WP243 (v1.0) April 19, 2006

M2C-Accelerator Facilitates Model-Based Design

By: Tom Feist

Model-Based Design (MBD) helps engineers and designers overcome the limitation of a document-based development process by replacing written specifications with comprehensive, system-level mathematical models. The models serve as an executable specification, reducing the need to build physical prototypes. Designers can simulate and explore architectures for implementation, quickly and comprehensively, throughout the development process inserting varying levels of abstraction to ensure that the end product meets both project requirements and system-level behavior. As application complexity increases, simulation performance requires acceleration. The *M2C-Accelerator* extends the Xilinx AccelDSP™ MBD solution by converting floating-point MATLAB to fixed-point C++ for accelerated MBD verification eliminating a potential bottleneck.

Model-Based Design Challenge

MBD applies mathematical models at various levels of abstraction to evaluate design options, predict system performance and test system functionality. However, Creating equivalent models in multiple languages with increasing levels of detail can be a time consuming and error prone process, often requiring additional test benches to ensure functional equivalence.

In the DSP domain, MATLAB is the algorithm modeling tool of choice, while C++ and System C are often used for system-level models. MATLAB is used by DSP algorithm developers as an alternative to C, because it provides both a highly productive verification environment and an efficient path to implementation. The built-in abstractions liberate the designer from the strict modeling style guides that are required by general-purpose languages, such a C/C++, allowing large design blocks to be represented with a high degree of efficiency.

Design Example

One line in MATLAB can represent a transform that can fill an entire FPGA. For example,

$$y = fft(x,64) \qquad \textit{Equation 1}$$

This returns the discrete Fourier transform of the vector "x", computed with a 64-point fast Fourier transform (FFT) algorithm. If "x" is a matrix, the MATLAB *fft()* function returns the Fourier transform for each column. Simulation at this level of abstraction is very fast and MATLAB provides a superior analysis environment for verification during algorithm development.

The MathWorks provides an interface to C simulators for integrating the DSP algorithm into system-level models written in C. Because both the MATLAB and C models are at a high-level of abstraction, there is a minimal performance impact on the system-level simulation. In this example, the floating-point FFT model and the equivalent C model run in about 6 seconds for a vector set of 10,000 frames.

As development proceeds towards the hardware, the FFT model needs to be replaced with a more detailed fixed-point hardware implementation model. At this point, the designer must decide on the hardware architecture required to meet the system-level requirements. The decision tree for selecting this architecture can be quite large as shown in Figure 1.

