Hybrid CPU/FPGA Computing and Applications

Achieving Performance, Productivity, Portability

Michael S. Babst, Ph.D.
President
301-977-5970 x705
mike.babst@dsplcogic.com
Reconfigurable Computing Mission

Achieve Maximum 

Algorithm Acceleration

with Minimum Investment

Key Requirements of any RC Development Flow:

- Performance
- Productivity
- Portability
Sweet RC Dreams…

I am so happy, my C code runs 100x faster without changing one line of code. It runs on everyone’s reconfigurable computer, so I can easily compare them and just buy the fastest one! It even runs on both Xilinx *and* Altera FPGAs!
RC Fears

- *What* kind of C do I have to learn?
- Can’t you partition my algorithm for me?
- Fixed point? You must be kidding!
- Did you say – VHDL?
CPU vs. FPGA Computing

```c
struct s_point pointprojection(struct s_plane plane, struct s_point p1)
{
    // Get the projection of point p1 on the plane
    v = p1 - plane.p; result = v - (v*plane.n)plane.n + plane.p
    //
    double temp;
    struct s_point vec1, proj;
    //
    //
    vec1.x = p1.x-plane.p.x;
    vec1.y = p1.y-plane.p.y;
    vec1.z = p1.z-plane.p.z;
    //
    // temp = vec1 dot plane.n
    //
    temp = plane.n.x*vec1.x + plane.n.y*vec1.y + plane.n.z*vec1.z;
    //
    // Get the global coordinates for p1's projection
    //
    proj.x = vec1.x - temp*plane.n.x + plane.p.x;
    proj.y = vec1.y - temp*plane.n.y + plane.p.y;
    proj.z = vec1.z - temp*plane.n.z + plane.p.z;
    return proj;
}
```

CPU: One Big, Fast Engine

FPGA: Many Small, Slow, Specialized Engines
Hybrid CPU/FPGA Platforms and Applications

- **Intelligence / Understanding**
- **Knowledge**
- **Information**
- **Data**
- **Signals**

**High Performance Computing**
- Informatics
- Cognitive computing
- Software acceleration
- Finance
- Data Fusion
- Cryptography
- Software Defined Radio
- Wireless/Satellite communications
- Signal Processing
- Sensor data acquisition

**Desktop Computing**

**Portable Computing**

**Embedded Computing**

**RC Toolbox Overview**
Success Depends Upon…

Algorithm Understanding

Interfaces

Programming and Debugging

Compilation

Implementation
What is the RC Toolbox?

- Programming Environment
  - MATLAB/Simulink under Windows

- Target Runtime Environment
  - Linux, Windows
  - Matlab/Simulink Not Required
  - Platform Support Packages

General Purpose
High-Level Programming Language

RC Debugging Toolbox

RC Platform Builder

SW API Library

HAL API Library

MATLAB/Simulink under Windows

Linux, Windows

Matlab/Simulink Not Required

Platform Support Packages

RC Toolbox Overview
### Interface Abstraction for Portability

- **How will you call the Accelerated Function?**

<table>
<thead>
<tr>
<th>Function</th>
<th>OpenFPGA GenAPI</th>
<th>RCIO API</th>
</tr>
</thead>
<tbody>
<tr>
<td>Setup Accelerator Function Call</td>
<td>ofpga_init</td>
<td>rcio_init</td>
</tr>
<tr>
<td></td>
<td>ofpga_load_algorithm</td>
<td>rcio_config</td>
</tr>
<tr>
<td></td>
<td>ofpga_status</td>
<td>rcio_status</td>
</tr>
<tr>
<td></td>
<td>ofpga_close</td>
<td>rcio_close</td>
</tr>
<tr>
<td></td>
<td>ofpga_run</td>
<td></td>
</tr>
<tr>
<td>Send/Receive Data to Accelerated Function</td>
<td>ofpg_send</td>
<td>rcio_send</td>
</tr>
<tr>
<td></td>
<td>ofpga_receive</td>
<td>rcio_receive</td>
</tr>
<tr>
<td></td>
<td></td>
<td>rcio_stream</td>
</tr>
<tr>
<td>Send/Receive Constants to Accelerated Function</td>
<td>ofpga_write_register</td>
<td>rcio_writeparam</td>
</tr>
<tr>
<td></td>
<td>ofpga_read_register</td>
<td>rcio_readparam</td>
</tr>
</tbody>
</table>
Interface Abstraction for Portability

