Designing High-Performance Memories and Multipliers

It is easy to create efficient, high-performance designs using the Xilinx CORE Generator™

by Krista M. Marks
Engineering Manager, IP Solutions Division, Xilinx Inc.
krista.marks@xilinx.com

The Virtex-II architecture offers exciting new design possibilities because it includes numerous high-performance embedded memories and multipliers. These two components form the basis for many applications, appearing in a wide range of functions including digital filters, FFTs, FIFOs, serializers, encoders, and analyzers. Because memories and multipliers typically appear in the critical processing functions of an application, it is essential that they be optimized for resource utilization and performance. Because of their ubiquity, it is equally essential that they can be implemented repeatedly without significant design overhead.

Xilinx provides flexible core generation software that produces ready-to-use high-performance design solutions for the embedded memories and multipliers of the Virtex-II family. The Xilinx CORE Generator enables you to easily create optimal solutions that are specifically tailored for your specific application.

Using the CORE Generator

When you select the dual port block memory LogiCORE in CORE Generator, the interface shown in Figure 1 appears. This interface allows you to customize the Virtex-II Block SelectRAM on your targeted device and to create the required memory array. You may select memory depths as large as 1M words and word widths up to 256 bits. The initial content of the memory can be specified by a file or by a global value.

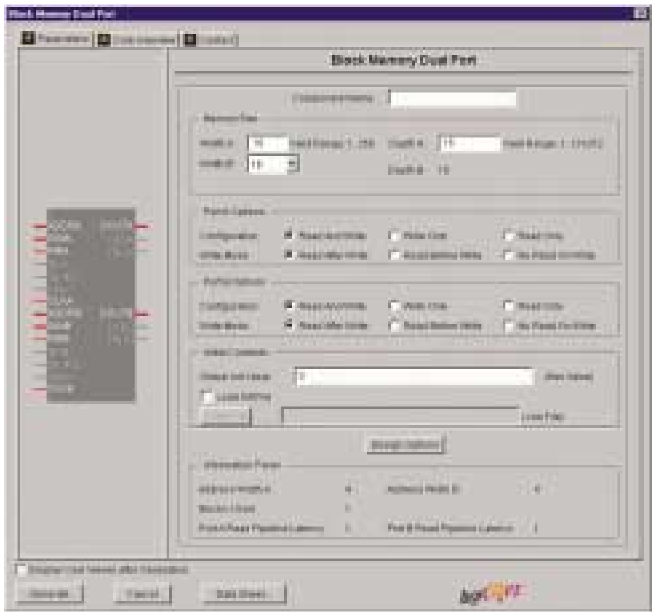

Figure 1: Parameterization window for the Dual Port Block Memory LogiCORE

You can configure the ports to have different views of the memory space and to independently obey one of the three write modes supported by the Virtex-II architecture. The Read-Before-Write mode offers the flexibility of using the output data bus during a write operation, which can increase the effective bandwidth of the block memory.

By selecting the Design Options button, another window is opened enabling further configuration choices including pipeline control and optional pins (synchronous initialization of the outputs, a clock enable, and various handshaking signals).

A similar user interface exists for multipliers, allowing you to generate a parallel or sequential multiplier implemented in either the dedicated Virtex-II multiplier fabric or in the general-purpose FPGA fabric. (A sequential or serial multiplier time-multiplexes the calculation over several clock cycles, thereby reducing the required FPGA resources in exchange for bandwidth.) The input data widths of the multiplier can be configured independently from 1 to 64 bits in width and can be signed, unsigned, or dynamically typed.

You can also create constant coefficient multipliers that can be statically or dynamically reloaded. If the constant input is dynamic, you are given the option to halt the multiplier's operation while a new constant is loaded. Options are also available to create pipelining, to vary the output width, and to include a clock enable pin.

Design Examples

To illustrate the power that lies behind the convenient interface of the Xilinx CORE Generator, it is useful to look at the designs it produces.

Consider creating a 66Kb memory configured as a 6Kx11 array using the dedicated Block SelectRAM resources. The most straightforward solution would be to divide the array in depth and to use six 1Kx18 primitives, as illustrated in Figure 2. For this solution you would need to add logic to multiplex the output of these primitives. To avoid creating this multiplexing logic, a second solution would be to partition the data bus in width, instantiate six 8Kx2 primitives, and to concatenate their output busses.

For both these solutions the memory requirement is 66Kb, but the implementation requires six SelectRAM blocks and represents a utilization of only 61% (66Kb/108Kb=61%).

