Zynq for Video Applications

Mike Mitchell
DSP Specialist,
Austin, TX
Serial vs. Parallel DSP Processing

**Standard DSP Processor ALU – Sequential (Generic DSP)**
- Data In
- Coefficients → X
- Single-MAC Unit
- +
- Reg
- Data Out

**FPGA - Fully Parallel Implementation (Virtex-6 FPGA)**
- Data In
- C0 → X
- C1 → X
- C2 → X
- C3 → X
- ... → X
- +
- Data Out

**Calculations**
- **Standard DSP Processor**
  - 1.2 GHz
  - \( \frac{1.2 \text{ GHz}}{2016 \text{ clock cycles}} = 595 \text{ KSPS} \)
- **FPGA**
  - 600 MHz
  - \( \frac{600 \text{ MHz}}{1 \text{ clock cycle}} = 600 \text{ MSPS} \)

**Demuxed Filter Architectures**
- Can enable even higher data rates
Software vs Hardware Acceleration

Different FFT architectures can push performance even higher, consider sub 5us 2D FFT (GSPS)

550MSPS rate
After pipeline is filled

5.4MSPS rate
Xilinx DSP Performance Leadership

DSP Performance Gap

Performance (Algorithmic and Processor Forecast)

Time

Algorithm Complexity

- 2300 GMACs
- 1000 GMACs
- 32 GMACs
- 10 GMACs

Virtex™-7

Virtex™-6

Spartan™-6

Traditional DSP Architectures

Source: Forward Concepts

DSP Processors:

- TI DM648 C64x @ 1.1GHz: 8.8GMACs (16 bit mults/cycle)
- TI DM8148 C674x @ 750MHz: 6GMACs (16 bit mults/cycle)
Comparing TI DM648 Functionality and Zynq
TI TMS320DM648

- 512MB address space
- DDR2 (32 bit)
- EMIFA 16 bit interface
- 46.875MHz - 167.67MHz
- Table 6.42 pg 112
- tms320dm648.pdf

- Up to 1.1GHz C64x clock rates
- 8 MIPs/MHz
- 6 ALUs, 2 Multipliers (32, 16 or 8 bit ops)
- 8.8GMACs

- 32KB I/D L1 Cache
- 512KB L2
- 64KB ROM

- 90nm process technology

Xilinx Copyright 2012
**TI DM648 vs Zynq**

**TI TMS320DM648**
- 90nm process
- Up to 1.1GHz C64x
  - 8.0 DMIPS/MHz max (based on clock frequency)
  - 32KB I/D cache
  - 512KB L2 cache
  - 64KB ROM
  - 8.8GMACs max (based on 1.1GHz clock frequency)

**Zynq**
- 28nm process
- Dual, 667-800MHz Cortex A9 + NEON coprocessor
  - 2.5 DMIPS/MHz & 2.88 CoreMark/MHz
  - 32KB I/D cache
  - 512KB L2 cache
  - 256KB on chip memory
  - NEON has FP (2MFLOPS/MHz) + fixed point SIMD capability per core
  - 7020 device has 220 DSP48s (628MHz -3 speed grade) for 138.1GMACs of performance
  - Speed grades -1 (464MHz), -2 (550), -3 (628)

**XILINX** ➤ ALL PROGRAMMABLE.
Comparing TI DM8148 Functionality and Zynq
TI TMS320DM8148

45nm process technology

Up to 750MHz C674x
32KB L1, 256KB L2
6 ALUs, 2 Multipliers
(32, 16 or 8 bit ops)
8.8GMACs,
6 MFLOPs/MHz

Cortex A8
32KB I/D L1 Cache
512KB L2 Cache
64KB RAM
**TI DM8148 vs Zynq**

**TI TMS320DM648**
- 45nm process
- Up to 1GHz Cortex A8 + NEON coprocessor
  - 2.0 DMIPs/MHz
  - 32KB I/D cache
  - 512KB L2 cache
  - 64KB on chip memory
- Up to 750MHz C674x
  - 8.0 MMACs/MHz max (based on clock frequency)
  - 6MFLOPs/MHz (based on clock frequency)

**Zynq**
- 28nm process
- Dual, 667-800MHz Cortex A9 + NEON coprocessor
  - 2.5 DMIPs/MHz
  - 32KB I/D cache
  - 512KB L2 cache
  - 256KB on chip memory
  - NEON has FP (2MFLOPS/MHz) + fixed point SIMD capability per core
  - 7020 device has 220 DSP48s (628MHz -3 speed grade) for 138.1GMACs of performance
  - Speed grades -1 (464MHz), -2 (550), -3 (628)

**Xilinx**
- All programmable.
Zynq – A Full Custom, Single Chip ASSP

Static Memory Controller
Quad-SPI, NAND, NOR
Dynamic Memory Controller
DDR3, DDR2, LPDDR2
AMBA® Switches
Programmable Logic:
System Gates, DSP, RAM

