The Self-Describing Wishbone Bus (SDWB)

Manohar Vanga

BE-CO-HT, CERN, Geneva

Jule 13, 2007

About

About

- Indian

About

- Indian, studying in Spain

About

- Indian, studying in Spain, working in Switzerland (CERN)

About

- 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)

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)

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

FPGAs

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

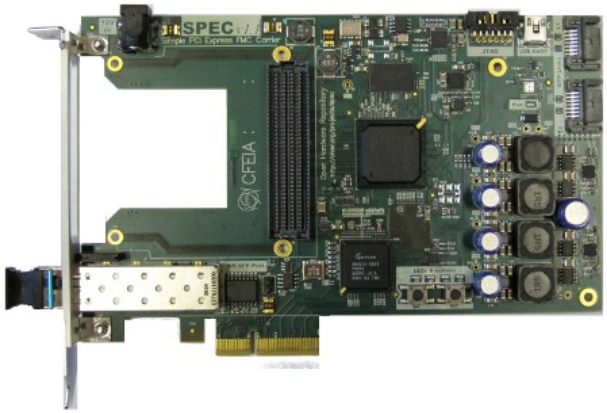
FPGAs

- 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

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

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?

Requirements

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

Requirements

- 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

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
- 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

Enabling Auto-Discovery: Part 1

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

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.

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

Enabling Auto-Discovery: Part 2

- 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

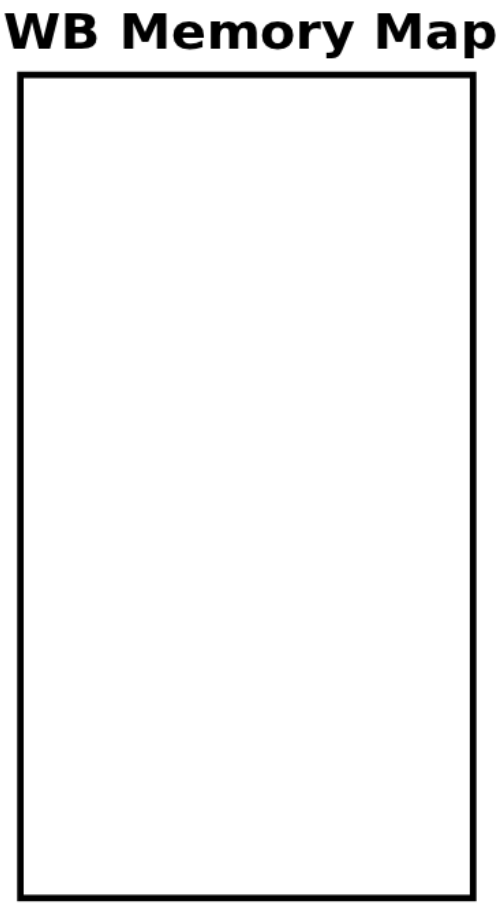
Enabling Auto-Discovery: Part 2

- Supporting hierarchy description
  - Parents have a variable list of child locations
  - Modification of parent doesn't require modifying children
- Supporting proprietary blocks
  - Relative offsets

