

Using MATLAB to Create IP for System Generator for DSP

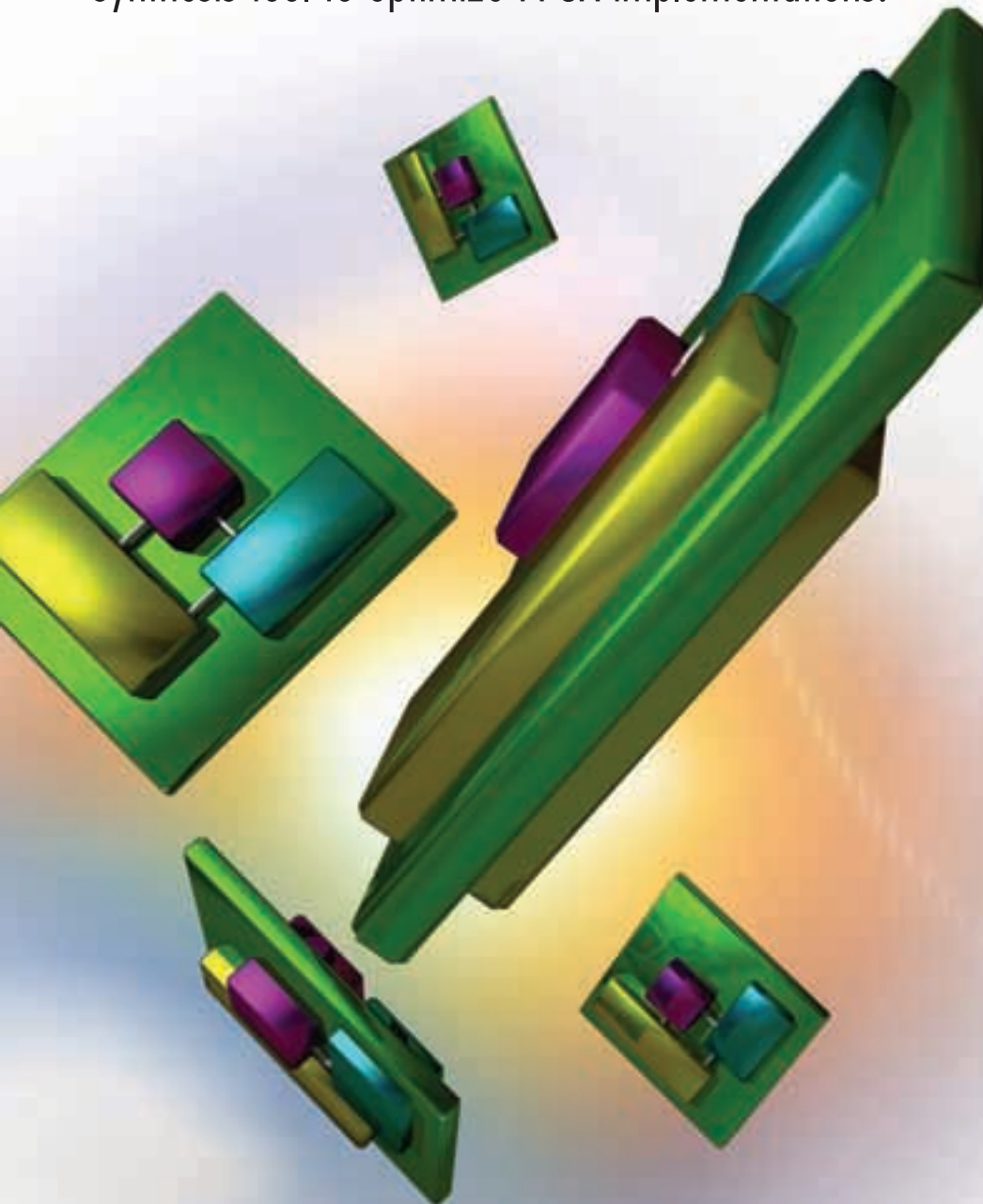
You can use MATLAB M-files with the AccelChip synthesis tool to optimize FPGA implementations.

by Tim Vanevenhoven
DSP Tools Product Manager
Xilinx, Inc.
tim.vanevenhoven@xilinx.com

DSP systems are best described by using a combination of both graphical and language-based methods. The MathWorks, an industry leader in DSP modeling software, caters to this dichotomy by providing a cycle-accurate graphical design environment called Simulink and a mathematical modeling language called MATLAB.

Simulink is well suited for the “system” aspects of DSP design, including control and synchronization of data flow to and from interfaces and memories. Simulink also provides a rich set of pre-defined DSP algorithms in the form of block sets that you can use to construct DSP systems. Simulink is not always the most effective environment for modeling proprietary algorithms, however. It unnecessarily burdens designers with cycle-accurate considerations and forces low-level arithmetic operations and array accesses to be constructed from graphical block sets rather than concise textual expressions.