16/32 bit, 533MHz

ARM® CoreSight™ Multi-core & Trace Debug
NEON™/ FPU Engine
Cortex™-A9 MPCore™
32/32 KB I/D Caches
512 KB L2 Cache
Snoop Control Unit (SCU)
Timer Counters
256 KB On-Chip Memory
General Interrupt Controller
DMA
Configuration

Multi-Standards I/Os (3.3V & High Speed 1.8V)
Multi-Gigabit Transceivers
PCIe

ACP
64 bit cache coherent access to CPU Data in L1/L2 cache

Xilinx Copyright 2012
Zynq Targeted to Signal Processing Applications

Processor Core Complex
- Dual ARM® Cortex™-A9 MPCore™ with NEON™ extensions
- Single / Double Precision Floating Point support – 2 MFLOPS/MHz
- Up to 800 MHz operation

Enables Massive Parallel Processing
- Up to 900 DSP blocks delivering over 1080 GMACs (symmetric FIR implementation)

Over 3000 Internal Interconnects
- Up to 100Gb of BW
- Memory-mapped interfaces

AMBA Open Standard Interconnect
- High bandwidth interconnect between processing system and programmable logic
- ACP port for enhanced hardware acceleration and cache coherency for additional soft processors

Application specific custom hardware accelerator

Processing System Ready to Program

Xilinx Copyright 2012
Zynq High Performance (HP) Ports

PL Interface to PS Memory Subsystem

Each high-performance AXI port has these characteristics:

- Reduced latency between PL and processing system memory
- 1 KB deep FIFO
- Configurable either as 32- or 64-bit AXI interfaces
- Supports up to a 32 word buffer for read acceptance
- Supports data release control for write accesses to use AXI interconnect bandwidth more efficiently
- Supports multiple AXI commands issuing to DDR and OCM
SW vs HW Task Implementation Decisions

Keep it in Software
- Not in critical path
- Enough “free” cycles
- Easier to code in software than hardware
  - Use math library functions
- NEON coprocessor
  - Supports integer vector operations
  - Single floating point operations

Move to Hardware
- Higher performance
  - Architecture determines area / throughput (ie: reference FFT page 3 & size vs latency discussion)
- Customize to meet your needs & reduce processor bandwidth requirements
- Excellent for iterative, pipelined & parallel processing

Move to another processor
- Both Microblaze & Picoblaze can exist in the fabric
TI OMAP35xx vs Zynq for H.264 CODEC

- [http://www.iqmagazineonline.com/archive27/pdf/Pg32-37.pdf](http://www.iqmagazineonline.com/archive27/pdf/Pg32-37.pdf) - h.264

- Base CODEC is C code (ie: not optimized for NEON coprocessor or C64x VLIW).
  - C code was also optimized for C64x VLIW core & CA8 NEON coprocessor.
  - Use of Cortex A8 + NEON shows better performance than the C64x VLIW core in an OMAP3530.

- Demonstrates required processor clock rate compared to achieve the same VGA 30fps 1Mbps video streams for a Cortex A8 vs. OMAP3430
  - lower clock rate MHz means better performance

- Demonstrates NEON coprocessor is better than OMAP3530 C64x VLIW core.

<table>
<thead>
<tr>
<th>MPEG-4 SW Decoder</th>
<th>Version</th>
<th>OMAP 3430 (MHz)</th>
<th>Profiler (MHz)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Base</td>
<td>225,3</td>
<td>244,0</td>
<td></td>
</tr>
<tr>
<td>Optimized</td>
<td>137,6</td>
<td>110,3</td>
<td></td>
</tr>
</tbody>
</table>

Table 3: Measured performance for VGA 30fps 1Mbps MPEG-4 video stream decoding

<table>
<thead>
<tr>
<th>MPEG-4 SW Encoder</th>
<th>Version</th>
<th>OMAP 3430 (MHz)</th>
<th>Profiler (MHz)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Base</td>
<td>1090,7</td>
<td>901,8</td>
<td></td>
</tr>
<tr>
<td>Optimized</td>
<td>488,3</td>
<td>369,0</td>
<td></td>
</tr>
</tbody>
</table>

Table 4: Measured performance for VGA 30fps 1Mbps MPEG-4 video stream encoding
Zynq NEON 3rd party ecosystem – Image, Audio, Video

