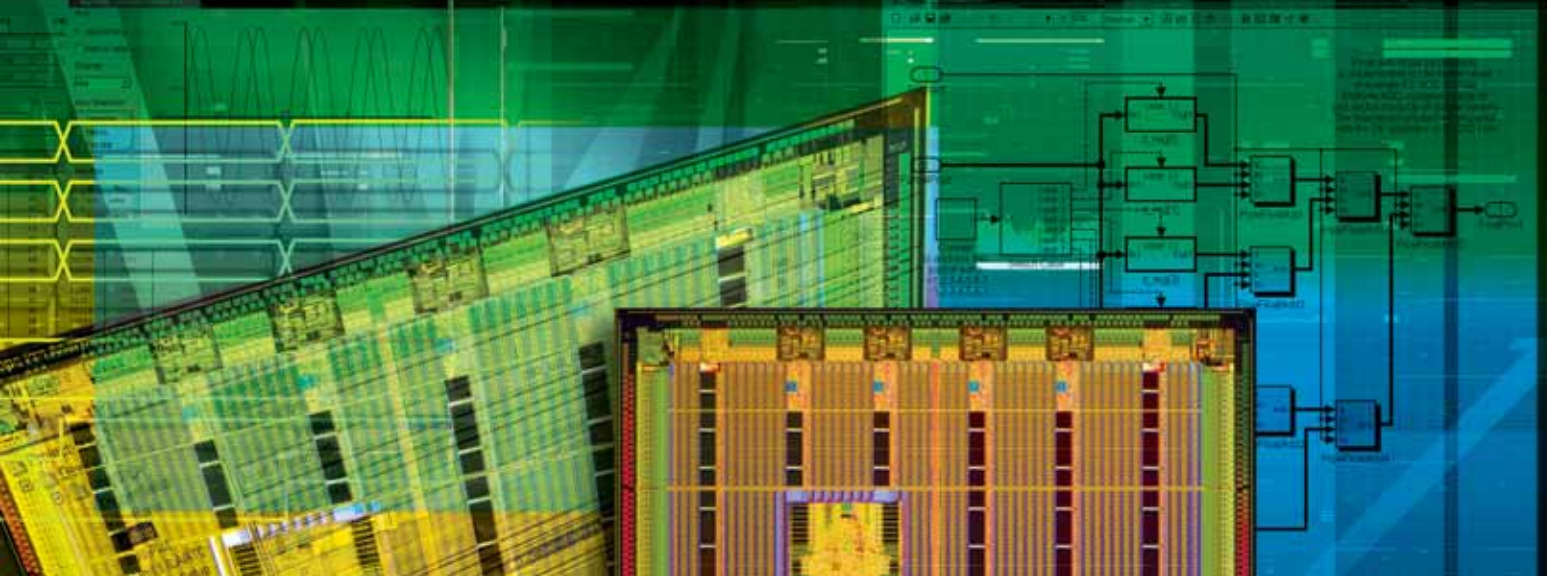


# Implementing Floating-Point DSP

Using PicoBlaze processors for high-performance, power-efficient, floating-point DSP.



by Jiří Kadlec  
DSP Researcher  
UTIA Prague, Czech Republic  
[kadlec@utia.cas.cz](mailto:kadlec@utia.cas.cz)

Stephen P.G. Chappell  
Director, Applications Engineering  
Celoxica Ltd., Abingdon, UK  
[steve.chappell@celoxica.com](mailto:steve.chappell@celoxica.com)

For developers using FPGAs for the implementation of floating-point DSP functions, one key challenge is how to decompose the computation algorithm into sequences of parallel hardware processes while efficiently managing data flow through the parallel pipelines of these processes.

In this article, we'll discuss our experiences exploring architectures with Xilinx® PicoBlaze™ controllers, and present a design strategy employing the ESL techniques of model-based and C-based design to demonstrate how you can rapidly integrate highly parameterizable DSP hardware primitives into power-efficient

high-performance implementations in Spartan™ devices.