- **Host Application**
  - SW-API Library
  - Messages: Vectors, bulk data, etc...
  - Parameters: Variables, constants, etc.
  - send(), receive(), writeparam(), readparam()

- **Vendor/Platform Specific Hardware, Software and Interconnect**
  - Messages between Host Application and Accelerator Application
  - Parameters between Host Application and Accelerator Application

- **HAL-API Library**
  - Input Messages
  - Output Messages

- **Accelerator Application**
  - Input Parameters
  - Output Parameters

- **Host CPU**
- **Accelerator FPGA**
Interface Abstraction - HAL

- How will you program the Accelerated Function?

  - HAL Interface
    - To/From CPU

  - User Application

  - Re-usability, Upgradeability, Portability
    - High-Level Programming Languages
    - Optimized IP Libraries

  - Happy Programmers and Scientists

  - Sad Programmers and Scientists

  - Platform Details
  - Signal Timing

  - Headaches
  - Happy Hardware Engineers!

  - Happy Hardware Engineers!

RC Toolbox Overview
Reconfigurable Computing Toolbox

- **High-Level, General Purpose**
  - Graphical Programming Language
  - Providing:
    - High Performance
      - 200 MHz Typical Clock Speeds on Virtex 2 and Virtex 4
      - Easily express of parallelism and pipelining
    - High Productivity
      - High-Level Programming Abstraction
      - Low learning curve
      - Debugging within Matlab
      - Optimized Bitstream Generation within MATLAB™
    - High Portability
      - Interface Abstraction
      - Multiple CPU(s) /FPGA(s)
      - Embedded, portable, desktop, HPC
  - Program Flow
    - sequential, parallel, and pipelined
  - Math
    - (including floats)
  - Parallel Memory Access
  - Portable RC Abstraction Layers

- **RC Toolbox Overview**
- **Integration with other technologies**
  - IP Cores
  - HDL
  - Other HLLs

- **RC Debugging Toolbox**
  - Program validation, data visualization

- **RC Platform Builder**
  - Fully automated bitstream compilation
  - Verified, robust, high-performance bitstream
Program, Debug, and Validate entire Accelerated Function within Matlab and Simulink.

- Simulate / Validate FPGA core
  - Only User Application Needs to be Simulated

- Generate Function inputs and Analyze Outputs in Matlab
  - Convenient use of standard Matlab vector/matrix handling, Plotting, etc.

- Debug for all target platforms simultaneously
  - Target Runtime Platform is Selectable in Platform Builder
  - Guaranteed operation up to SW API level

- Target/Runtime Environment
  - Windows or Linux Platform
  - Does not require Matlab, Simulink, Windows
High-Level Graphical Programming

For Performance, Productivity, and Portability
Multiple FPGA Programming Abstractions
- High level for domain experts and programmers
- Low level for FPGA experts

Programming flow
- Sequential, parallel, pipelining constructs
- if-then-else, conditional branches
- Floating-point Operations

Complex functions
- FFTs, Computation libraries

Simple functions
- Fixed-point math operations, counters

FPGA Resources
- Registers, Block RAM, HW Multipliers, slices, flip-flops
Sequential Programming

- Example main() program

```c
// Equivalent pseudo code
outputs main(inputs){
    temps = function1(inputs);
    outputs = function2(temps);
}
```