<table>
<thead>
<tr>
<th>Company</th>
<th>Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ingenient</td>
<td>H.264, VC1, MPEG-4</td>
</tr>
<tr>
<td>On2 Technologies</td>
<td>VP6/7, MPEG-4, VC-1, H.264 (enc+dec), video stabilization</td>
</tr>
<tr>
<td>Ittiam</td>
<td>MPEG-4, MPEG-2, H.263, H.264, WMV9, VC1, DD+</td>
</tr>
<tr>
<td>Aricent</td>
<td>MPEG-4, H.263, H.264, WMV9, audio</td>
</tr>
<tr>
<td>Tata Elxsi</td>
<td>H.264, VC1</td>
</tr>
<tr>
<td>Spirit DSP</td>
<td>TEAMSpirit voice &amp; video</td>
</tr>
<tr>
<td>VisualOn</td>
<td>H.264, MPEG-4, H.263, WMV</td>
</tr>
<tr>
<td>Actimagine</td>
<td>MobiClip</td>
</tr>
<tr>
<td>Fraunhofer</td>
<td>Codecs</td>
</tr>
<tr>
<td>CoreCodec</td>
<td>Multichannel audio processing</td>
</tr>
<tr>
<td>Adobe</td>
<td>Flash products</td>
</tr>
<tr>
<td>TMC</td>
<td>MPEG-4</td>
</tr>
<tr>
<td>YAPPA</td>
<td>GUI visual effects</td>
</tr>
<tr>
<td>drawElements</td>
<td>2D GUI library</td>
</tr>
<tr>
<td>Espico Ltd</td>
<td>Audio: low-bitrate &amp; digital theater, consulting</td>
</tr>
<tr>
<td>CoreAVC</td>
<td>CoreAVC ultra fast codec</td>
</tr>
</tbody>
</table>

ARM NEON widely supported by software partners

Full list on [www.arm.com](http://www.arm.com) NEON ecosystem page
Xilinx IP Library for Camera Systems

Same IP supports video processing applications and displays

- Image Processing
  - Defective Pixel Correction
  - Color Filter Array Interp
  - Color Correction Matrix
  - Gamma Correction
  - Color Space conversion
  - Statistics Module
  - Noise Reduction
  - Edge Enhancement

- Video Processing
  - Video Scaler
  - On-Screen-Display
  - Motion Adaptive Noise Reduction
  - Image Characterization
  - Object Segmentation
  - Memory and timing
    - Timing Controller
    - MPMC + VFBC
    - Video DMA

- Compression
  - H.264/MPEG-4
  - MPEG-2
  - MPEG-4
  - JPEG
  - JPEG-2000

- Connectivity
  - 10/100/1G Ethernet
  - SDI
  - Ethernet AVB
  - Display port
    - Vx1, Vx4
  - HDMI/DVI
  - Firewire
  - GigE Vision
  - Camera Link

Cortex A9

Memory Buffer

Command and control
- Auto White Balance
- Auto Exposure
- Auto Focus

Video Out, Ethernet, etc.
Some XADC Examples

Motor Control / Energy Conversion

Sensor Interfacing

Xilinx Copyright 2012
Video Applications
Computer Vision Processing Complexity

- **Image Pre-processing**
  - Ultra high data rates
  - Low algorithm complexity
  - (complexity implies code amount for a HLL)

- **Lens Correction**
  - High to medium data rates
  - Medium algorithm complexity

- **Segmentation and Object Analysis**
  - Better FPGA fit

- **Heuristics or Expert System**
  - Low data rates
  - High algorithm complexity
  - CPU type tasks
  - Realm of objects vs pixels

Better Processor fit
Computer Vision

Implementing Computer Vision Applications is challenging
- Limited engineering experience in building practical systems
- Most solutions are ad hoc and highly application specific
- Embedded systems are often highly constrained in cost, size, and power consumption

Computer Vision Is Computationally Intensive
- A 720p optical flow algorithm optimized for a C64x VLIW DSP architecture, consumed about 200MHz/frame/second for a throughput of 5 frames per second

Embedded Computer Vision is a Good Fit for FPGAs
- Highly custom designs based on application
- High computational requirements, algorithms are diverse and dynamic, no off the shelf ASSPs to address
Computer Vision Hardware Design

» Simulation modeling may be developed & tested using Matlab or C

» Converting the algorithmic portions (segmentation, object analysis, lens correction, image warping) into a hardware accelerator block may be challenging.

» Xilinx has options to simplify conversion of C or Matlab or Simulink designs into hardware accelerators.
Hardware/Software DSP Design Tradeoffs

- Hardware Accelerators can be exponentially faster than software and significantly reduce my code complexity and debug.
- Building Hardware Accelerators requires a hardware design methodology.
- I would like to write everything in software and then determine what needs to be accelerated which leads to…
The Dilemna:
- Which functionality should be in software vs hardware?
AutoESL Introduction
Agenda

- High-Level Synthesis
- AutoESL
- Design Examples
- Success Stories
High-Level Synthesis

- resource mapping
- scheduling
- throughput/latency

- map FPGA resource
- memory/mult/logic/IO

- placement/routing of resource
Agenda

- High-Level Synthesis
- AutoESL
- Design Examples
- Success Stories
- Roadmap
AutoESL: Block Creation Design Flow

- **Starts at C**
  - C
  - C++
  - SystemC

- **Produces RTL**
  - Verilog
  - VHDL
  - SystemC

- **Automates Flow**
  - Verification
  - Implementation

---

Mapping C to FPGA

Xilinx Copyright 2012
Benefits

- **Automatic Scheduling and Resource Sharing**
  - Most time consuming part of FPGA design
  - Parallelization
  - Pipelining