Some DSP algorithm developers have found that the MATLAB language best meets their preferred style of development. With more than 1,000 built-in functions as well as toolbox extensions for signal processing, communications, and wavelet processing, MATLAB offers a rich and easy-to-use environment for the development and debugging of sophisticated algorithms.



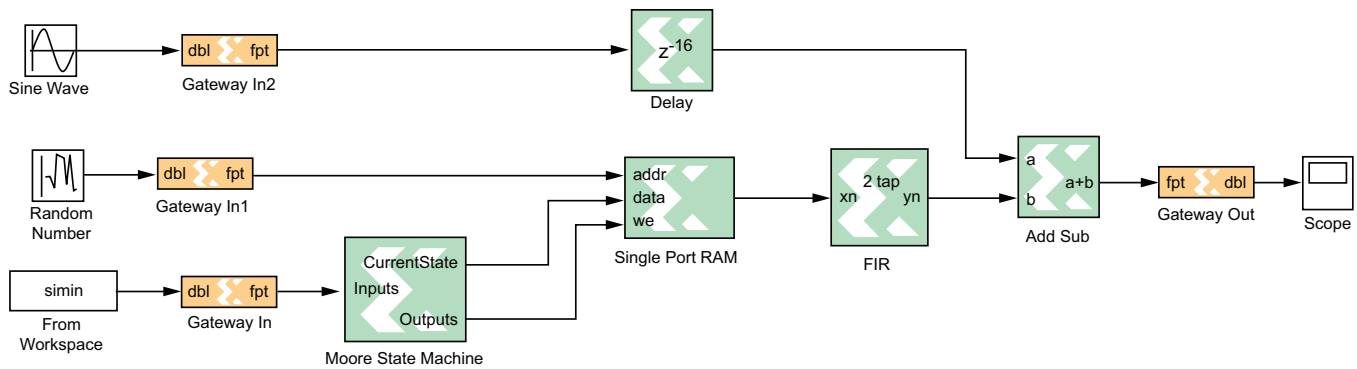


Figure 1 – Xilinx System Generator diagram showing system control and synchronization logic

Simulink unifies both modeling environments with an embedded MATLAB block, which allows MATLAB models to simulate within Simulink and compile into C code through Real-Time Workshop for processor-based DSP hardware implementations.

Xilinx® System Generator for DSP is well established as a productive tool for creating DSP designs in FPGAs. With a graphical environment based on Simulink and a pre-defined block set of Xilinx DSP cores, System Generator for DSP meets the needs of both system architects who need to integrate the components of a complete design and hardware designers who need to optimize implementations. System Generator for DSP, however, lacks support for a design flow based on MATLAB.

The Xilinx AccelDSP™ synthesis tool was developed specifically for algorithm developers and DSP architects who have embraced language-based DSP algorithm modeling. With the AccelDSP synthesis tool, algorithm developers can use their floating-point MATLAB M-files to perform stimulus creation, algorithm evaluation, and results post-processing.

DSP Hardware Systems

System Generator for DSP is well suited for modeling the DSP system, which comprises not only the core DSP algorithm but synchronous interfaces to external buses, memory read/write accesses, system data synchronization, and overall system control. System Generator for DSP provides control-oriented blocks such as the MicroBlaze™ processor as well as individ-

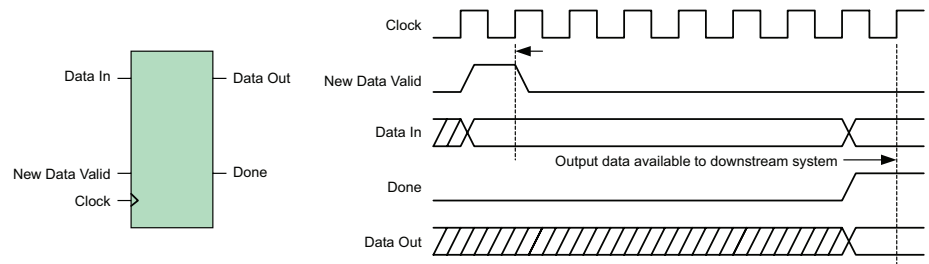


Figure 2 – AccelDSP handshaking interface

ual register, delay, and memory blocks for implementing the synchronous components of a DSP system (Figure 1).

Custom DSP Algorithms

The heart of any DSP system is the algorithm. Algorithms distinguish themselves from systems in that a resulting output is a function of a given set of inputs without a sense of clock or hardware. This is described by the simple equation:

$$y = f(x)$$

You can execute an algorithm defined in MATLAB on an FPGA, DSP processor, or software; each will have a different sense of cycle accuracy.

