very quickly makes the look up table based technique much more attractive since coefficients or other parameters stored in the lookup tables can be changed rapidly. The configuration values required for a particular coefficient can be calculated rapidly on the fly from input data by a microprocessor - no complex CAD tools are required.

References

[1] Xilinx Inc, "XC6200 FPGA Family Advanced Product Description", Available from Xilinx Inc. 2100 Logic Drive San Jose CA.

[2] Wayne Luk and Nabeel Shirazi, "Modelling and Optimising Run Time Reconfigurable Systems", Proc. IEEE Symposium on FPGA's for Custom Computing Machines, Napa CA 1996.

[3] Tom Kean, "Configurable Logic: A Dynamically Programmable Cellular Architecture and its VLSI Implementation" Phd Thesis CST-62-89, University of Edinburgh, Dept. Computer Science.

[4] Tom Kean and John Gray, "Configurable Hardware: A New Paradigm for Computation", Advanced Research in VLSI, Proc. Decennial Caltech Conference, MIT Press 1989.

[5] Ken Chapman, "Fast Integer Multipiers fit in FPGA's", EDN 1993 Design Idea Winner, EDN May 12th 1994.