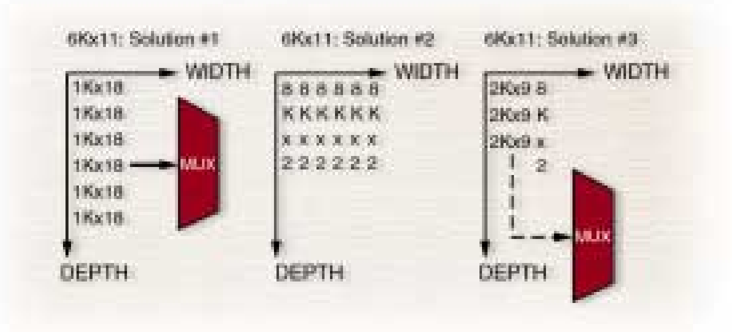

Figure 2 - Three 6Kx11 Block Memory solutions

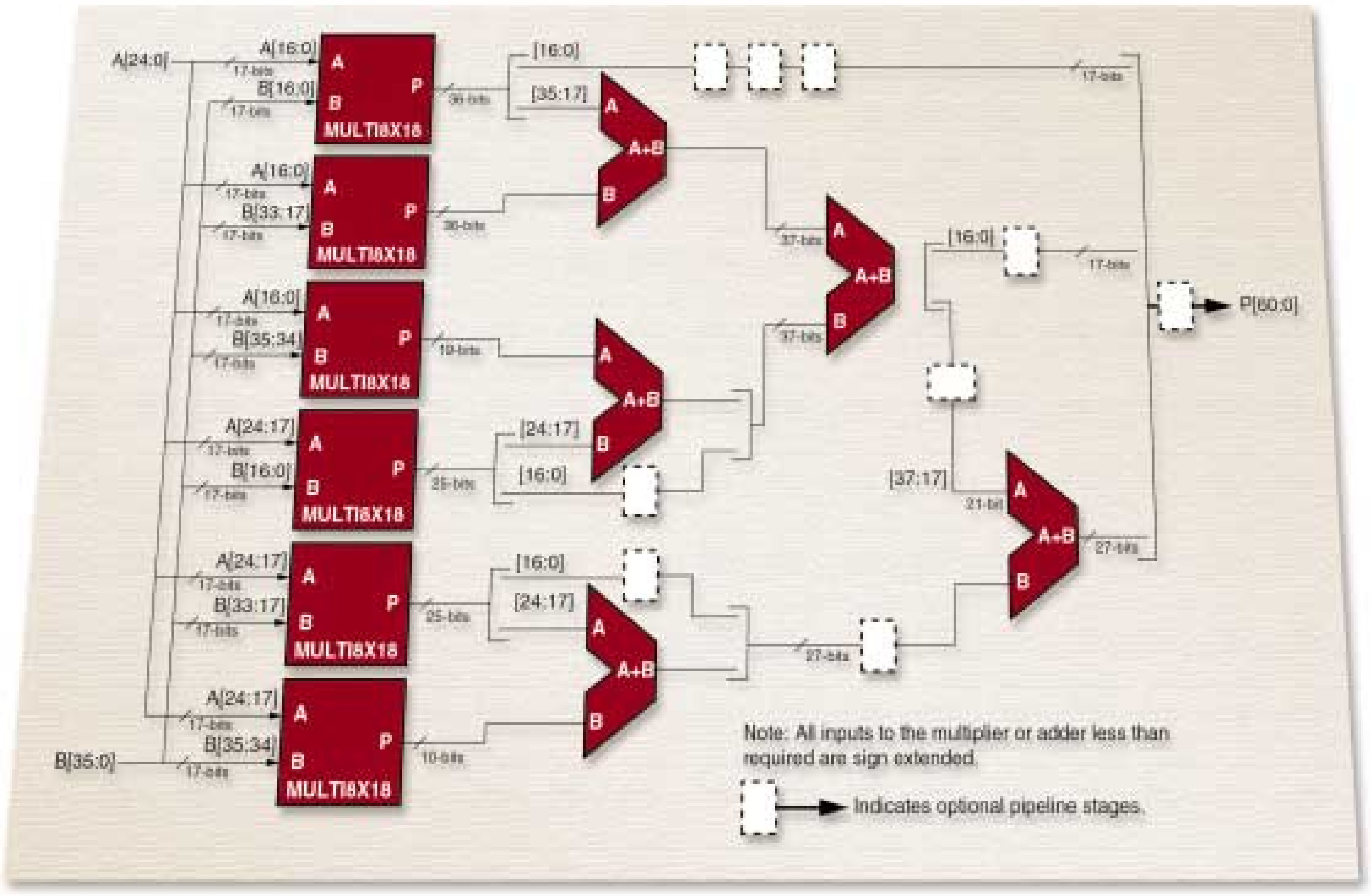

Figure 3 - Implementation produced by the Multiply Generator LogiCORE

The optimal solution, and the one implemented by the Xilinx LogiCORE, uses three 2Kx9 primitives combined with one 8Kx2 primitive. It uses four Block SelectRAMs and represents a resource utilization of 92%. In addition, the multiplexing logic is included with the generated core.