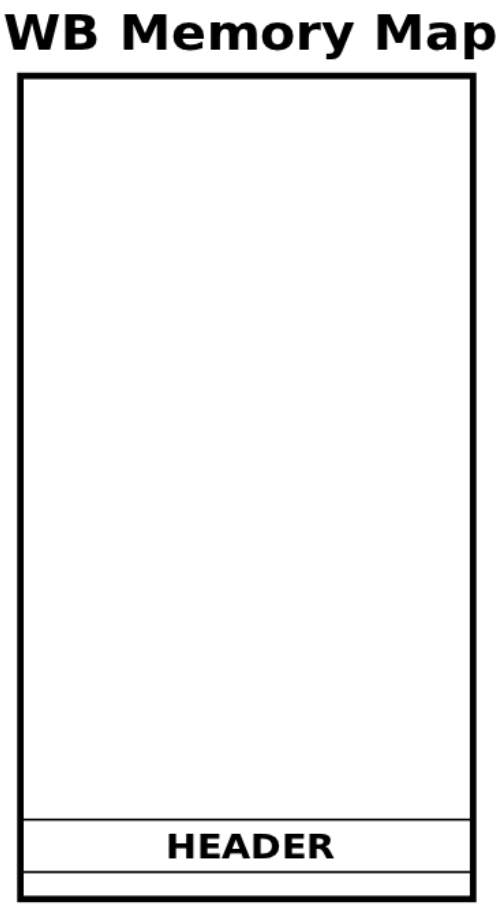
Enabling Auto-Discovery: Part 2

- 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

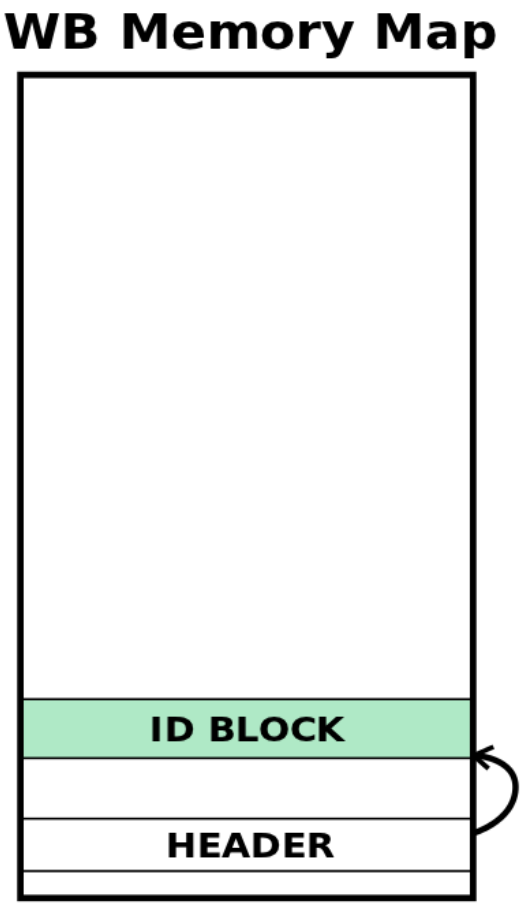
Auto-discovery!



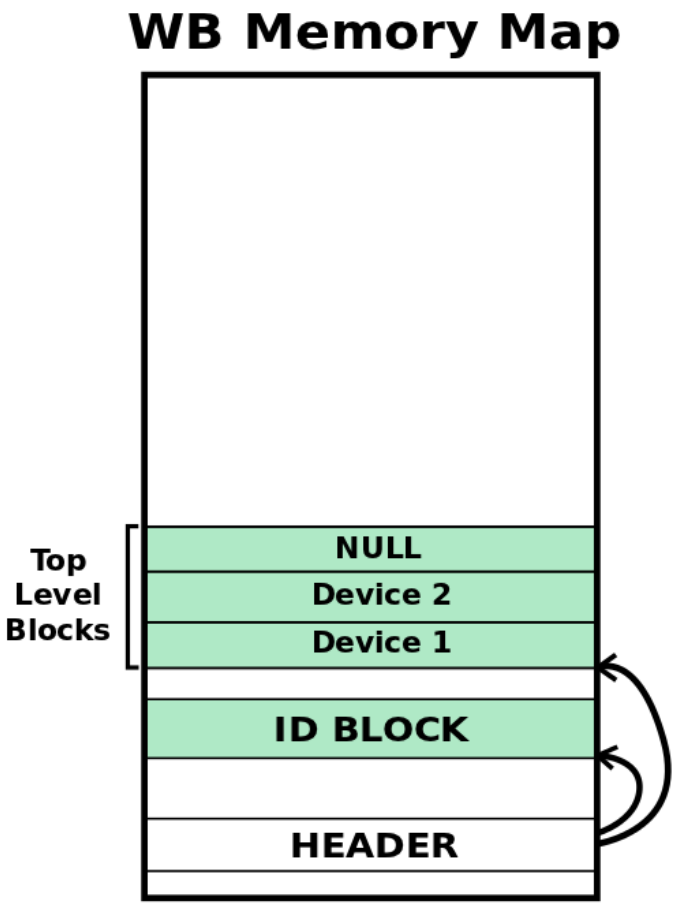
Auto-discovery!



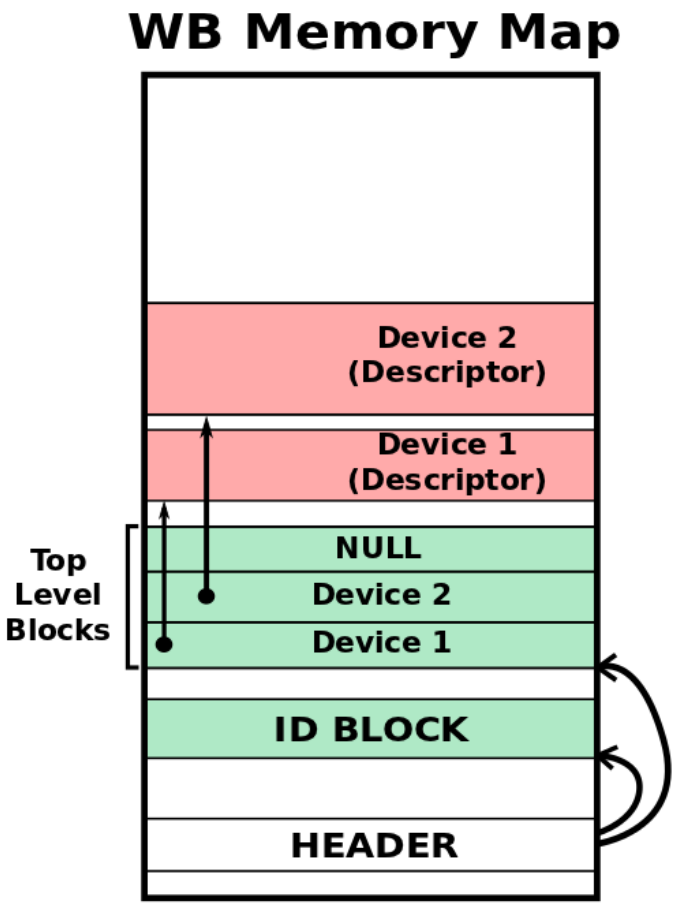
Auto-discovery!