The unique characteristics of an algorithm offer two distinct advantages. First, algorithm developers are completely unencumbered by hardware implementation details and can focus solely on functional behavior. This is exactly why an estimated 90% of the algorithms used today in DSP originate as MATLAB models, even when a design flow dictates that they be re-imple-

mented at a later point in time as Simulink or System Generator for DSP diagrams. Calculating the fast Fourier transform (FFT) of a $4 \times 1,024$ matrix of data is possible with a simple MATLAB statement, without concern for radix, scaling, buffering, or synchronization of valid signals, as shown here:

$$y = \text{fft}(\text{data}, 1024);$$

Second, when modeling an algorithm, a given set of outputs corresponds to a given set of inputs; therefore, synchronization issues do not need to be addressed in the generated hardware. This makes an algorithm inherently “schedule-able” through a tool like the AccelDSP synthesis tool. Hardware requirements may dictate the need to use multiple clock cycles to compute an output, as in the case of a resource-shared MAC FIR filter, but this operation lends itself nicely to the AccelDSP synthesis tool’s automated flow. The introduction of a simple hardware handshaking interface enables easy integration into the overall system, as shown in Figure 2.

MATLAB operators such as matrix transposition allow the MATLAB code to be compact and easily readable. And complex operations like matrix inversion are done using MATLAB's extensive linear algebra capabilities.

Using the AccelDSP and System Generator Tools Together

The AccelDSP synthesis tool enables System Generator for DSP to support both DSP system and algorithm modeling methods by generating System Generator IP blocks based on floating-point MATLAB models. This results in a design flow for FPGAs that is similar to the functionality obtained through the use of the embedded MATLAB block (Figure 3). You can capture the system aspects of the design using the Xilinx DSP block set and capture the algorithm using floating-point MATLAB. The System Generator IP blocks created by the AccelDSP synthesis tool are fixed-point and cycle-accurate.

Kalman Filter Example

Let's take an advanced algorithm written in MATLAB, use the AccelDSP synthesis tool to synthesize the design, and then integrate the algorithm into a System Generator model. A Kalman filter is a special class of recursive, adaptive filters that is well suited to combining multiple noisy signals into a clearer signal. Kalman filters start with a mathematical model of an object – such as a commercial aircraft being tracked by ground-based radar – and use the model to predict future behavior. The filter then uses measured signals (such as the signature of the aircraft returned to the radar receiver) to periodically correct the prediction.

The following is a MATLAB M-file for a Kalman filter. The algorithm defines

matrices R and I that describe the statistics of the measured signal and the predicted behavior. The last nine lines of the algorithm are the code that predicts forward and the code that corrects itself.

```
function [S] = simple_kalman(A)
```

```
    DIM = size(A,2);
    persistent p P_cap
    if isempty(P_cap)
        P_cap = [8 0 0; 0 8 0; 0 0 8];
        p = ones(DIM,1)/2;
    end;
```

```
    I = eye(DIM);
    R = [128 0 0; 0 128 0; 0 0 128];
```

```
    % estimate step:
    P_cap_est = P_cap+I;
```

```
    % correction step:
    K = P_cap_est * inv(P_cap_est+R);
    p = p + K * ( A' - p );
    P_cap = (I - K)*P_cap_est;
    S = p';
```

Common operators like addition and subtraction operate on arrays like A or P_cap without the need to write loops, which would be required in languages like C. Two-dimensional arrays are automatically multiplied as matrices without any special annotation.

MATLAB operators such as matrix transposition allow the MATLAB code to be compact and easily readable. And com-

plex operations like matrix inversion are performed using MATLAB's extensive linear algebra capabilities. Although such an algorithm can be constructed as a block diagram, doing so will obscure the algorithm structure so readily apparent in MATLAB.

With the AccelDSP synthesis tool, you can take complex MATLAB toolbox and built-in functions (such as the matrix inverse used in the Kalman filter example) directly to hardware using the AccelWare IP tool kits. These tool kits offer multiple matrix inverse methods. Core selection will depend on the size, structure, and values of the matrix. Returning to the Kalman filter example, the most suitable approach is to use the AccelWare QR matrix inverse core. AccelWare cores are generated based on MATLAB syntax and are available in a variety of implementations that let users optimize designs for speed, area, power, or noise. This small code edit would be required to use the AccelWare function:

```
    % K = P_cap_est * inv(P_cap_est+R);
```

```
    K = P_cap_est *
        accel_qr_inverse (P_cap_est+R);
```

Synthesizing MATLAB with the AccelDSP Synthesis Tool

With the AccelDSP synthesis tool, you can perform a floating-point simulation to establish a baseline. The design is then converted into a fixed-point representation so that the fixed-point effects can be simulated. An array of features helps analyze these effects and fixed-point design options like saturation and rounding modes. Bit growth is automatically propagated through the design in a manner that allows for user-controlled overrides. This algorithmic design exploration process helps you obtain the ideal quantization that minimizes bit widths while managing overflows/underflows, allowing early trade-offs of silicon area versus performance metrics.