- **QoR (Quality of Results)**
  - Results rivaling hand-coded RTL

Rapid design exploration
More design iterations / day can be achieved by migrating functional verification to C/C++

- Orders of magnitude faster than RTL for large designs
- RTL verification becomes final check

**Benefits**

Block level verification *significantly* reduced

*RTL Simulations performed using ModelSim*
Benefits

Portability/Design Reuse
- Technology migration
- Cost reduction
- Full Resource/
  Performance/Power analysis

Design and IP reuse

Xilinx Copyright 2012
Benefits

Floating Point Support

- Use ‘float’ and ‘double’ C/C++ data types
- Use standard math operators (+, -, *, /)
  - Additional float functions may require C/ C++ declarations
    - e.g. for single-precision square root
      ```c
      #include <math.h>
      extern “C” float sqrf(float);
      ```
- Using Xilinx floating-point core library reference

Floating Point Computation using AESL
Agenda

- High-Level Synthesis
- AutoESL
- Design Examples
- Success Stories
Example: Matrix Multiplication

Matrix Multiplication Example

- Multiply the rows of A with the columns of B and sum

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>Res</th>
</tr>
</thead>
<tbody>
<tr>
<td>11 12 13</td>
<td>21 22 23</td>
<td>870 906 942</td>
</tr>
<tr>
<td>14 15 16</td>
<td>24 25 26</td>
<td>1086 1131 1176</td>
</tr>
<tr>
<td>17 18 19</td>
<td>27 28 29</td>
<td>1302 1356 1410</td>
</tr>
</tbody>
</table>

- Example in detail

\[
\begin{align*}
\mathbf{r}_{00} &= 11 \times 21 + 12 \times 24 + 13 \times 27 = 870 \\
\mathbf{r}_{01} &= 11 \times 22 + 12 \times 25 + 13 \times 28 = 906
\end{align*}
\]

Etc…
The C code for a matrix multiplication is fairly intuitive

- A series of nested loops

```c
// Iterate over the rows of the A matrix
Row: for(int i = 0; i < MAT_A_ROWS; i++) {
    // Iterate over the columns of the B matrix
    Col: for(int j = 0; j < MAT_B_COLS; j++) {
        // Do the inner product of a row of A and col of B
        res[i][j] = 0;
        Product: for(int k = 0; k < MAT_B_ROWS; k++) {
            res[i][j] += a[i][k] * b[k][j];
        }
    }
}
```

Algorithm easily parallelizable without changing code
Matrix Multiply: Code

```c
#include "mat_mult.h"

void mat_mult(complex_t c[M][M], complex_t a[M][M], complex_t b[M][M]) {
    int i, j, k;
    complex_t tmp, prod;

    loop_i:
    for (i=0; i<M; i++) {
        loop_j:
        for (j=0; j<M; j++) {
            tmp.c = 0;
            tmp.i = 0;

            loop_k:
            for (k=0; k<M; k++) {

                //pragma AP PIPELINE
                prod.c = a[i][k].c * b[k][j].c - a[i][k].i * b[k][j].i;
                prod.i = a[i][k].c * b[k][j].i + a[i][k].i * b[k][j].c;
                tmp.c += prod.c;
                tmp.i += prod.i;

            }
            c[i][j].c = tmp.c;
            c[i][j].i = tmp.i;

        }
    }
}
```

Xilinx Copyright 2012
Matrix Multiply: Results (fixed pt)

▶ Computes 4x4 matrix multiplication
  – complex data sample of 16 bit each
  – one output element per clock cycle with 6 clk cycle latency
  – can be changed to produce all 16 elements in a single clk cycle at the cost of more IO and resources
  – 12M matrix mult operations per second

▶ Target device: V6LX240
▶ Timing : 5.2 ns (≈192 MHz)
▶ Resource:
  – 199 LUT, out of 150720
  – 391 FF, out of 301440
  – 18 DSP48, out of 768
  – 0 BRAMs, out of 416 of 36kbit

▶ Can easily converted to floating point implementation
  – by changing the variable types to float/double
Matrix Multiply: Results (floating pt)

- Computes 4x4 matrix multiplication
  - complex data sample of single precision floating pt each
  - one output element per clock cycle with 30 clk cycle latency
  - 12M matrix mult operations per second
- Target device: V6LX240 -2
- Timing: 5.3 ns (=189 MHz)
- Resource:
  - 5612 LUT, out of 150720
  - 5486 FF, out of 301440
  - 76 DSP48, out of 768
  - 0 BRAMs, out of 416 of 36kbit
Floating Point Matrix Multiply Software vs. Hardware Acceleration

- Multiplication of two 32x32 floating point Matrices
  - 4.69x acceleration factor in a Zynq 7020
- AXI based accelerator created using AutoESL, 166MHz Fabric clock