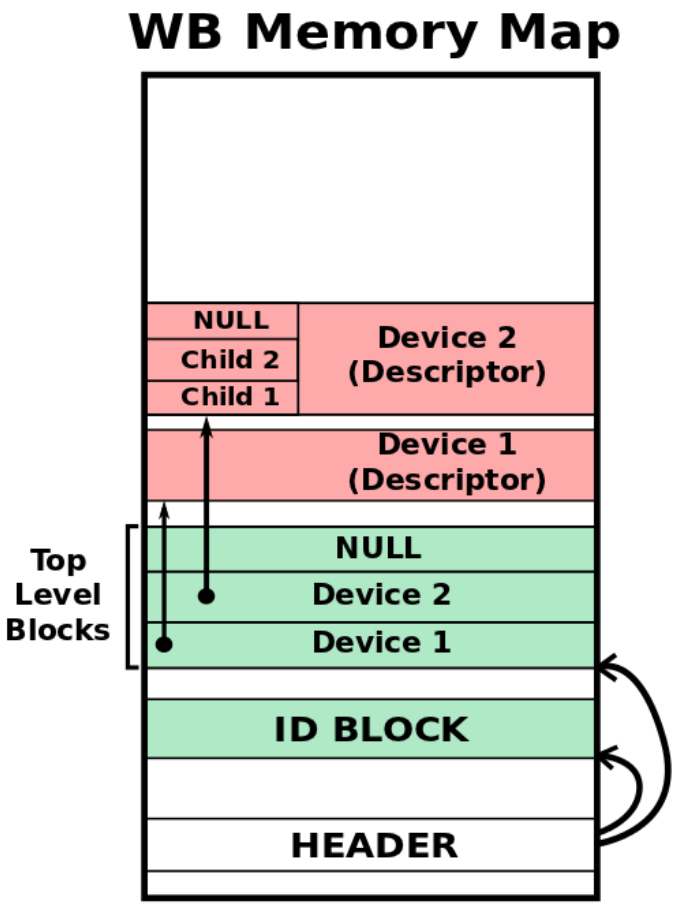
Auto-discovery!



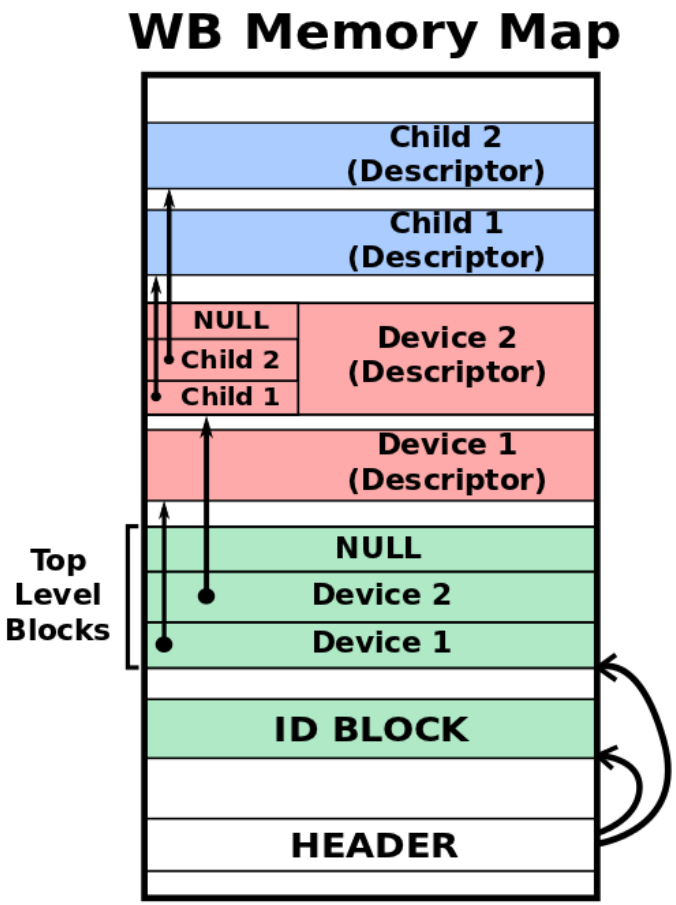
Auto-discovery!



