Tools for Reconfigurable Supercomputing

Kris Gaj
George Mason University
Application Development for Reconfigurable Computers

Program Entry

Platform mapping

Compilation

Debugging & Verification

Execution
Tasks Addressed in This Presentation

- Program Entry
- Platform mapping
- Compilation
- Debugging & Verification
- Execution
Program Entry

Program
Platform Mapping
SW/HW Partitioning

Program

Software
(executed in the microprocessor system)

Hardware
(executed in the reconfigurable processor system)
SW/HW Partitioning & Coding
Traditional Approach

Specification → SW/HW Partitioning →

<table>
<thead>
<tr>
<th>SW Coding</th>
<th>HW Coding</th>
</tr>
</thead>
<tbody>
<tr>
<td>SW Compilation</td>
<td>HW Compilation</td>
</tr>
<tr>
<td>SW Profiling</td>
<td>HW Profiling</td>
</tr>
</tbody>
</table>
SW/HW Partitioning & Coding

New Approach

Specification

SW/ HW Coding

SW/ HW Partitioning

SW Compilation

SW Profiling

HW Compilation

HW Profiling
Program Entry for FPGA Accelerator Boards

Traditional
Software
Hardware

Extended
Software
Hardware

HDL
Graphical Data Flow Diagram
HLL

Increased productivity
Increased capability to describe parallel execution
Program Entry for Reconfigurable Computers

Star Bridge

Software

Hardware

Graphical Data Flow Diagram

COM objects

porting

EDIF

SRCHardware

Software

Increased productivity

Increased capability to describe parallel execution

HDL macros

HDL

Increased productivity

HLL
Examples of Software Environments for Reconfigurable Computers

DSP-oriented

• Corefire from Annapolis Microsystems

• Xtreme DSP from Xilinx Inc. & MathWorks

General-purpose

• Viva from Star Bridge Systems

• SRC Software Environment from SRC Computers, Inc.
CoreFire FPGA Application Builder

- Design viewer
- Library
- Cores window
- Message Window
- Diagram Editor
Xtreme DSP Environment

1. DSP System Modeling
   - The MathWorks
   - MATLAB® / Simulink®

2. System Generation
   - XILINX®
   - System Generator for DSP

3. HDL Synthesis
   - Synplicity
   - Synplify Pro
   - Mentor Graphics®
   - FPGA Advantage

4. Simulation (optional)
   - Mentor Graphics®
   - FPGA Advantage

5. FPGA Implementation
   - XILINX®
   - ISE
Star Bridge Software Environment

Graphical User Interface

User input

VIVA

Netlists

Place & Route

.Xilinx

 Application executable

.bit files

.Configuration bitstreams
SRC Compilation Process

Application sources
- .c or .f files
  - MAP Compiler
  - .o files
  - Linker
    - Application executable

Macro sources
- .vhd or .v files
  - Logic synthesis
  - .v files
  - Netlists
    - .ngo files
    - Place & Route
      - .bin files

Object files
- .o files
  - μP Compiler
  - MAP Compiler
Cray XD1 - Traditional Design Flow

**HDL**
- VHDL, Verilog, C

**Synthesize**
- Synplicity, Leonardo, Precision, Xilinx ISE

**Implement**
- Xilinx ISE

**Simulate**
- Modelsim

**Download**
- Binary File
  - Metadata: 0100101010 1010101111 0100101011 0101011010 1001110101 0110101010

**Verify**
- Xilinx ChipScope

Source: [Cray, MAPLD04]
Cray XD1 – New design flow

Source: [Cray, MAPLD04]
SGI Altix Design Flow
(HDLs)

Design Entry
(Verilog, VHDL)

Design Synthesis
(Synplify Pro, Amplify)

Design Implementation
(ISE)

Behavioral Simulation
(VCS, Modelsim)

Static Timing Analysis
(ISE Timing Analyzer)

Real-time Verification
(gdb)

Metadata Processing
(Python)

Device Programming
(RASC Abstraction Layer,
Device Manager, Device Driver)

Design Verification

IA-32 Linux Machine

Altix
SGI Altix Design Flow (HLLs)

- **HLL Design Entry** *(Handel-C, Impulse C, Mitrion C, Viva)*
- **RTL Generation and Integration with Core Services**
- **Design Synthesis** *(Synplify Pro, Amplify)*
- **Design Implementation** *(ISE)*
- **Device Programming** *(RASC Abstraction Layer, Device Manager, Device Driver)*
- **Behavioral Simulation** *(VCS, Modelsim)*
- **Static Timing Analysis** *(ISE Timing Analyzer)*
- **Real-time Verification** *(gdb)*