AutoESL hardware accelerator IP for Zynq
Zynq HD Video Processing Demo

1080p 60fps

Sobel Filter Output (Edge Detect)

Demonstrate Sobel filter Running in Software (CA9) & Hardware (via AutoESL)
Representative Processor Implementation

- Processor implementation uses arrays for input and output
- Full frame delay between input and output – all output pixels available to the next function in the pipeline at the same time
- 3x3 Sobel computation window is created by random access into input_image array

```c
for(i = 0; i < height; i++){
    for(j=0; j < width; j++){  
        x_dir = 0; 
        y_dir = 0;  
        if((i > 0) && (i < (height-1)) && (j > 0) && (j < (width-1))){
            for(rowOffset = -1; rowOffset <= 1; rowOffset++){
                for(colOffset = -1; colOffset <= 1; colOffset++){
                    x_dir = x_dir + input_image[i+rowOffset][j+colOffset] * Gx[1+rowOffset][1+colOffset]; 
                    y_dir = y_dir + input_image[i+rowOffset][j+colOffset] * Gy[1+rowOffset][1+colOffset];
                }
            }
            edge_weight = abs(x_dir) + abs(y_dir); 
            output_image[i][j] = edge_weight;
        }
    }

Xilinx Copyright 2012
```
FPGA Optimized Implementation

Changes with processor code center on data movement

- Creating line buffers “buff_A”
- Creating 3x3 memory window for edge processing “buff_C”
- Memory data movement operations on “buff_A” and “buff_C”
- Iteration space extension to account for line buffers “rows+1, cols+1”

Memory management code required for high performance implementation

Original processor code will synthesize in AutoESL, but will have poor performance.
Agenda

- High-Level Synthesis
- AutoESL
- Design Examples
- Success Stories
QR Decomposition

**Application**
- Radar
  - 1024x64 QRD
  - 120MHz
  - Floating Point

**Key Benefit**
- Reduced design time
- Floating Point synthesis
- Improved QoR

<table>
<thead>
<tr>
<th></th>
<th>Hand-coded VHDL</th>
<th>AutoESL C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Design Time (weeks)</td>
<td>12</td>
<td>1</td>
</tr>
<tr>
<td>Latency (ms)</td>
<td>37</td>
<td>21</td>
</tr>
<tr>
<td>Memory (RAMB18E1)</td>
<td>134 (16%)</td>
<td>10 (1%)</td>
</tr>
<tr>
<td>Memory (RAMB36E1)</td>
<td>273 (65%)</td>
<td>138 (33%)</td>
</tr>
<tr>
<td>Registers</td>
<td>29686 (9%)</td>
<td>14263 (4%)</td>
</tr>
<tr>
<td>LUTs</td>
<td>28152 (18%)</td>
<td>24257 (16%)</td>
</tr>
</tbody>
</table>

Quick path to FPGA

Xilinx Copyright 2012
Sphere Decoder

Overview
- Key block in receiver structure in MIMO-OFDM systems
- 5MHz MIMO-OFDM; 360 data sub-carriers every 102 usec
- 4×4 antenna configuration; 64-QAM
- *Challenging computational requirements*

RTL Reference Implementation
- Target clock frequency was 225MHz in Virtex-5
- Implemented in System Generator
- Reference Matlab

HLS implementation
- Same BER performance
- Target same design point as reference implementation

Sphere Decoder: Results

Compare Development Time and Quality of Results

* Development time for AutoESL includes:
  - Learning the tool
  - Producing results
  - Design space exploration
  - Detailed verification

<table>
<thead>
<tr>
<th>Metric</th>
<th>SysGen</th>
<th>AutoESL – Expert Result</th>
<th>% Diff</th>
</tr>
</thead>
<tbody>
<tr>
<td>Development Time</td>
<td>16.5</td>
<td>15*</td>
<td>-9%</td>
</tr>
<tr>
<td>LUTs</td>
<td>27,870</td>
<td>29,060</td>
<td>+4%</td>
</tr>
<tr>
<td>Registers</td>
<td>42,035</td>
<td>31,000</td>
<td>-26%</td>
</tr>
<tr>
<td>DSP48 slices</td>
<td>237</td>
<td>201</td>
<td>-15%</td>
</tr>
<tr>
<td>18K BRAMs</td>
<td>138</td>
<td>99</td>
<td>-28%</td>
</tr>
</tbody>
</table>

Xilinx Copyright 2012
Summary

- Design Productivity
- Automatically schedules/resource sharing
- Portability/Design Reuse
- Floating point support
- QOR
- Good for algorithmically complex design
MathWorks Support for ZYNQ (Processing System + Programmable Logic)
Compiling Applications to CPU+FPGA

Application
MATLAB/Simulink

MATLAB Turnkey
ZYNQ Support

Embedded Coder
Automatic C/C++ Code Generation

HDL Coder
Automatic RTL Code Generation

CPU (ARM)