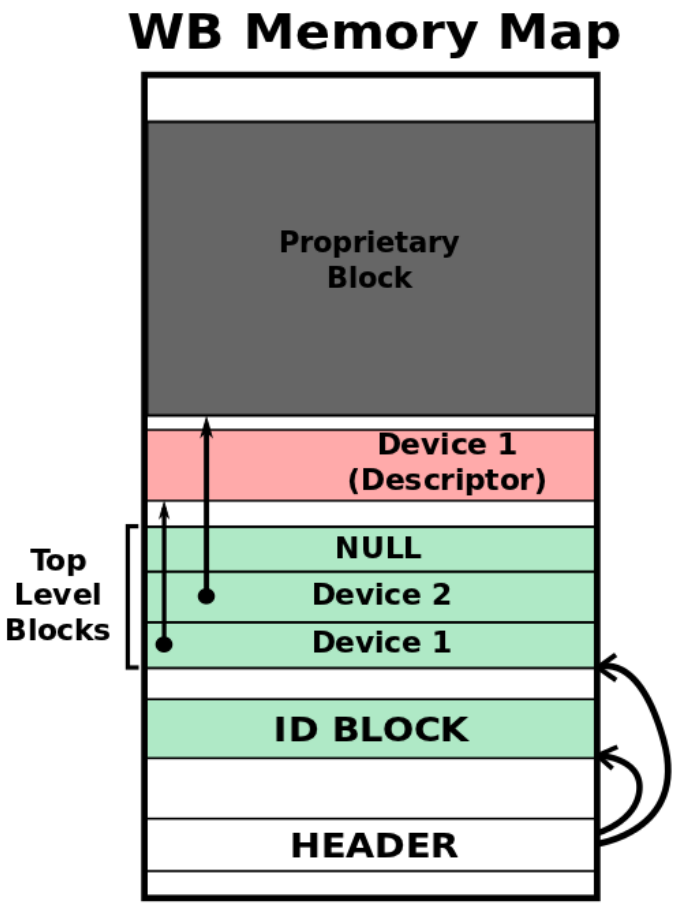
Auto-discovery!



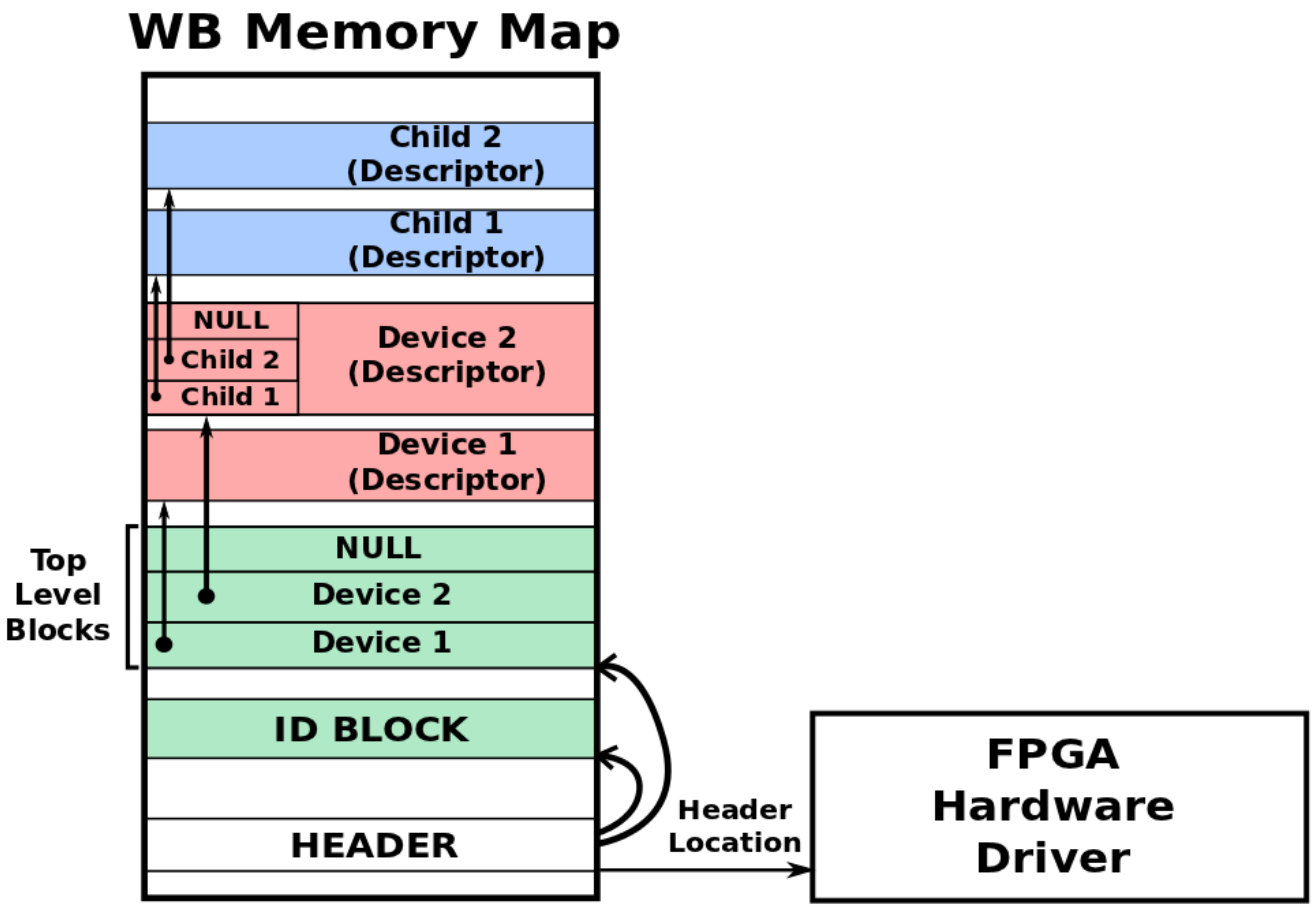
Auto-discovery!