## Hardware Acceleration and Reuse

High-performance implementations of floating-point DSP algorithms in FPGAs require single-cycle parallel memory accesses and effective use of pipelined arithmetic operators. Many common DSP vector and matrix operations can be split into batch calculations fulfilling these requirements. Our architectures comprise Xilinx PicoBlaze worker processors, each with a dedicated DSP hardware accelerator (Figure 1). Each worker can do preparatory tasks for the next batch in parallel with its hardware accelerator. Once the DSP hardware accelerator finishes the computation, it issues an interrupt to the worker. The worker's job is to combine the accelerated parts of the computation into a complete DSP algorithm.

It is ideal if you limit implementations to the batch operations of each worker starting in a block RAM, performing a rel-

atively simple sequence of pipelined operations at the maximum clock speed and returning the result(s) back to another block RAM. You can effectively map these primitives to hardware, including the complete autonomous data-flow control in hardware. You can also code the related dedicated generators of address counters and control signals in Handel-C, using several synchronized do-while loops. Simulink is effective for fast derivation of bit-exact models of the batch calculations in DSP hardware accelerators.

## Floating-Point Processor on a Single FPGA

Let's consider an architecture for the evaluation of a 1024 x 1024 vector product in 18m12 floating point. (In the format AmB, A is for the word length and B is for the number of bits in the mantissa, including the leading hidden bit representing 1.0.)

We implemented this architecture using five PicoBlaze processors on a single FPGA: one master and four simplified workers (Figure 1). The master is connect-

ed to the workers by I/O-mapped dual-ported block RAMs organized in 2,048 8-bit words. The master maintains the real-time base with 1  $\mu$ s resolution and provides RS232 user-interface functions. Each worker serves as a controller to a dedicated floating-point DSP hardware accelerator connected through three dual-ported block

RAMs organized in 1,024 18-bit words. Two block RAMs hold source vectors and one holds results data. In this case, the workers perform one quarter of the computation each, namely a 256 x 256 vector product. The DSP hardware accelerators are implemented in hardware, from block RAM data source to block RAM data sink,

using one 18m12 multiplier (FP MUL) and one 18m12 adder (FP ADD).

### Scalable, Short-Latency Floating-Point Modules

We used a newly released version of a scalable, short-latency pipelined floating-point library from Celoxica to build our DSP hardware accelerators. Table 1 considers some of the parameterizations of this FPGA vendor-independent library to the formats 18m12, 32m24, and 64m53. The library includes IEEE754 rounding, including the round to even. It provides bit-exact results to the Xilinx LogiCORE™ floating-point operators (v2.0), with latency set to approximately one half. The resulting maximum system clock is compatible with PicoBlaze and MicroBlaze™ embedded processors.

### Simulink and the DK Design Suite

Our design flow is based on the bit-exact modeling of Handel-C floating-point units in a Simulink framework, where the Handel-C is developed in the DK Design Suite combined simulation and synthesis environment. This enabled us to decompose a floating-point algorithm into a sequence of simple operations with rapid development and testing of different combinations.

### Step 1: Model in Simulink

First, we built a model of the DSP hardware accelerator in Simulink (Figure 2). The data sources and sinks in this model will be the block RAMs shared with the PicoBlaze worker in the final implementation.

Because the FPGA floating-point operations are written in cycle-accurate and bit-exact Handel-C, we benefited from a single source for both implementation and simulation. For modeling, we exported the Handel-C functions to S-functions using Celoxica's DK Design Suite. We then incorporated these into a bit-exact Simulink model. In this fast functional simulation, we use delay blocks in Simulink to model pipeline stages (see the 5-stage pipeline of the FP ADD operator and related registers in Figure 2). We used separate Simulink subsystems to model the bit-exact operation of the final "pipeline flushing," or "wind-up opera-