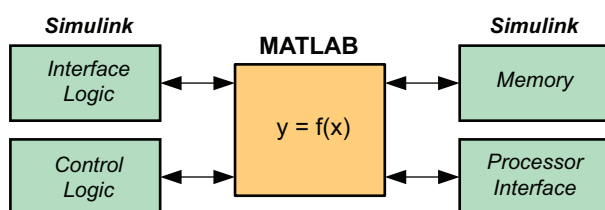


Figure 3 – DSP system block diagram

DSP Synthesis Directive	Effect on Generated Hardware
Rolling/Unrolling of For Loops	Improves input sampling rate by reducing throughput
Expansion of Vector and Matrix Additions and Multiplications	Improves input sampling rate by reducing throughput
RAM/ROM Memory Mapping of Arrays	Improves FPGA utilization by mapping arrays into dedicated Xilinx block RAM resources
Pipeline Insertion	Improves input sampling rate by improving clock frequency performance
Shift Register Mapping	Improves FPGA utilization by mapping shift register logic into SRL16s

Table 1 – Synthesis directives to control the generated hardware

Once suitable quantizations have been determined, the next step with the AccelDSP synthesis tool is to generate RTL for the target Xilinx device. You can dictate hardware intent through the use of the synthesis directives listed in Table 1.

Using these directives constitutes hardware-based design exploration that allows the design team to further improve quality of results. In synthesizing the RTL, the AccelDSP synthesis tool evaluates and schedules the entire algorithm, performing boundary optimization when possible.

Throughout this flow, the AccelDSP tool maintains a uniform verification environment through the use of a self-checking test bench: the input/output vectors that were generated when verifying the fixed-point MATLAB design are used to verify

the generated RTL. The AccelDSP synthesis tool will also report the throughput and latency of the Kalman filter, which is essential to gauge whether the design meets specifications and to produce a cycle-accurate System Generator model.

Generating a System Generator Model

Upon successful completion of RTL verification, the design is now ready to gener-

ate a System Generator model by clicking on the “Generate System Generator” icon, as shown in Figure 4. The AccelDSP tool generates a System Generator IP block that supports both simulation and RTL code generation.

At this point, the design flow transitions to System Generator for DSP, where the new block for the Kalman filter is available in the Simulink library browser. You only need to select the Kalman filter block and drag it into the destination model to incorporate the AccelDSP-generated Kalman filter into a System Generator design, illustrated in Figure 5.

Conclusion

Custom DSP algorithms are best modeled mathematically using MATLAB, while complete systems are best modeled as cycle-accurate using Simulink. The marriage of these two modeling domains provides an efficient means to design DSP systems into FPGAs by allowing the use of the rich MATLAB language, including the built-in and toolbox functions, to create System Generator IP blocks of complex DSP algorithms. 🌟



Figure 4 – “Generate System Generator” command in the AccelDSP synthesis tool

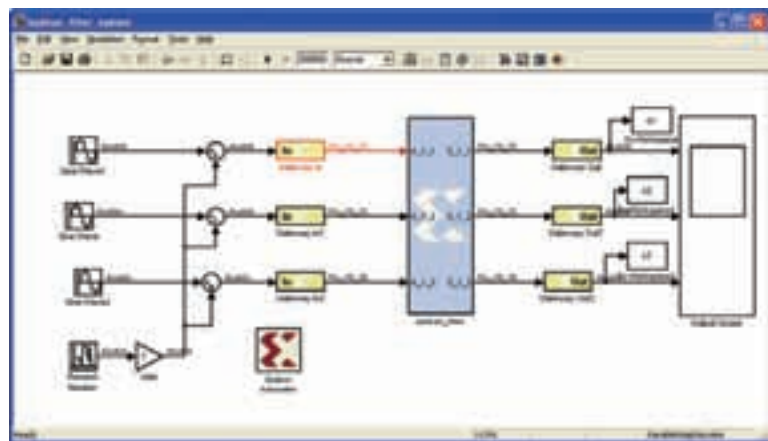


Figure 5 – Kalman filter added to a System Generator design

TAKE THE NEXT STEP (Digital Edition: www.xcellpublications.com/subscribe/)

- See an online demo of the System Generator and AccelDSP tools.
- Evaluate or purchase:
 - The AccelDSP synthesis tool.
 - System Generator for DSP.