**Metadata Processing** *(Python)*

**IA-32 Linux Machine**

---

19
Platform Mapping

FPGA mapping

Program

Hardware

FPGA 1

FPGA 2

FPGA 3

FPGA 4
Example of FPGA Mapping

```
multiply  divide
  add
```

```
multiply
  add
```

```
divide
```

```
multiply  divide
  add
```

```
multiply  divide
  add
```

```
multiply
  add
```

21
FPGA Mapping in SRC

FPGA1.mc

```c
void fpga1(int64_t a, int64_t b,
           int64_t *sum, int mapno)
{
    int64_t c, temp;

    send_to_bridge(b);
    c = a * const1;
    recv_from_bridge(&temp);
    *sum = temp+Mult;
}
```

FPGA2.mc

```c
void fpga2()
{
    int64_t a, d;

    recv_from_bridge(&a);
    d = a/const2;
    send_to_bridge(d);
}
```

Makefile

```makefile
MAPFILES = FPGA1.mc FPGA2.mc
PRIMARY    = FPGA1.mc
SECONDARY  = FPGA2.mc
CHIP2      = FPGA2.mc
```

Diagram:

- FPGA 1
  - Multiply
  - Add
  - Sum

- FPGA 2
  - Divide

Connections:
- a -> Multiply
- b -> Multiply
- Multiply -> Add
- Add -> Sum
- Sum -> Divisor
- d -> Divisor
FPGA Mapping in VIVA™

By changing the attributes one can specify where an object is to be located.
Platform Mapping
FPGA-FPGA data transfer & synchronization

Program

Hardware

Software

FPGA 1

FPGA 2

FPGA 3

FPGA 4
**FPGA-FPGA Data Transfer in SRC**

**FPGA1.mc**

```c
void fpga1(int64_t a, b, c, *d)
{
    send_to_bridge(a, b, c);
    computation1
    recv_from_bridge(d);
}
```

**FPGA2.mc**

```c
void fpga2()
{
    int64_t a, b, c, d;

    recv_from_bridge(&a, &b, &c);
    computation2
    send_to_bridge(d);
}
```
FPGA-FPGA Data Transfer in SRC

Bridge Port
FPGA-FPGA Data Transfer in VIVA™

Special partitioning objects placed between the modules to be synthesized automatically map the relevant lines between the FPGAs.

For designs mapped over several FPGAs: The system description must include those FPGAs over which the design is to be mapped,
Platform Mapping
Use of Internal and External Memories

Software

Program

Hardware

FPGA 1

FPGA 2

FPGA 3

FPGA 4

OCM

LM

SM

OCM – On-Chip Memory
LM – Local Memory
SM – Shared Memory

28
void sum(int64_t a[], int *c, int mapno)
{
    BANK_A_ALLOC(AL, int64_t, SIZE);
    ocm_a [SIZE];
    int i;
    cm2obm_0(AL, a, byteLength);
    wait_server_0();
    for(i=0; i<SIZE; i++)    {
        ocm_a[i] = AL[i]; }
    for(i=0; i<SIZE; i++)    {
        tmp = ocm_a[i] + tmp; }
}
Using On-Chip Memory (OCM) in VIVA™

Special Objects under the Memory Subsystem of the library allows the programmer to use the on chip memory of the Xilinx Virtex II chip.
Run Time Reconfiguration in SRC

Program in C or Fortran

Main program

......

Function_1(a, d, e)

......

Function_2(d, e, f)

FPGA contents after the Function_1 call

FPGA

Macro_1(a, b, c)

Macro_2(b, d)

Macro_2(c, e)

Macro_3(s, t)

Macro_1(n, b)

Macro_4(t, k)

Macro_1

Macro_2

Macro_2

Macro_2
Run-time Reconfiguration in VIVA™

Reconfiguration is possible by using the spawn object.

By specifying the FileName attribute a VIVA executable (.vex file) or a VIVA project can be loaded onto the same or a different FPGA.
Ideal Program Entry

Function

Program Entry
Actual Program Entry

- Preferred Architectures
- Function
- SW/HW Partitioning
- Use of FPGA Resources (multipliers, µP cores)
- Sequence of Run-time Reconfigurations
- FPGA Mapping
- Data Transfers & Synchronization
- SW/HW Interface
- Use of Internal and External Memories

35
### Evolution and the current status of tools

<table>
<thead>
<tr>
<th>SRC</th>
<th>Star Bridge and other vendors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not implemented</td>
<td>Manual Entry</td>
</tr>
<tr>
<td></td>
<td>Compiler Automated</td>
</tr>
<tr>
<td></td>
<td><strong>μP-FPGA Partitioning</strong></td>
</tr>
<tr>
<td></td>
<td><strong>FPGA-FPGA Partitioning</strong></td>
</tr>
<tr>
<td></td>
<td><strong>μP-FPGA Data Transfer</strong></td>
</tr>
<tr>
<td></td>
<td><strong>FPGA-FPGA Data Transfer</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Computation-Data transfer Overlapping</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Choosing component version</strong></td>
</tr>
</tbody>
</table>

