The Self-Describing Wishbone Bus (SDWB)

Manohar Vanga

BE-CO-HT, CERN, Geneva

Jule 13, 2007
Introduction

The Hardware

Device Drivers

Self-Describing Wishbone Bus (SDWB)

Future Work

About

Indian, studying in Spain, working in Switzerland (CERN), living in France

Hardware & Timing section (Beam Controls group) @ CERN

Develop timing & data acquisition hardware for big toys

Open Hardware Repository (http://ohwr.org)

VME and PCI hardware (FPGA based)

I work on Linux device drivers

Manohar Vanga
About

- Indian
About

- Indian, studying in Spain
Indian, studying in Spain, working in Switzerland (CERN)
Indian, studying in Spain, working in Switzerland (CERN), living in France
About

- Indian, studying in Spain, working in Switzerland (CERN), living in France
- Hardware & Timing section (Beam Controls group) @ CERN
About

- Indian, studying in Spain, working in Switzerland (CERN), living in France
- Hardware & Timing section (Beam Controls group) @ CERN
- Develop timing & data acquisition hardware for big toys
  - Open Hardware Repository (http://ohwr.org)
Indian, studying in Spain, working in Switzerland (CERN), living in France

Hardware & Timing section (Beam Controls group) @ CERN

Develop timing & data acquisition hardware for big toys
  - Open Hardware Repository (http://ohwr.org)

VME and PCI hardware (FPGA based)
About

- Indian, studying in Spain, working in Switzerland (CERN), living in France
- Hardware & Timing section (Beam Controls group) @ CERN
- Develop timing & data acquisition hardware for big toys
  - Open Hardware Repository (http://ohwr.org)
- VME and PCI hardware (FPGA based)
- I work on Linux device drivers
<table>
<thead>
<tr>
<th>FPGAs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Made up of logic blocks</td>
</tr>
<tr>
<td>Logic described in a Hardware Description Language (HDL)</td>
</tr>
<tr>
<td>Synthesized</td>
</tr>
<tr>
<td>Loaded into FPGA</td>
</tr>
<tr>
<td>Dynamic hardware logic!</td>
</tr>
</tbody>
</table>
FPGAs

- Field-Programmable Gate Arrays
FPGAs

- Field-Programmable Gate Arrays
- Made up of 'logic blocks'
FPGAs

- Field-Programmable Gate Arrays
- Made up of 'logic blocks'
- Logic described in a Hardware Description Language (HDL)
FPGAs

- Field-Programmable Gate Arrays
- Made up of 'logic blocks'
- Logic described in a Hardware Description Language (HDL)
- Synthesized
FPGAs

- Field-Programmable Gate Arrays
- Made up of 'logic blocks'
- Logic described in a Hardware Description Language (HDL)
- Synthesized
- Loaded into FPGA
FPGAs

- Field-Programmable Gate Arrays
- Made up of ‘logic blocks’
- Logic described in a Hardware Description Language (HDL)
- Synthesized
- Loaded into FPGA
- Dynamic hardware logic!
FPGA Hardware at CERN

(a) SPEC Board  (b) White Rabbit Switch

Figure: Open Hardware from OHWR
Wishbone Bus

- Community-developed open bus protocol
Wishbone Bus

- Community-developed open bus protocol
- Great for connecting logic blocks
Wishbone Bus

- Community-developed open bus protocol
- Great for connecting logic blocks
  - OpenCores (http://opencores.org)
Wishbone Bus

- Community-developed open bus protocol
- Great for connecting logic blocks
  - OpenCores (http://opencores.org)
- Integrator places logic blocks in Wishbone address space
Wishbone Bus

- Community-developed open bus protocol
- Great for connecting logic blocks
  - OpenCores (http://opencores.org)
- Integrator places logic blocks in Wishbone address space
- Mapped and accessed as usual
Device Driver Model

- Monolithic driver?
Device Driver Model

- Monolithic driver?
  - Modular Hardware + FPGA = Extremely vast problem space
Device Driver Model

- Monolithic driver?
  - Modular Hardware + FPGA = Extremely vast problem space
- Blocks reused in different designs
Device Driver Model

- Monolithic driver?
  - Modular Hardware + FPGA = Extremely vast problem space
- Blocks reused in different designs
  - Should be exploited
Device Driver Model

- Monolithic driver?
  - Modular Hardware + FPGA = Extremely vast problem space
- Blocks reused in different designs
  - Should be exploited
- Clean design: Bus auto-discovery
Monolithic driver?
- Modular Hardware + FPGA = Extremely vast problem space
- Blocks reused in different designs
  - Should be exploited
- Clean design: Bus auto-discovery
  - Not defined in the Wishbone specification
Device Driver Model

- Monolithic driver?
  - Modular Hardware + FPGA = Extremely vast problem space
- Blocks reused in different designs
  - Should be exploited
- Clean design: Bus auto-discovery
  - Not defined in the Wishbone specification
  - Let’s add one!
Requirements

- So you want to auto-discover a bus?
So you want to auto-discover a bus?