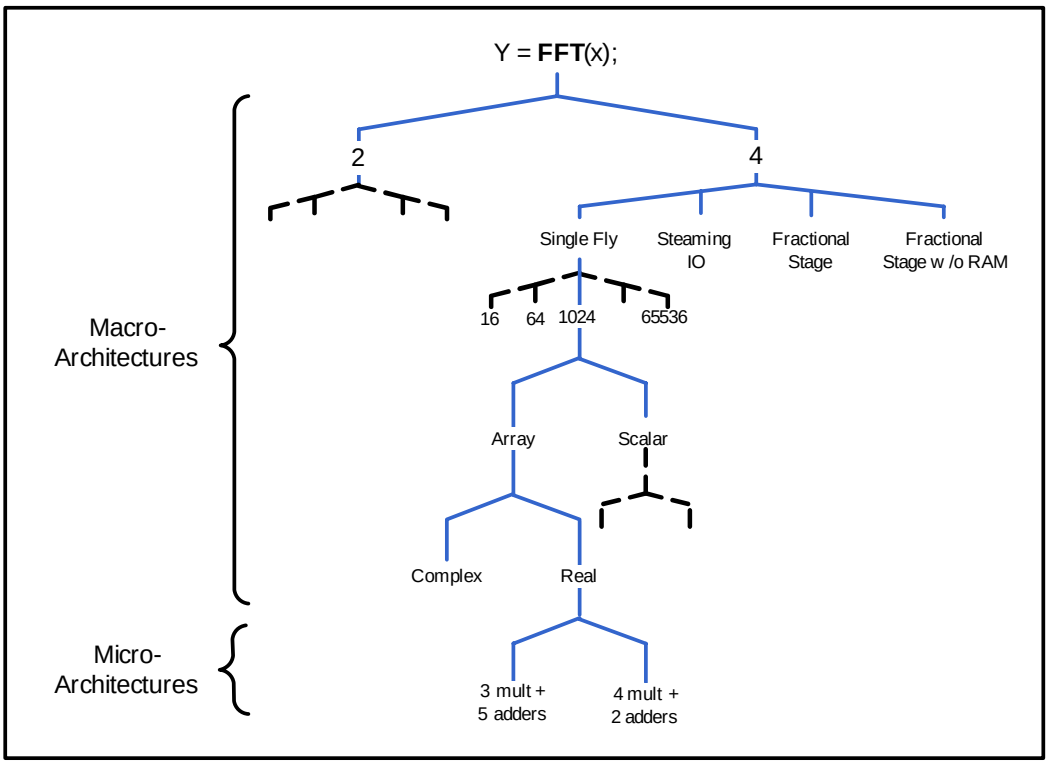

Figure 1: **FFT Hardware Architectures**

AccelDSP™ Synthesis Tool and AccelWare™ DSP IP Toolkits core generators have proven to significantly reduce the time required to develop FPGAs and ASICs. The AccelDSP Synthesis Tool reads in floating-point MATLAB, automates the conversion to fixed-point, and synthesizes RTL (VHDL or Verilog) along with a self-checking testbench that uses vectors captured from the original MATLAB simulations. AccelWare cores are designed to be compatible with built-in MATLAB functions and The MathWorks Signal Processing and Communications Toolboxes. They support a variety of macro- and micro-architectures for linear algebra functions, such as matrix inversions, as well as forward error correction cores like FFTs.

The following floating-point MATLAB example (Figure 2) is for the butterfly operation of a radix-2, 64-point FFT and represents a hardware accurate coding style.

This lower-level of abstraction results in verification times that are significantly slower. After the model has been converted to a hardware accurate, fixed-point representation, simulation time increases from 6 seconds to 21,643 seconds. To accelerate fixed-point MATLAB, AccelDSP provides performance optimized versions of the MATLAB quantize and quantizer functions. In this example, using the AccelDSP quantizer, simulation times improve 40X to 521 seconds. This performance gain, although significant, falls far short of the original 6 seconds and has a negative impact on overall system verification times.

```
for stage = 1:LOG2N
    base = 1; % Base index for accessing data vector components

    % Do processing based on flag; this effectively unrolls the first stage of the
FFT algorithm.
    if ppflag == 1

        % Repeat for each group of radix-2 butterflies
        for m1 = 1:groups
            % Repeat for each butterfly in group
            xtemp_real(base) = (x_real(brtbl(base))+x_real(brtbl(base+1)))/2;
            xtemp_imag(base) = (x_imag(brtbl(base))+x_imag(brtbl(base+1)))/2;

           xtemp_real(base+btrflys)=(x_real(brtbl(base))-x_real(brtbl(base+1)))/2;
           xtemp_imag(base+btrflys)=(x_imag(brtbl(base))-x_imag(brtbl(base+1)))/2;

            base = base + 2; % Move base index to first component in next group
        end;

        % Change flag
        ppflag = 0;
    else

        % Repeat for each group of radix-2 butterflies
        for m = 1:groups
            twidindex = 1; % Index into twiddle table

            % Repeat for each butterfly in group
            for k=1: btrflys
               Wn_real = sintbl(twidindex+NOVER4); % Cosine sample
               Wn_imag = sintbl(twidindex); % Sine sample; Note positive sign

tmp_real=xtemp_real(base+btrflys)*Wn_real+xtemp_imag(base+btrflys)*Wn_imag;
tmp_imag=xtemp_imag(base+btrflys)*Wn_real-xtemp_real(base+btrflys)*Wn_imag;

               xp_real = xtemp_real(base) + tmp_real;
               xp_imag = xtemp_imag(base) + tmp_imag;

               xq_real = xtemp_real(base) - tmp_real;
               xq_imag = xtemp_imag(base) - tmp_imag;

               xtemp_real(base) = xp_real/2;
               xtemp_imag(base) = xp_imag/2;

               xtemp_real(base+btrflys) = xq_real/2;
               xtemp_imag(base+btrflys) = xq_imag/2;

               base = base + 1; % Move base index to next component in group
               twidindex = twidindex + groups; % Update index into twiddle table
            end; % end of butterfly

          base = base + btrflys; % Move base index to first component in next group

        end;

    end; % End processing based flag

    groups = groups/2; % Update number of groups for next stage
    btrflys = btrflys * 2; % Update number of butterflies per group for next stage

end;
```