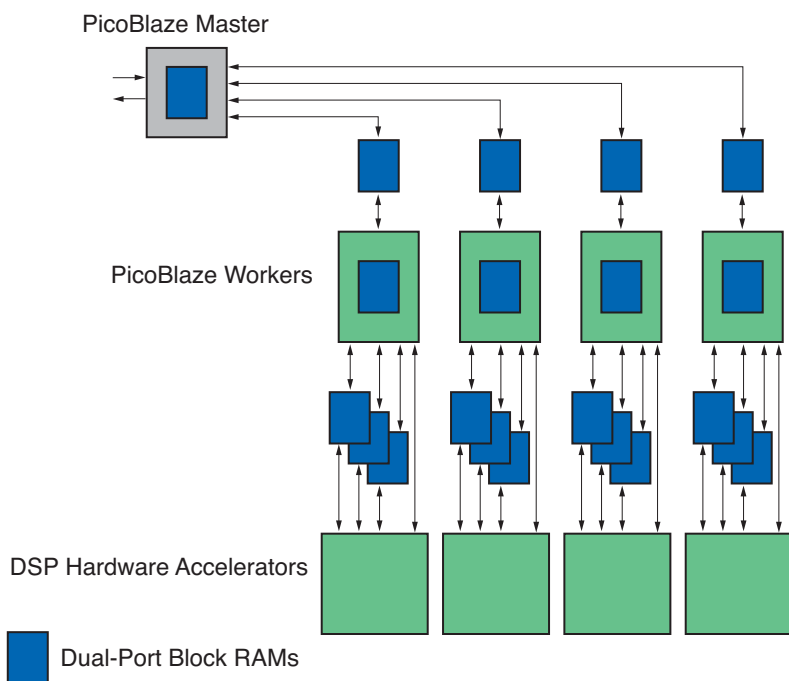


Figure 1 – PicoBlaze-based architecture for floating-point DSP. The DSP hardware accelerators are modeled and implemented using Celoxica DK.

Part:	18m12:			32m24:			64m53:		
xc2v1000-4	125 MHz			110 MHz			100 MHz		
xc3s1500L-4	84 MHz			84 MHz			72 MHz		
Pipelined:	FF	LUT	Pipe	FF	LUT	Pipe	FF	LUT	Pipe
ADD	834	793	5	1158	1290	5	1686	2007	5
MULT	639	488	3	967	626	4	2029	1256	5
F2FIXPT	581	637	4	649	744	4	808	1053	4
FIXPT2F	695	709	6	792	787	6	1008	946	6
Sequential:			Cycles			Cycles			Cycles
DIV	739	605	17	987	772	29	2119	1143	58
SQRT	766	604	16	1053	802	28	1729	1303	57

Table 1 – Used flip-flops, LUTs, pipelinellatency, and maximum clock for Celoxica floating-point modules in system implementations in the Celoxica RC200E (Virtex-II FPGA) and RC10 (Spartan-3L FPGA) boards. Modules are pipelined, with the exception of DIV and SQRT.

tion.” In this case, six partial sums have to be added by a single reused FP ADD module (Figure 3). The corresponding hardware computes the final sum of the partial sums by reconnecting the pipelined floating-point adder to different contexts for several final clock cycles.

### Step 2: Cycle-Accurate Verification

Our next stage was to create test vectors using Simulink and feed these into a bit-exact and cycle-accurate simulation of the DSP hardware accelerator in the DK Design Suite’s debugger. Once we confirmed identical results for both the DK and Simulink models, we compiled the Handel-C code to an EDIF netlist.

### Step 3: Hardware Test

We took advantage of a layered design approach by using a single communication API for data I/O functions that applies to both simulation and implementation. This allowed us to verify the DSP hardware accelerator design on real FPGA hardware by “linking” with an appropriate board support library for implementation. We can optionally insert this hardware test back into the Simulink model for hardware-in-the-loop simulations. The test on FPGA hardware provides reliable area and clock figures.