- Hierarchical variable scope
- Data Types Automatically Propagated
- Variables with Global and Local Scope Minimize Clutter
// Equivalent pseudo code

parallel_function(inputs, outputs1, outputs2){
    #pragma parallel
    outputs1 = function1(inputs);
    outputs2 = function2(inputs);
}

parallel_function()
Conditional Branch (if-then-else)

- If condition is true, initiate a task
  - otherwise, initiate an alternate task
- Requires Combine block, i.e. “}”

// Equivalent pseudo code
if (a xx b) {
do_this();
} else {
do_that();
}
Parallel vs. Branch/Switch operations

Parallel Function

More Efficient Resource Usage

Switch

RC Toolbox Overview
// Equivalent software operation

for (k=10; k>=0; k=k-1) {
    loop_function(index);
}

Function Block Parameters: Loop

Loop controller - Create a loop process, equivalent to the C statement:
for (INDEX = initValue; INDEX <= limitValue; INDEX = INDEX + increment)
The following relational operators may be used to determine the end of the loop:
<, >, <=, >=

Loop parameters may be fixed or variable.
When Loop Parameters = 'Fixed', all loop parameters are entered into the block mask.
When Loop Parameters = 'Use Inputs', The Minimum and Maximum Loop Count should be
set greater than the largest range of expected input values. Using smaller values for
Minimum and Maximum Loop Count will conserve hardware resources.

NEXT will be asserted each time a new loop iteration starts. The value of NEXT is
persistent and will be valid during the entire loop process.
There is a 1 cycle latency between START and the first NEXT assertion.
There is a 1 cycle latency between CONTINUE and NEXT.

Parameters:

- Loop Parameters: Fixed
- Initial Value: 10
- Limit Value: 0
- Increment: -1
- Limit Value Comparison Type: Greater than or equal to
Pipelined Loop

// Equivalent software operation

for (k=0;k<loop_count;k++){
    loop_function(data1);
}

- Simple loop execution
  - ‘Loop’ initiates execution of ‘loop_function’ in accordance with loop equation
  - ‘K’ is updated and valid on each ‘loop_function/start’

- Pipelined
  - ‘Loop’ will start ‘loop_function’ on consecutive clock cycles until loop is complete or a ‘break’ event occurs.

- Uses
  - When maximum loop execution speed is required
  - When loop_function() has no data dependencies, or well-known dependencies.

- Break
  - ‘loop_function’ can force a break from loop any time

- Automatic pipeline latency handling
  - ‘Pipeline Loop/done’ is not asserted until ‘loop_function’ pipeline is completed.

// Equivalent software operation

for (k=0;k<loop_count;k++){
    loop_function(data1);
}
Variable read/write operations

- Declaration
- Write Access
  - Select Port
  - Variable type automatically determined
  - All ports must write same type
- Read Access
Memory / Arrays

- Declaration

- Write Access
  - Select Port
  - Variable type automatically determined
  - All ports must write same type

- Read Access
Example Function

Reference function

```matlab
function indx = testfunction(x1,y1,z1);

lcount = length(x1);
for j = 1:lcount
    for k = 1:lcount
        indx(j,k) = x1(j)*x1(k) + y1(j)*y1(k) + z1(j)*z1(k);
    end
end
```
This simple program performs 6 memory reads, 3 Floating-point Multiplies and 2 Floating Point Adds in Parallel and Fully Pipelined (once every clock cycle)
You Can Do It!

- Bioinformatics
- Seismic Processing
- Molecular Dynamics
- Crash Simulation
- Cryptography
- Database searching
- Finance
- Astrophysics
- Remote Sensing
- Signal and Image Processing
- Traffic Simulation

Not just for DSP!
Contact Information

For Additional Information, Please Contact:

Michael S. Babst, Ph.D.
President
301-977-5970 x705
mike.babst@dspllogic.com

DSPlogic
1-877-DSP-LOGIC
www.dspllogic.com