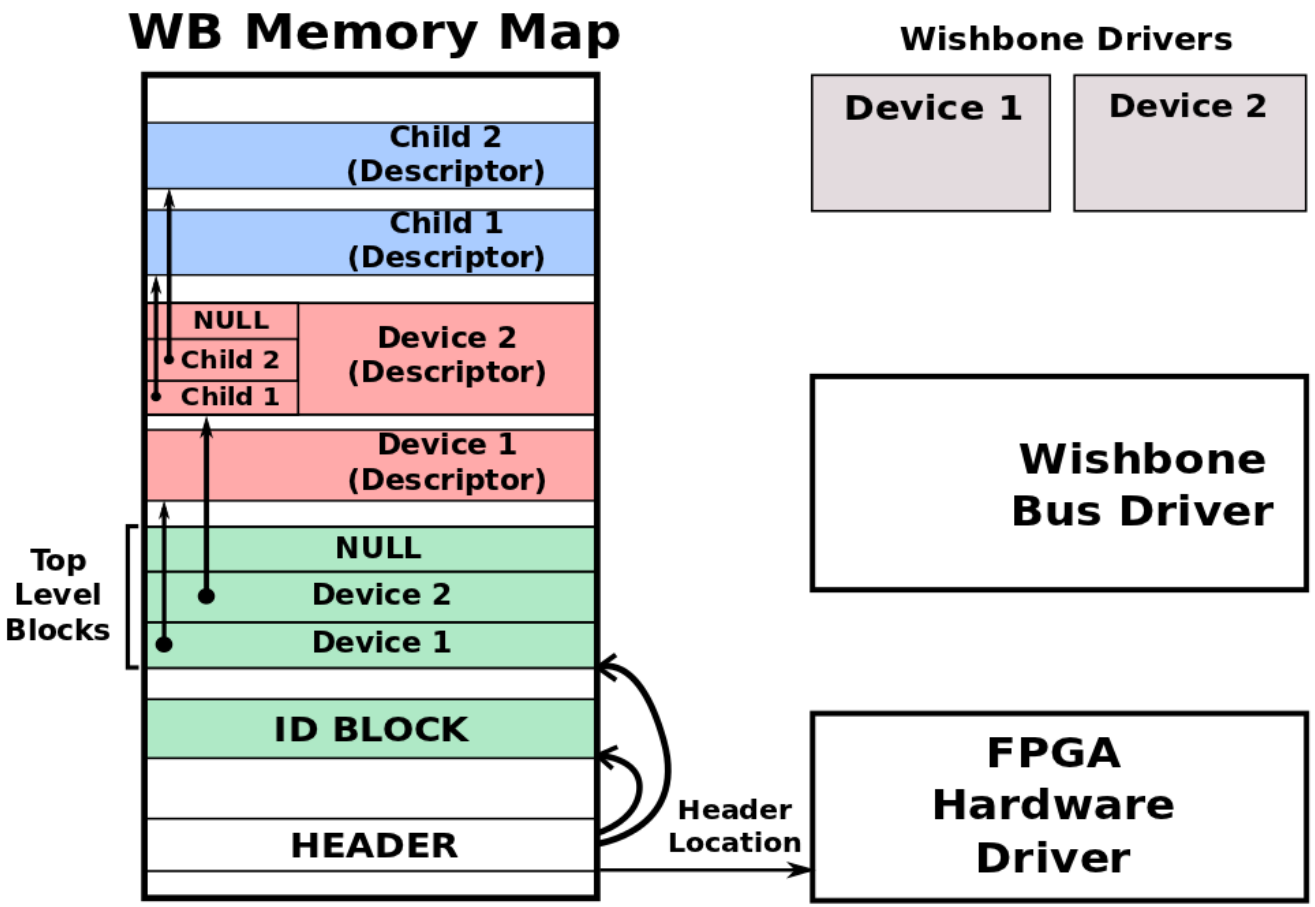
Auto-discovery!