Data Movement Interconnect

Memory

HW Accelerator

HW Accelerator

HW Accelerator

Xilinx Copyright 2012
HW / SW Partitioning

Simulink Algorithm

Signal Source → A → B → C → D → View Results

Execute on Processor

ZYNQ TDP

Processor Subsystem

Input Interface → A → B → C → D → Output Interface

Programmable Logic

HW / SW Partition

ZYNQ TDP

Processor Subsystem

Input Interface → A → B → D → Output Interface

Programmable Logic

Xilinx Copyright 2012
Harris-Stephens’ Corner Detection Algorithm

- Corner detection is used in many image processing & computer vision applications
  - image mosaicking
  - object recognition
Measuring Block Execution Time – DSP only

TI C674x DSP @ 300MHz

S6 FPGA @ 75Mhz

- Harris Metric 96.8%
- Find Local Max 3.1%
- Draw Markers 0.1%

- Toggle GPIO at code-block boundaries
- Pulse-width = execution time

Xilinx Copyright 2012
Calculating the Latency of the FPGA’s Harris Metric

- **Acceleration**
  - DSP time / FPGA latency = $1204 / 0.08 = 15,000X$

- TI DSP processing is done sequentially (DSP takes 1.203 secs)
- FPGA processing and UPP transfer occur ***simultaneously***

- **Latency of FPGA Harris Metric**
  - Delay Lines
    - Sobel Gradients (3x3) = 1 line
    - Gaussian Filters (5x5) = 2 lines
    - Total Cycles (including blanking) = 3 * 1056 = 3,168 cycles
  - Additional Cycles (adders, multipliers, etc…) = 28 cycles
  - Total latency = $3,196 \times (1/40MHz) = 80 \text{ usec}$
Summary

- Leverages power of MathWorks modeling environment for DSP and Video Processing
  - MATLAB, Simulink, IP Toolboxes
- Includes automatic code generation for HW / SW
  - HDL Coder, MATLAB Coder, Simulink Coder, Embedded Coder
- Developing New Technology to improve abstraction
  - Turnkey Platform Support
  - System Performance Estimation
  - “One-Click” implementation
- Path to Optimized Results
  - Generated C-code optimized with NEON instructions
  - ARM Intrinsic IP
  - Xilinx DSP IP
# Artix-7 FPGA Product Table

**Artix-7 FPGAs**

Optimized for Lowest Cost and Power with Small Form-Factor Packaging for Highest Volume Applications (1.0V, 0.9V)

<table>
<thead>
<tr>
<th>Logic Resources</th>
<th>Part Number</th>
<th>XC7A100T</th>
<th>XC7A200T</th>
<th>XC7A350T</th>
</tr>
</thead>
<tbody>
<tr>
<td>Slices</td>
<td></td>
<td>15,850</td>
<td>33,650</td>
<td>56,250</td>
</tr>
<tr>
<td>Logic Cells</td>
<td></td>
<td>101,440</td>
<td>215,360</td>
<td>360,000</td>
</tr>
<tr>
<td>CLB Flip-Flops</td>
<td></td>
<td>126,800</td>
<td>269,200</td>
<td>450,000</td>
</tr>
<tr>
<td>Memory Resources</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Maximum Distributed RAM (Kbits)</td>
<td>1,188</td>
<td>2,888</td>
<td>4,638</td>
<td></td>
</tr>
<tr>
<td>Block RAM/FIFO with ECC (36Kbits each)</td>
<td>135</td>
<td>365</td>
<td>515</td>
<td></td>
</tr>
<tr>
<td>Total Block RAM (Kbits)</td>
<td>4,860</td>
<td>13,140</td>
<td>18,540</td>
<td></td>
</tr>
<tr>
<td>Clock Resources</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CMOS (1 MMCM + 1 PLL)</td>
<td>6</td>
<td>10</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>I/O Resources</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Maximum Single-Ended I/O(6)</td>
<td>300</td>
<td>500</td>
<td>600</td>
<td></td>
</tr>
<tr>
<td>Maximum Differential I/O Pairs(6)</td>
<td>144</td>
<td>240</td>
<td>288</td>
<td></td>
</tr>
<tr>
<td>Embedded Hard IP Resources</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DSP48E1 Slices</td>
<td></td>
<td>240</td>
<td>740</td>
<td>1,040</td>
</tr>
<tr>
<td>Embedded Hard IP Resources</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PCI Express®(6)</td>
<td></td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Agile Mixed Signal (AMS) / XADC</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Configuration AES / HMAC Blocks</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>GTP 5.4 / 6.6 Gb/s Transceivers</td>
<td>16</td>
<td>16</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>Speed Grades</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Commercial</td>
<td>-1, -2</td>
<td>-1, -2</td>
<td>-1, -2</td>
<td></td>
</tr>
<tr>
<td>Extended</td>
<td>-2L, -3</td>
<td>-2L, -3</td>
<td>-2L, -3</td>
<td></td>
</tr>
<tr>
<td>Industrial</td>
<td>-1, -2</td>
<td>-1, -2</td>
<td>-1, -2</td>
<td></td>
</tr>
<tr>
<td>Configuration</td>
<td></td>
<td>29.3</td>
<td>62.4</td>
<td>96.1</td>
</tr>
<tr>
<td>Available User I/O: 3.3V SelectIO™ Pins (GTP Transceivers)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Package(6)</td>
<td>Dimensions (mm)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CSG324</td>
<td>15 x 15</td>
<td>210 (0)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FTG256</td>
<td>17 x 17</td>
<td>170 (0)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SBG484</td>
<td>19 x 19</td>
<td>285 (4)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FGG484(2)</td>
<td>23 x 23</td>
<td>285 (4)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FGG676(2)</td>
<td>27 x 27</td>
<td>300 (8)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FFG1156(2)</td>
<td>35 x 35</td>
<td>500 (16)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Notes:**
1. Supports PCI Express Base 2.1 specification at Gen1 and Gen2 data rates.
2. Leaded package options available.
3. Device migration is available within the Artix-7 family for like packages but is not supported between other 7 series families.

