Towards a Standard Hardware API and a Standard Device Model

Martin Killenberg
on behalf of the
PICMG software working group

MicroTCA Workshop 2014
DESY, Hamburg

Introduction

PICMG Design Guides

- Give implementation recommendations
- Facilitate re-usability and portability

Standard Hardware API

- Better interoperation of modules from different vendors

Standard Device Model

- Easier software integration of hardware, independent from vendor and protocol

Standard Hardware API (SHAPI)

- Devices should be able to identify themselves
- Standard Register Set
  - A set of defined registers which can be found in all devices
  - Firmware name, version, vendor, ID
  - Hardware name, version, vendor, ID
  - Device capabilities and resources
    - DMA
    - Interrupts
- Independent from the hardware protocol
- Mechanism to address sub-devices

SHAPI Extensibility

- Devices can have sub-devices
  - Sub-device firmware name, version, ID
  - Sub address range for the specific functionality

- Sub-device examples
  - Mezanine cards
  - MicroTCA.4 rear transition modules
  - Reusable algorithm blocks

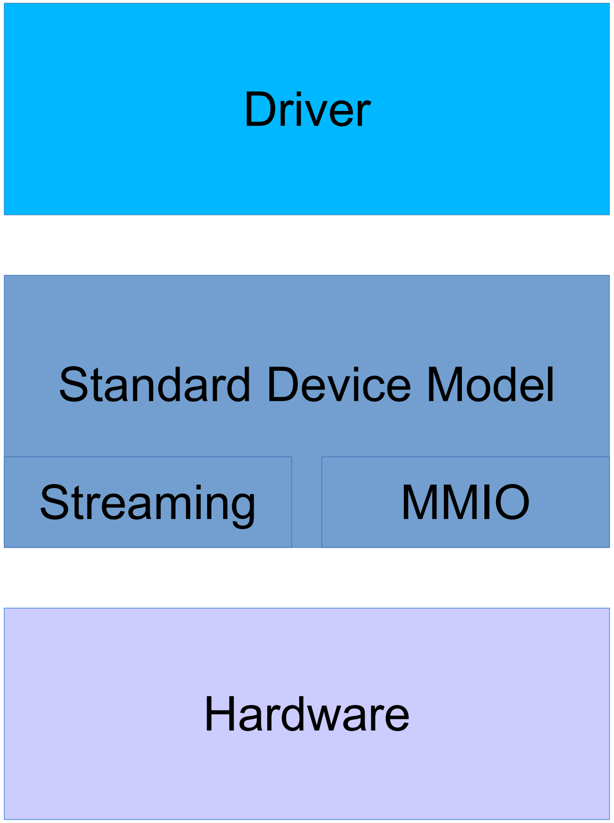
SHAPI Advantages

- Better interoperability of different vendors
- Devices can use a common driver
- Drivers can detect resources present on a particular device

Standard Device Model

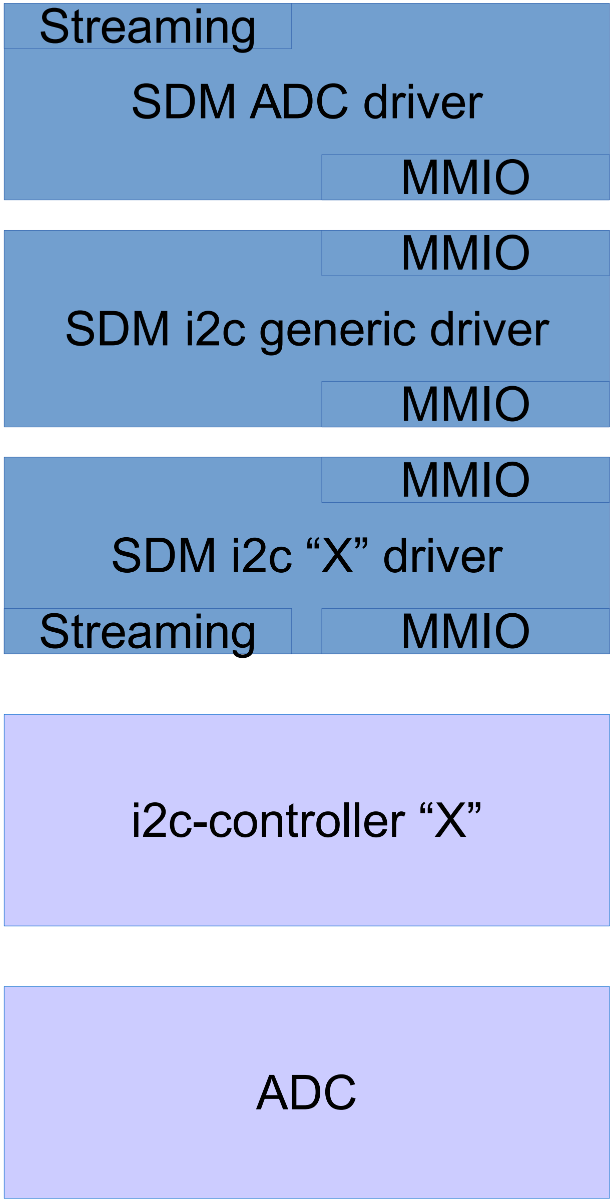
Abstract interface to communicate with a device

- Complete abstraction of the hardware
- Device name mapping
  - Instantiation and connection independent from the protocol
- Device I/O
  - Streaming device
    - Write a sequence of data
  - Address (or “memory-”) mapped device (MMIO)
    - Access individual registers



Stackability

- SDM Drivers present SDM API up- and downstream
- Can build hierarchies of “stacks” and “groups” of devices.



Device Name Mapping

Standard device names

- URIs:
  //host/interface:[instance][;protocol][=parameters]
- Alias Names
- SDM knows how to open different protocols on different platforms, all accessed via the same interface
- High level software use aliases (functional names) which are independent from protocol and instance

Streaming and Address Based I/O

Two main I/O methods

- Streaming I/O
  - A stream of data is written to the same channel
  - Seeking in the stream can be possible (device dependent)
- Address based I/O
  - Address space with registers
  - Registers are accessed by their offset
  - Random access

Both versions are flavours of the Standard I/0 Device

Sub-Devices

A device can consist of several sub-devices

- Different Base Address Ranges in PCIe
- Connect different hardware devices to one logical entity
  - ADC and piezo actuator of a feedback loop
- Streaming and address based access in the same device
  - Address based for configuration of an ADC
  - Stream to read the actual ADC output

Examples

- Access PCIe addresses in user space
- Virtual devices for testing and software development
- Tunnel an address space through a different protocol
  - Address based access over Ethernet or RS232
  - Access an SPI or I^2C bus on an MicroTCA AMC via PCIexpress

Reference Implementations

Provide reference implementations for different languages and platforms

- C (procedural)
- C++ (object oriented)
- Java?
- Python?

- Windows
- Linux

- Implementation examples
- Ready-to-use code for a fast start

Status

- Draft versions getting last iterations in the PICMG SW working group
- Reference implementations being written for C (Windows) and C++ (Linux)
- Help welcome for Java and Python