Device identification
So you want to auto-discover a bus?
- Device identification
- Firmware metadata if possible
Requirements

- So you want to auto-discover a bus?
- Device identification
- Firmware metadata if possible
- Support for device hierarchy
Requirements

- So you want to auto-discover a bus?
- Device identification
- Firmware metadata if possible
- Support for device hierarchy
  - eg. 'Router block' controls multiple 'Ethernet port blocks'
Requirements

- So you want to auto-discover a bus?
- Device identification
- Firmware metadata if possible
- Support for device hierarchy
  - eg. 'Router block' controls multiple 'Ethernet port blocks'
- Shouldn’t leave proprietary blocks out in the cold
Requirements

- So you want to auto-discover a bus?
- Device identification
- Firmware metadata if possible
- Support for device hierarchy
  - eg. 'Router block' controls multiple 'Ethernet port blocks'
- Shouldn’t leave proprietary blocks out in the cold
  - Cannot be modified
Requirements

- So you want to auto-discover a bus?
- Device identification
- Firmware metadata if possible
- Support for device hierarchy
  - eg. 'Router block' controls multiple 'Ethernet port blocks'
- Shouldn’t leave proprietary blocks out in the cold
  - Cannot be modified
  - Can contain entire device hierarchy
Requirements

- So you want to auto-discover a bus?
- Device identification
- Firmware metadata if possible
- Support for device hierarchy
  - eg. 'Router block' controls multiple 'Ethernet port blocks'
- Shouldn’t leave proprietary blocks out in the cold
  - Cannot be modified
  - Can contain entire device hierarchy
- Should try not to constrain designers/integrators
So you want to auto-discover a bus?

- Device identification

- Firmware metadata if possible

- Support for device hierarchy
  - eg. 'Router block' controls multiple 'Ethernet port blocks'

- Shouldn’t leave proprietary blocks out in the cold
  - Cannot be modified
  - Can contain entire device hierarchy

- Should try not to constrain designers/integrators

- Should not force the use of external sources for metadata
Enabling Auto-Discovery: Part 1

- Device identification
Enabling Auto-Discovery: Part 1

- Device identification
  - Each logic block gets vendor/device ID
Device identification
- Each logic block gets vendor/device ID
- 64-bit Vendor IDs
Device identification

- Each logic block gets vendor/device ID
- 64-bit Vendor IDs
- Vendor/Device name strings, Device flags etc.
Enabling Auto-Discovery: Part 1

- Device identification
  - Each logic block gets vendor/device ID
  - 64-bit Vendor IDs
  - Vendor/Device name strings, Device flags etc.

- Firmware Metadata
Enabling Auto-Discovery: Part 1

- Device identification
  - Each logic block gets vendor/device ID
  - 64-bit Vendor IDs
  - Vendor/Device name strings, Device flags etc.

- Firmware Metadata
  - ID block containing firmware version information
Enabling Auto-Discovery: Part 1

- **Device identification**
  - Each logic block gets vendor/device ID
  - 64-bit Vendor IDs
  - Vendor/Device name strings, Device flags etc.

- **Firmware Metadata**
  - ID block containing firmware version information
  - Header block holds pointer to ID block and to list of devices
Enabling Auto-Discovery: Part 1

- **Device identification**
  - Each logic block gets vendor/device ID
  - 64-bit Vendor IDs
  - Vendor/Device name strings, Device flags etc.

- **Firmware Metadata**
  - ID block containing firmware version information
  - Header block holds pointer to ID block and to list of devices

- **Only header block location needed**
Enabling Auto-Discovery: Part 2

- Supporting hierarchy description
Enabling Auto-Discovery: Part 2

- Supporting hierarchy description
  - Parents have a variable list of child locations
Supporting hierarchy description
- Parents have a variable list of child locations
- Modification of parent doesn’t require modifying children
Enabling Auto-Discovery: Part 2

- Supporting hierarchy description
  - Parents have a variable list of child locations
  - Modification of parent doesn’t require modifying children
- Supporting proprietary blocks
Enabling Auto-Discovery: Part 2

- Supporting hierarchy description
  - Parents have a variable list of child locations
  - Modification of parent doesn’t require modifying children
- Supporting proprietary blocks
  - Relative offsets
Enabling Auto-Discovery: Part 2

- Supporting hierarchy description
  - Parents have a variable list of child locations
  - Modification of parent doesn’t require modifying children

- Supporting proprietary blocks
  - Relative offsets

- Array of top level devices. Location in header
Auto-discovery!

WB Memory Map
Auto-discovery!

**WB Memory Map**

HEADER
Auto-discovery!

WB Memory Map

ID BLOCK
HEADER
Auto-discovery!

WB Memory Map

Top Level Blocks
- NULL
- Device 2
- Device 1
- ID BLOCK
- HEADER
Auto-discovery!