As a second scenario consider the implementation of a multiplier which has a 25-bit signed number on one input and a 35-bit unsigned number on the second input, and uses the Virtex-II multiplier fabric. The solution requires splitting the operands into slices that are no larger than the input width of the Virtex-II multiplier primitive (18-bits). The width of B is 35-bits, and an additional bit is needed to treat B as signed; the full precision product will be 61-bits.

The implementation produced by the Multiply Generator LogiCORE is shown in Figure 3. The six boxes on the left hand of the diagram represent individual 18-bit multiplier primitives, and the final output is shown as P[60:0]. Not only is this implementation the optimal solution, it automatically provides you with all the additional logic and adders trees required to implement the multiplier. Pipelining the calculation can dramatically increase multiplier throughput (pipelining registers are shown as dashed boxes in Figure 3); The core allows you to minimize or maximize the amount of pipelining. As with all LogiCORE implementations, the generated design is fully tested and verified.

Conclusion

These examples illustrate both the complexity and utility that underlie designs generated via the Xilinx CORE Generator. It provides full support for the wealth of architectural features of the Virtex-II family and provides you with the ability to quickly create solutions with fundamental building blocks. By simply invoking the CORE Generator and entering the required configuration, an optimized and verified design is created using SmartIP technology. The combination of powerful core generation technology and the new Virtex-II device features guarantees you a fast and painless time-to-market.

Virtex-II Building Blocks

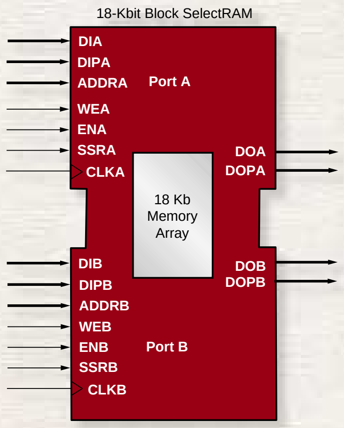

Figure 4 - True Dual-Port RAM

Each embedded Virtex-II block SelectRAM is 18Kb of True Dual-Port RAM with two fully independent access ports as illustrated in Figure 4. Each port behaves synchronously relative to its own clock input. There are two separate data out busses, one for accessing data and the other for accessing dedicated parity bits. Table 1 lists the aspect ratio of the ports available in the 18-Kbit Block SelectRAM primitives. For applications that do not require parity information, the two busses can be combined to yield larger memory widths.

WIDTH	DEPTH	ADDRESS BUS	DATA BUS	PARITY BUS
1	16,384	ADDR[13:0]	DATA[0]	N/A
2	8,192	ADDR[12:0]	DATA[1:0]	N/A
4	4,096	ADDR[11:0]	DATA[3:0]	N/A
9	2,048	ADDR[10:0]	DATA[7:0]	Parity[0]
18	1,024	ADDR[9:0]	DATA[15:0]	Parity[1:0]
36	512	ADDR[8:0]	DATA[31:0]	Parity[3:0]

Table 1 - Port configuration

Each embedded multiplier block is an 18x18 2's complement signed multiplier. The MULT18X18 primitive, illustrated in Figure 5, has two 18 bit inputs and a 36-bit product. These blocks are optimized for performance and low power consumption, and can vastly outperform an 18x18 multiplier implemented in general logic resources. These multipliers can be coupled with the block RAMs and use dedicated high-speed interconnects between the multiplier and RAM blocks, allowing for efficient multiply-accumulate filter constructs.

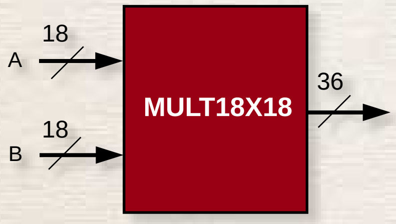

Figure 5 - MULT18X18 primitive