---

Xilinx Copyright 2012
### Zynq-7000 Device Table

**HW Designer’s View**

#### Artix Fabric

<table>
<thead>
<tr>
<th>Device Name</th>
<th>Part Number</th>
<th>XC7Z010</th>
<th>XC7Z020</th>
<th>XC7Z030</th>
<th>XC7Z045</th>
</tr>
</thead>
<tbody>
<tr>
<td>Z-7010</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Z-7020</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Z-7030</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Z-7045</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Kintex Fabric

<table>
<thead>
<tr>
<th>Device Name</th>
<th>Part Number</th>
<th>XC7Z010</th>
<th>XC7Z020</th>
<th>XC7Z030</th>
<th>XC7Z045</th>
</tr>
</thead>
<tbody>
<tr>
<td>Z-7010</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Z-7020</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Z-7030</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Z-7045</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Zynq™-7000 Extensible Processing Platform

<table>
<thead>
<tr>
<th>Processing System</th>
<th>Artix™-7 FPGA</th>
<th>Artix™-7 FPGA</th>
<th>Kintex™-7 FPGA</th>
<th>Kintex™-7 FPGA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Programmable Logic Cells (Approximate ASIC Gates)</td>
<td>28K Logic Cells (~430K)</td>
<td>65K Logic Cells (~1.3M)</td>
<td>125K Logic Cells (~1.9M)</td>
<td>350K Logic Cells (~5.2M)</td>
</tr>
<tr>
<td>Logic Cells</td>
<td>28,160</td>
<td>85,120</td>
<td>125,760</td>
<td>349,760</td>
</tr>
<tr>
<td>Look-Up Tables LUTs</td>
<td>17,600</td>
<td>53,200</td>
<td>78,600</td>
<td>218,600</td>
</tr>
<tr>
<td>Flip Flops</td>
<td>35,200</td>
<td>106,400</td>
<td>157,200</td>
<td>437,200</td>
</tr>
<tr>
<td>Extensible Block RAM (# 36 Kb Blocks)</td>
<td>240 KB (60)</td>
<td>560 KB (140)</td>
<td>1,060 KB (265)</td>
<td>2,180 KB (545)</td>
</tr>
<tr>
<td>Programmable DSP Slices (18x25 MACCs)</td>
<td>80</td>
<td>220</td>
<td>400</td>
<td>900</td>
</tr>
<tr>
<td>Gen2 Performance (Symmetric FIR)</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>PCI Express® (Root Complex or Endpoint)</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>Agile Mixed-Signal (AMS) / XADC</td>
<td>2x 12 bit, 1 MSPS ADCs with up to 17 Differential Inputs</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>Multi-Standards and Multi-Voltage SelectIO™ Interfaces</td>
<td>—</td>
<td>—</td>
<td>63</td>
<td>150</td>
</tr>
<tr>
<td>Multi-Standards and Multi-Voltage High Performance SelectIO™ Interfaces</td>
<td>—</td>
<td>—</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>Security</td>
<td>AES and SHA 256 for secure configuration</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
</tbody>
</table>

#### Notes:

1. Security is shared by the Processing System and the Programmable Logic.
2. Equivalent ASIC gate count is dependent of the function implemented. The assumption is 1 Logic Cell ~ 15 ASIC Gates.
3. Preliminary product information. Subject to change. Please contact your Xilinx representative for the latest information.