WB Memory Map

Device 2
(Descriptor)

Device 1
(Descriptor)

NULL

Device 2

Device 1

ID BLOCK

HEADER
Auto-discovery!

WB Memory Map

- Top Level Blocks
  - ID BLOCK
  - HEADER

Device 1 (Descriptor)
- NULL
- Device 2
- Device 1

Device 2 (Descriptor)
- NULL
- Child 1
- Child 2

NULL
Auto-discovery!

WB Memory Map

- Child 2 (Descriptor)
- Child 1 (Descriptor)
- NULL
- Child 2
- Child 1
- Device 2 (Descriptor)
- Device 1 (Descriptor)
- NULL
- Device 2
- Device 1
- ID BLOCK
- HEADER
Auto-discovery!

WB Memory Map

Proprietary Block

Device 1 (Descriptor)

NULL

Device 2

Device 1

ID BLOCK

HEADER
Auto-discovery!

WB Memory Map

- Child 2 (Descriptor)
- Child 1 (Descriptor)
- NULL
  - Child 2
  - Child 1
- Device 2 (Descriptor)
- Device 1 (Descriptor)
- NULL
  - Device 2
  - Device 1
- ID BLOCK
- HEADER

FPGA Hardware Driver

Header Location
Auto-discovery!

WB Memory Map

- Child 2 (Descriptor)
- Child 1 (Descriptor)
- NULL
- Device 2 (Descriptor)
- Device 1 (Descriptor)
- NULL
- Device 2
- Device 1
- ID BLOCK
- HEADER

Wishbone Drivers

- Device 1
- Device 2

Wishbone Bus Driver

FPGA Hardware Driver

Top Level Blocks

Header Location
Auto-discovery!

**WB Memory Map**

- **Child 2** (Descriptor)
- **Child 1** (Descriptor)
- **NULL**
- **Device 2** (Descriptor)
- **Device 1** (Descriptor)
- **NULL**
- **Device 2**
- **Device 1**
- **ID BLOCK**
- **HEADER**

**Wishbone Drivers**

- **Device 1**
- **Device 2**

**Wishbone Bus Driver**

- `wb_register_bus()`

**FPGA Hardware Driver**

**Introduction**

The Hardware

Device Drivers

Self-Describing Wishbone Bus (SDWB)

Future Work
Auto-discovery!

**WB Memory Map**

```
HEADER
ID BLOCK
Device 1 (Descriptor)
Device 2 (Descriptor)
NULL
Child 1
Child 2

```

**Wishbone Drivers**

```
Device 1
Device 2

```

**FPGA Hardware Driver**

```
wb_register_bus()

```

**SDWB Auto Discovery**

```
Wishbone Bus Driver

```
Auto-discovery!

**WB Memory Map**

- **Top Level Blocks**
  - ID BLOCK
  - HEADER

- **Child 2 (Descriptor)**
  - Child 1 (Descriptor)

- **Device 2 (Descriptor)**
  - Child 2
  - Child 1

- **Device 1 (Descriptor)**

**Wishbone Drivers**

- **Device 1**
  - `probe()`
  - `wb_register_bus()`

- **Device 2**
  - `probe()`

**FPGA Hardware Driver**

**SDWB Auto Discovery**

**Wishbone Bus Driver**
Auto-discovery!

WB Memory Map

<table>
<thead>
<tr>
<th>Top Level Blocks</th>
<th>ID BLOCK</th>
<th>HEADER</th>
</tr>
</thead>
<tbody>
<tr>
<td>Child 2 (Descriptor)</td>
<td>NULL</td>
<td>Device 2</td>
</tr>
<tr>
<td>Child 1 (Descriptor)</td>
<td>Child 2</td>
<td>Device 1</td>
</tr>
<tr>
<td>NULL</td>
<td>Device 2</td>
<td>Device 1</td>
</tr>
</tbody>
</table>

Wishbone Drivers

Device 1

Device 2

probe()

SDWB Auto Discovery

Wishbone Bus Driver

FPGA Hardware Driver

Manohar Vanga
Examples

- Test drivers
Examples

- Test drivers
  - Wishbone 1-wire and I2C block driver
Examples

- Test drivers
  - Wishbone 1-wire and I2C block driver
  - ADC controller driver (SPEC board)
Development Efforts

- Test specification with more complex hardware
Development Efforts

- Test specification with more complex hardware
- Integrate with Wishbone standard
Development Efforts

- Test specification with more complex hardware
- Integrate with Wishbone standard
- Go upstream
Development Efforts

- Test specification with more complex hardware
- Integrate with Wishbone standard
- Go upstream
  - Provide support for older kernels
Development Efforts

- Test specification with more complex hardware
- Integrate with Wishbone standard
- Go upstream
  - Provide support for older kernels
  - Drivers for specific blocks
Thanks!

- OHWR Page:
  http://www.ohwr.org/projects/fpga-config-space