...
Library Development - SRC

- μP system
  - LLL (ASM)
  - HDL (VHDL, Verilog)
  - Library Developer

- FPGA system
  - HLL (C, Fortran)
  - HLL (C, Fortran)
  - Application Programmer
Library Development - StarBridge

μP system

HLL, LLL (C++, ASM)

GDF (Viva)

FPGA system

HDL (VHDL, Verilog)

GDF (Viva)

Library Developer

Application Programmer
Debugging & Verification
CoreFire™ FPGA Application Debugger
Corefire Simulation

- Insert Debug Modules During Design Editing
- Step Through Design Using Data Flow
- One Step = One Module
- View Value and Status of Each Debug Module
- Waveform or Table of Values
- Read and Write Directly to Registers
- Read and Write Directly to Memory
X86 System in VIVA™

The Fileln Object as it appears when the x86 system is loaded
X86 System in VIVA™

FileIn object as it appears when the FPGA system description is loaded.
Data can be viewed with the help of widgets, which are basically input and output ‘horns’ placed in a worksheet.

Various display options are available to view data, options to include the kind of view desired by the viewer and the data viewed can be switched between HEX or INT.
MAP Board Execution
MAP Emulator + DFG Simulator
MAP Emulator + Verilog Simulator
## Summary – Program Entry

<table>
<thead>
<tr>
<th></th>
<th>SRC</th>
<th>Star Bridge</th>
<th>Xtreme DSP</th>
<th>Corefire</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Program entry model</strong></td>
<td>HLL</td>
<td>GDF</td>
<td>HLL, GDF</td>
<td>HLL, GDF</td>
</tr>
<tr>
<td><strong>Programming languages</strong></td>
<td>HLL</td>
<td>C, Fortran</td>
<td>Matlab</td>
<td>Java</td>
</tr>
<tr>
<td><strong>HDL</strong></td>
<td>VHDL, Verilog</td>
<td>EDIF</td>
<td>VHDL, Verilog</td>
<td>?</td>
</tr>
</tbody>
</table>
### Summary - Partitioning & Data Transfer

<table>
<thead>
<tr>
<th>SRC</th>
<th>Star Bridge</th>
<th>Xtreme DSP</th>
<th>Corefire</th>
</tr>
</thead>
<tbody>
<tr>
<td>FPGA Mapping</td>
<td>Separate HLL functions</td>
<td>System attributes of objects</td>
<td>Separate design sheets</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Separate design sheets</td>
</tr>
<tr>
<td>FPGA-FPGA</td>
<td>send-to bridge, recv-from bridge</td>
<td>special data transfer objects</td>
<td>interface library components</td>
</tr>
<tr>
<td>Data Transfer</td>
<td>macro</td>
<td>such as PE1 =&gt;PE2_50</td>
<td>interface library components</td>
</tr>
</tbody>
</table>
Summary – Synchronization

<table>
<thead>
<tr>
<th>SRC</th>
<th>Star Bridge</th>
<th>Xtreme DSP</th>
<th>Corefire</th>
</tr>
</thead>
<tbody>
<tr>
<td>Implicit</td>
<td>Explicit: Go-Done-Busy-Wait</td>
<td>Explicit: done, empty, full, etc.</td>
<td>Implicit</td>
</tr>
</tbody>
</table>
## Summary - Run-time reconfiguration

<table>
<thead>
<tr>
<th>Run-time reconfiguration</th>
<th>SRC</th>
<th>Star Bridge</th>
<th>Xtreme DSP</th>
<th>Corefire</th>
</tr>
</thead>
<tbody>
<tr>
<td>sequence of MAP function calls</td>
<td>using spawn objects associated with VIVA executables</td>
<td>?</td>
<td>?</td>
<td></td>
</tr>
</tbody>
</table>
Summary - Use of Internal Resources

<table>
<thead>
<tr>
<th>Using internal components of FPGA</th>
<th>SRC</th>
<th>Star Bridge</th>
<th>Xtreme DSP</th>
<th>Corefire</th>
</tr>
</thead>
<tbody>
<tr>
<td>block RAM’s</td>
<td>Arrays defined inside MAP functions</td>
<td>Special objects under the memory subsystem</td>
<td>Memory block sets</td>
<td>Memory operators and resources</td>
</tr>
<tr>
<td>multipliers</td>
<td>Library functions</td>
<td>Objects under arithmetic subsystem</td>
<td>Math block sets</td>
<td>Modules in math library</td>
</tr>
</tbody>
</table>
# Summary - Data Types