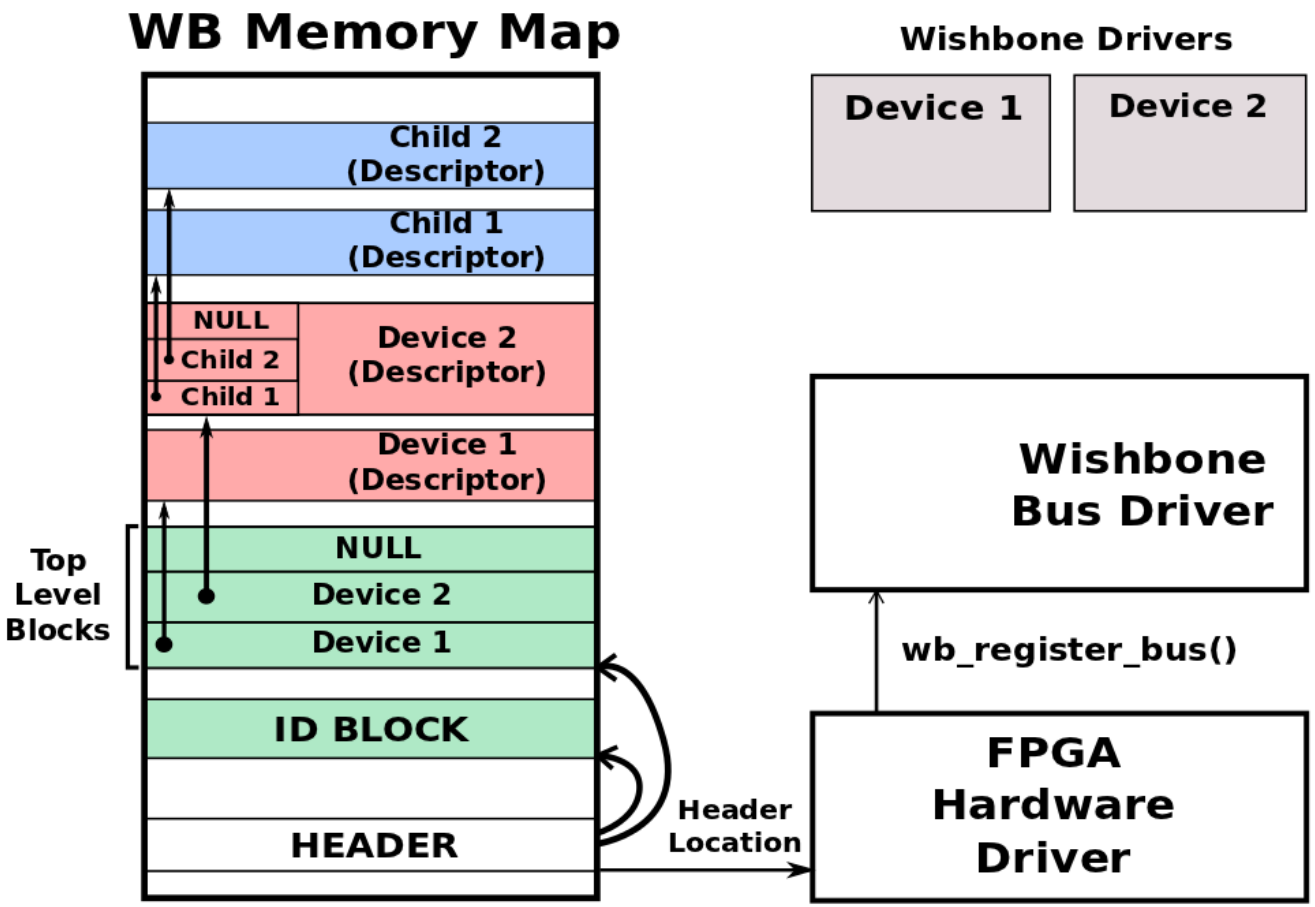
Auto-discovery!