Xilinx Copyright 2012
# Zynq-7000 Device Table
## SW Developer’s View

## Artix Fabric

<table>
<thead>
<tr>
<th>Device Name</th>
<th>Part Number</th>
<th>Processor Core</th>
<th>Processor Extensions</th>
<th>Maximum Frequency</th>
<th>L1 Cache</th>
<th>L2 Cache</th>
<th>On-Chip Memory</th>
<th>On-Chip Memory</th>
<th>External Memory Support</th>
<th>External Static Memory Support</th>
<th>DMA Channels</th>
<th>Peripherals</th>
<th>Peripherals and Static Memory Multiplexed I/Os</th>
<th>Security</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zynq-7000 Extensible Processing Platform</td>
<td>Zynq™-7000 Extensible Processing Platform</td>
<td>Dual ARM® Cortex™-A9 MPCore™ with CoreSight™</td>
<td>NEON™ &amp; Single / Double Precision Floating Point</td>
<td>800 MHz</td>
<td>32 KB Instruction, 32 KB Data per processor</td>
<td>512 KB</td>
<td>256 KB</td>
<td>DDR3, DDR2, LPDDR2</td>
<td>2x Quad-SPI, NAND, NOR</td>
<td>2x USB 2.0 (OTG) w/DMA, 2x Tri-mode Gigabit Ethernet w/DMA, 2x SD/SDIO w/DMA, 2x UART, 2x CAN 2.0B, 2x I2C, 2x SPI, 4x 32b GPIO</td>
<td>2x Programmable Logic Ports</td>
<td>AES and SHA 256b for secure boot</td>
<td>54</td>
<td></td>
</tr>
</tbody>
</table>

### Processing System

- **Zynq-7000 Extensible Processing Platform**
  - **Zynq™-7000 Extensible Processing Platform**
  - **Zynq™-7000 Extensible Processing Platform**
  - **Zynq™-7000 Extensible Processing Platform**

### Programmable Logic

- **Zynq™-7000 Extensible Processing Platform**
  - **Zynq™-7000 Extensible Processing Platform**
  - **Zynq™-7000 Extensible Processing Platform**
  - **Zynq™-7000 Extensible Processing Platform**

### Notes:
1. Static memory interface combined with the usage of many peripherals could require more than 54 I/Os. A designer can use the Programmable Logic I/Os.
2. Total Number of I/O and Transceivers depends on package used.
3. Equivalent ASIC gate count is dependent of the function implemented. The assumption is 1 Logic Cell = ~15 ASIC Gates.
4. Preliminary product information. Subject to change. Please contact your Xilinx representative for the latest information.

---

Xilinx Copyright 2012
Video and Image Processing IP Pack

*Bundled IP Pack Providing Savings and Convenience*

- Single IP core bundle providing low cost option for collection of IP
- Xilinx Deinterlacer is included
- Price $3k

**Video and Image Processing Pack**

- Defective Pixel Correction
- Color Filter Array Interpolation
- Gamma Correction
- Color Correction Matrix
- Edge Enhancement
- Noise Reduction (2D)
- Image Statistics
- Chroma Resampler
- Video Scaler
- On-Screen Display
- Timing Controller
- Motion Adaptive Noise Reduction (3D)
- Image Characterization
- Object Segmentation
- Deinterlacer

Xilinx Copyright 2012
Comprehensive DSP Design Platform - V6 DSP48E1

Power Consumption Benefits
- Lowest power operation of any FPGA solution
- 1.23mW/100Mz at 38% toggle rate

Performance Benefits
- 600MHz operations for any DSP operation
- ~1.2 TeraMAC/s in a single device

Cost Benefits
- Hardened pre-adder and adder cascade saves significant resources
- Logic functions can be mapped into DSP blocks
The XtremeDSP Slice

- Xilinx DSP FPGAs include dedicated DSP processing blocks called DSP48s
  - Spartan-6: DSP48A1 (18 bit pre-adder + 18x18 multiplier)
  - Virtex-6 DSP48E1 (25 bit pre-adder + 18x25 multiplier)
- Blocks individually configurable to perform over 30 unique arithmetic functions
  - Dynamic operational modes adapts DSP48 slice from clock to clock
- V6 DSP48E1 has expanded capabilities compared to S6 DSP48A1

V6 DSP48E1 or S6 DSP48A1 = One Multiplier + 300 Logic Cells

- 300 is typical
- actual range dependent upon application (range 50-1300 logic cells saved)
How Else Can the DSP48E1 Be Utilized?

- 48 bit compare, absolute value
- Counter (count limited and free running)
- 3 input 48 bit adder, subtract
- 25x18 bit multiply
- 25x25 bit pre-adder, 48 bit accumulator
- Cascade support that allows for larger mults or adds
- SIMD mode – 4x12 bit, 2x24, or 2x48 bit add or subtract using a single DSP48 slice (SAD operations!)
- Bus Multiplexer
- Shifters (left, right, barrel)
- Bit operations (and, or, not, nand, nor, xor, xnor, magnitude compare, equal to zero, greater than zero, less than zero)
- Complex multiply (4 clock cycles in one DSP48E1)
- Overflow, underflow reporting, convergent rounding support, symmetric rounding support, random rounding, terminal count detection support and auto resetting, saturation
- Supports block floating point operations
- Divide (via successive mults or shift and subtract operations), square root operations (successive mults and subs).
- Filters!