<table>
<thead>
<tr>
<th>Data types</th>
<th>SRC</th>
<th>Star Bridge</th>
<th>Xtreme DSP</th>
<th>Corefire</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unsigned integers</td>
<td>8,16,32,64 bits</td>
<td>1-128 bits</td>
<td>-</td>
<td>32, 64 bits</td>
</tr>
<tr>
<td>Signed integers</td>
<td>8,16,32,64 bits</td>
<td>1-128 bits</td>
<td>-</td>
<td>8 bits</td>
</tr>
<tr>
<td>Fixed point</td>
<td>-</td>
<td>fix16, fix32</td>
<td>signed, unsigned</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>variable size</td>
<td></td>
</tr>
<tr>
<td>Floating point</td>
<td>Single &amp;</td>
<td>32-bit</td>
<td>64-bit</td>
<td>32-bit</td>
</tr>
<tr>
<td></td>
<td>double</td>
<td>single precision</td>
<td>double precision</td>
<td>single precision</td>
</tr>
<tr>
<td>Arrays</td>
<td>8, 16, 32, 64</td>
<td>vectors of</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>bit</td>
<td>1-128 bits</td>
<td></td>
<td></td>
</tr>
<tr>
<td>User defined</td>
<td>-</td>
<td>yes</td>
<td>user-defined</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>types</td>
<td></td>
<td>precision options</td>
<td></td>
</tr>
<tr>
<td>Complex</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>16-bit signed</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>integers, float</td>
</tr>
</tbody>
</table>
## Summary - Libraries

<table>
<thead>
<tr>
<th>Libraries</th>
<th>SRC</th>
<th>Star Bridge</th>
<th>Xtreme DSP</th>
<th>Corefire</th>
</tr>
</thead>
<tbody>
<tr>
<td>arithmetic</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>logic</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>storage</td>
<td>implicit</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>memory allocation and access</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>control</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>-</td>
</tr>
<tr>
<td>data transfer</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>debugging and profiling</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>DSP</td>
<td>-</td>
<td>-</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>communication</td>
<td>-</td>
<td>-</td>
<td>yes</td>
<td>-</td>
</tr>
<tr>
<td>User defined components</td>
<td>Macros in VHDL or Verilog</td>
<td>Objects in VIVA</td>
<td>Black boxes in Simulink</td>
<td>Macros in Corefire</td>
</tr>
</tbody>
</table>
## Summary – Debugging & Verification

<table>
<thead>
<tr>
<th>Debugging and verification</th>
<th>SRC</th>
<th>Star Bridge</th>
<th>Xtreme DSP</th>
<th>Corefire</th>
</tr>
</thead>
<tbody>
<tr>
<td>software emulation</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>HDL simulation</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>no</td>
</tr>
</tbody>
</table>
# Summary - Third Party Tools & I/O

<table>
<thead>
<tr>
<th>Use of external tools</th>
<th>SRC</th>
<th>Star Bridge</th>
<th>Xtreme DSP</th>
<th>Corefire</th>
</tr>
</thead>
<tbody>
<tr>
<td>logic synthesis</td>
<td>Synplicity Synplify Pro</td>
<td>-</td>
<td>Synplicity Synplify Pro, Mentor Graphics Leonardo Spectrum, Xilinx XST</td>
<td>-</td>
</tr>
<tr>
<td>MAP, PAR</td>
<td>Xilinx ISE</td>
<td>Xilinx ISE</td>
<td>Xilinx ISE</td>
<td>Xilinx ISE</td>
</tr>
<tr>
<td>µp compilation</td>
<td>Intel</td>
<td>-</td>
<td>Matlab</td>
<td>-</td>
</tr>
<tr>
<td>Schematic capture</td>
<td>-</td>
<td>VIVA</td>
<td>Simulink</td>
<td>Application Builder</td>
</tr>
<tr>
<td>HDL simulation</td>
<td>VCS</td>
<td>-</td>
<td>ModelSim</td>
<td>-</td>
</tr>
<tr>
<td>Components of the environment</td>
<td>Editors, compiler, DFG behavioral simulator, VCS HDL simulator</td>
<td>VIVA (program entry, compiler, debugging and verification, execution environment)</td>
<td>Simulink program entry, HDL simulator, synthesis compiler, place and route tool</td>
<td>Application builder, debugger</td>
</tr>
<tr>
<td>Input and output</td>
<td>standard HLL i/o functions, files</td>
<td>widgets, files</td>
<td>files, block sets</td>
<td>files, waveforms, tables</td>
</tr>
</tbody>
</table>
Acknowledgements

- SRC
- Star Bridge
- Sashisu Bajracharya (GMU)
- Esmail Chitalwala (GWU)
- Esam El-Araby (GWU)
- Miaoaqing Huang (GWU)
- Allen Michalski (USC)
- Nghi Nguyen (GMU)
- Proshanta Saha (GWU)
- Nandkishore Sastry (GMU)
- Chang Shu (GMU)
- Mohamed Taher (GWU)