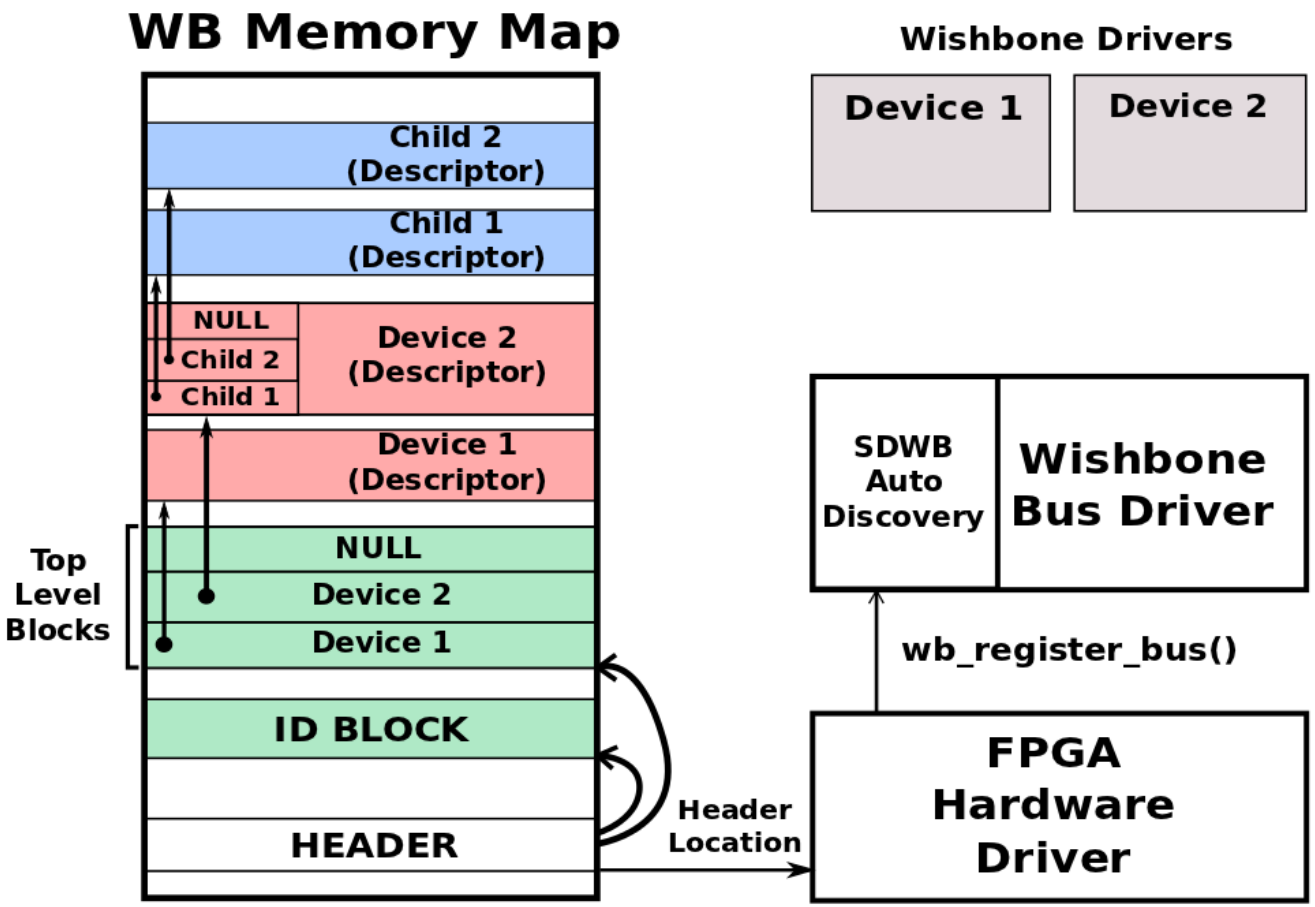
Auto-discovery!



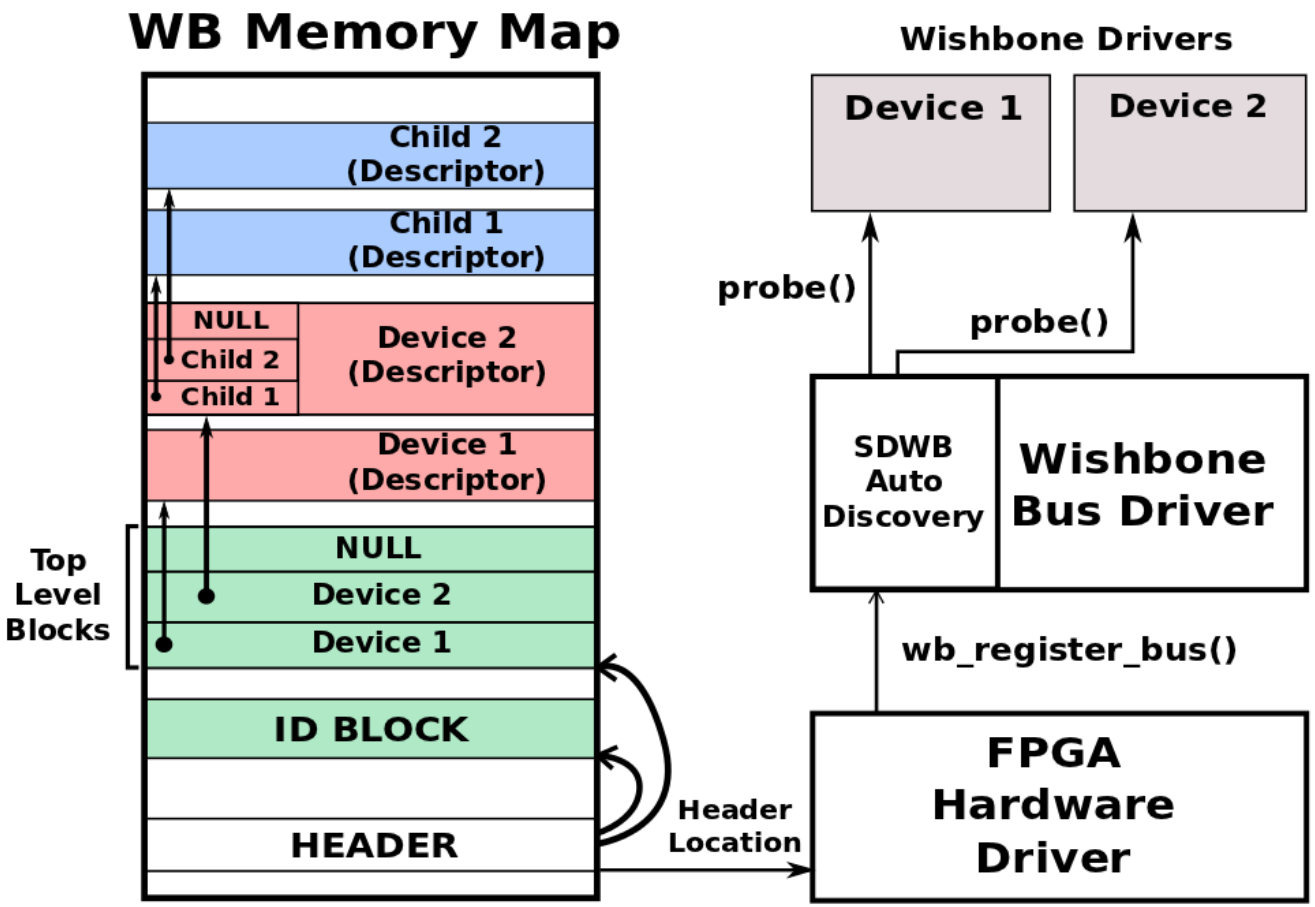
Auto-discovery!