### Step 4: Create Reusable Module and Connect to Worker

Finally, we treated the verified block RAM-to-block RAM DSP hardware accelerator as a new module and integrated it into our main design by compiling the Handel-C to EDIF or RTL using the DK Design Suite. This reusable module is connected to the PicoBlaze network by wiring the ports of the block RAMs and the

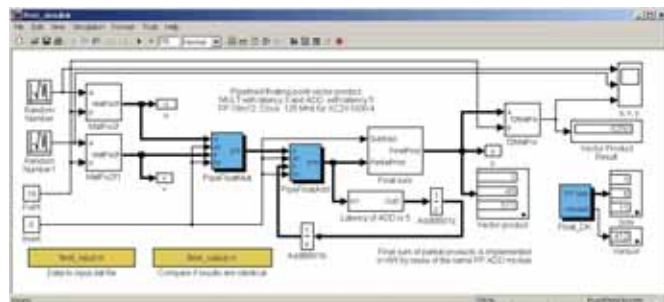


Figure 2 – Simulink test bench for floating-point 18m12 vector product based on Handel-C bit-exact models

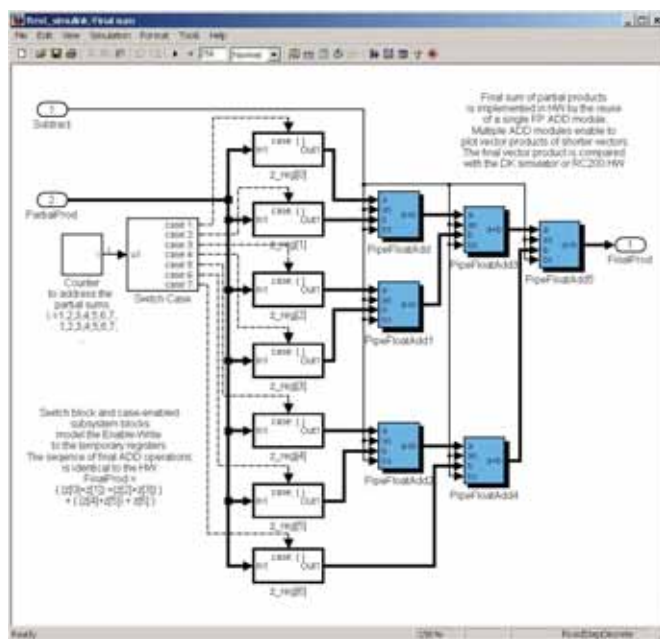


Figure 3 – Simulink subsystem based on Handel-C bit-exact models, including delay model of calculation wind-up at the end of the vector product batch

Part:	MHz	MFLOPs	mW
xc2v1000-4	100	700	1360
xc3s1500L-4	84	588	263

Table 2 – Results for 1024 x 1024 vector product in 18m12 floating point on the Celoxica RC200E (Virtex-II FPGA) and RC10 (Spartan-3L FPGA) boards

appropriate enable and controller interrupt signals. At this stage we tested the function of the DSP hardware accelerator under worker control using memory dump user support from the master.

### Step 5: Develop Complete DSP Design

We next assembled the complete design of workers and master, moving to assembly pro-

gramming of individual PicoBlaze workers and their interactions.

### Performance Results

Test results using the Celoxica RC200E (Virtex™-II FPGA) and RC10 (Spartan™-3L FPGA) boards are shown in Table 2. It is interesting to compare the power consumption of the PicoBlaze network architecture on Virtex-II devices (RC200E) with the identical design on the low-power 90 nm Spartan-3L device (RC10). The latter part gives a highly favorable floating-point performance-to-power ratio.

### Conclusion

With minimal overhead, PicoBlaze workers add flexibility to floating-point DSP hardware accelerators by their ability to call and reuse software functions (even if in assembly language only). Our proposed architecture enables more flexible and generic floating-point algorithms without the additional increase of hardware complexity associated with hardware-only implementations due to irregularities and complex multiplexing of pipelined structures. PicoBlaze cores are compact, simple, and

therefore manageable, without designers needing to combine too many new skills.

The use of floating-point designs developed using the DK Design Suite in combination with a Simulink framework provides an effective design path that is relatively easy to debug and scalable to more complex designs.

Spartan-3L technology considerably reduces power consumption compared to Virtex-II devices. Considering the benefits in terms of performance/power/price, Spartan-3L FPGA implementations of floating-point DSP pipelines using networks of PicoBlaze processors are an interesting option.

You can find complete information on the design and technology discussed in this article at [www.celoxica.com/xilinx](http://www.celoxica.com/xilinx).