Figure 2: **64-Point Radix 2 FFT MATLAB Model**

Converting Floating-Point MATLAB Model to Fixed-Point C++

To accelerate MATLAB fixed-point verification performance, teams manually convert MATLAB models to C. The creation of this additional model is time consuming and error prone. To ensure model equivalence, additional test benches are required to provide checks between models.

To streamline this process, M2C-Accelerator automates the conversion from floating-point MATLAB to a fixed-point, bit-true C++. M2C-Accelerator also eliminates the need to manually rewrite MATLAB to C and to build additional test benches.

In the previously discussed FFT example, M2C-Accelerator generated a fixed-point C model in seconds, and when used in a mixed MATLAB / C++ simulation, verification times drop to 40.5 seconds - a 534X improvement. If this same model is used inside a C verification environment, the verification time drops to 20 seconds, providing an overall performance increase of 1082X (Figure 3).

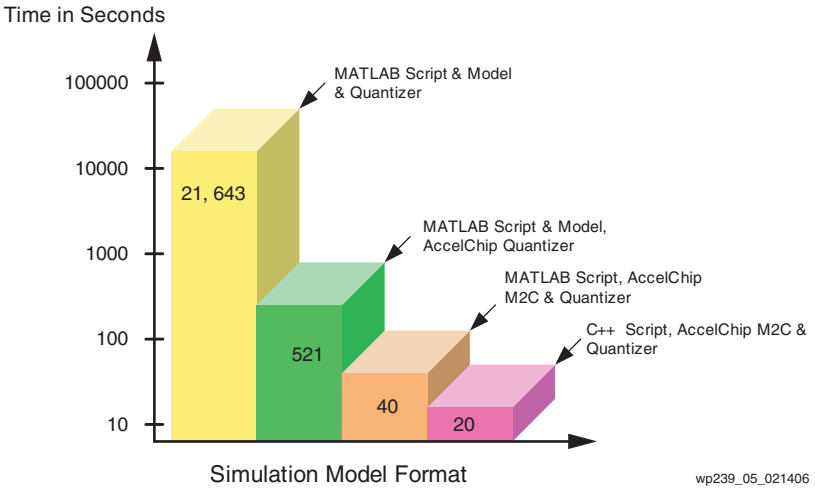

Figure 3: **FFT Example Simulation Run Times**

Summary

Design teams who adopt MBD will continue to use a mix of languages because of their specific advantages at different stages in the design flow. The AccelDSP Synthesis Tool and AccelWare DSP IP Toolkits core generator solutions provide an accelerated path to verified ASIC and FPGA designs for MATLAB to RTL. With M2C-Accelerator, Xilinx has extended its MBD solution to provide a verified path from MATLAB to fixed-point C that eliminates the need to manually convert the design, improves verification times, and reduces errors. This new capability has the added benefit of enabling designer teams, who previously would not use MATLAB as a development solution for verifying fixed-point designs, to use the M2C-Accelerator option to alleviate simulation bottlenecks.

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
04/19/06	1.0	Initial Xilinx release.