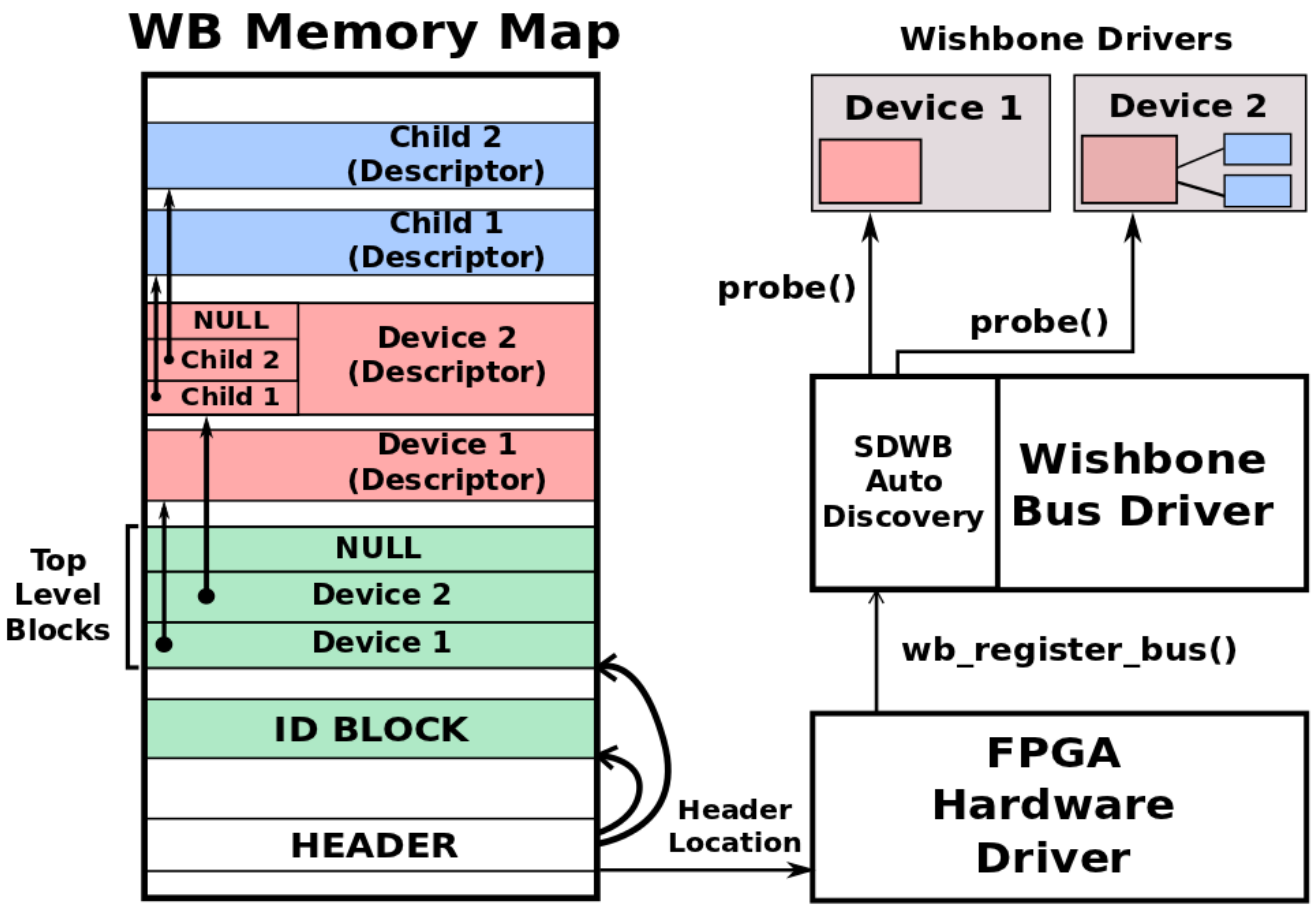
Auto-discovery!



Auto-discovery!



Auto-discovery!



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