• Modern SoCs are very heterogenous
  - MPSoC: A53s, R5s, PMU, MicroBlazes

• System Software needs a lot of HW info
  - Memory allocated for each domain
    - Including shared pages
  - Devices assigned to each domain
  - Addresses of memory and registers
    - Same device can have different addresses
  - Topologies (clocks, busses, …)

• Config info comes from different personas
  - HW configuration – HW architect
    - The topology of the HW resources, including SoC resources, board resources, PL resources
  - Domain allocation – System Architect
    - Which HW resources are allocated to which domain
  - Domain configuration – BSP creator
    - OS specific information

• Config happens at different times
  - Compiled in during build time
    - Typical for RTOS, BM
  - Dynamically during boot time
    - Typical for Linux, Xen
  - Dynamically during runtime
    - Reassigning resources while running
Domains and Operating Environments

- Multiple core clusters
  - A53s, R5s, PMU, MicroBlazes

- Multiple Execution Levels (EL) on some cores
  - EL0 – User space
  - EL1 – OS
  - EL2 – Hypervisor
  - EL3 – Firmware

- Multiple Security Environments
  - TrustZone (TZ) – HW protecting resources (e.g. memory)
  - Trusted Execution Environment (TEE) – SEL1

- Domains
  - A Domain is a separate address space, including devices
  - Defined by different cores, Execution Levels and security environments

- Multiple Operating Environments (OE)
  - An OE is the system SW that runs in a Domain, including:
    - Linux (including Android), Free and commercial RTOS’s
    - Bare metal (no OS), Hypervisors
    - Firmware/boot loaders – Trusted FW, PLM, PMU FW, uboot, ...

Secure State
- Trusted Firmware (TF)

Non-secure State
- Linux
- RTOS

EL0
- Container
- App
- App
- App

EL1
- Linux
- RTOS

EL2
- Hypervisor

EL3
- Trusted Execution Environment (TEE)

A53 Core 0
A53 Core 1
A53 Core 2
A53 Core 3

PMU Firmware
Platform Management Unit (PMU)
Problem Statement

● Allocation and configuration of HW resources are complex
  ○ Typically done today in an ad-hoc way depending SoC and OS/Firmware
  ○ Especially tricky to define shared resources, such as OpenAMP/virtIO buffers

● Industry standards and common tools are needed to simplify configuration
  ○ Define resource allocation to Domains in one place, including shared buffers
  ○ Allows common flows for any SoC and Operating system/Firmware
  ○ Simplifies integration of OSes, Hypervisors and Firmware from different places and vendors

● Verification is critical at integration time and/or runtime
  ○ Make sure that different Domains are not erroneously using the same resources
Device Trees and System Device Trees

- **Device Trees (DTs) express HW information relevant to** Operating Environments
  - Been used by PPC and ARM SSW to define HW that cannot be dynamically discovered
  - Used by uboot, Linux, Xen and increasingly being used by RTOS vendors
- **Device Trees describes HW nodes and topologies**
  - *Traditional Device Trees are only describing the world seen from one Address Space*
- **Additional system level Device Tree information is proposed**
  - A **System Device Tree (S-DT)** describes all HW that later can be divided into different partitions
- **System DT additions include two parts:**
  1. DeviceTree.org specification and tooling additions
     - Describing multiple cpu clusters and corresponding views of their address spaces
     - Enabling source-to-source translations by adding options to keep labels and comments
  2. **AMP configuration information**
     - Resource allocation using domains and resource groups
     - Using the Device Tree **chosen** section to specify AMP configuration
     - Specification of shared resources, such as pages for virtIO buffers
     - Intent is to align with hypervisor information (e.g. Xen Dom0-less configuration)
System Device Tree Data Flow (RTOS & Linux)

**HW Personas**
- SoC info
  - Fixed per SoC
- Board info
  - I/O pin selection
  - Board devices
- Devices in Programmable Logic

**System Architect**
- Domain Info
  - soc.dts
  - board.dts
  - pl.dts

**Domain Integrator RTOS/BM**
- Domain Integrator Linux
  - lopper
    - dom_r5
    - dom_a53
  - linux_conf.dts

**User BSP Config**
- User
  - libfdt, C code
  - *.ch files
  - RTOS/BM Driver Device tree

**System Integrator**
- ELF
  - rtos.dts
  - dtsverify
  - dts2rtos
  - BSP selection

**Standardized Data files**
- Open source tool
- New open source code
- RTOS specific data

1) .dts files before lopper is a System Device Tree
2) The .dts can be used to select drivers
3) RTOS specific transformations
System Device Tree Data Flow (Xen & QEMU)

**HW Personas**
- SoC info: Fixed per SoC
  - Board info: I/O pin selection, Board devices
- Devices in Programmable Logic

**System Architect**
- Domain Info
  - soc.dts
  - board.dts
  - pl.dts

**Domain Integrator XEN**
- xen_conf.dts
- xen.dtb

**Domain Integrator QEMU**
- qemu_conf.dts
- qemu.dtb

**Standardized Data files**
- Open source tool
- New open source code
- RTOS specific data

> 1) .dts files before lopper is a System Device Tree
2) VM config from XEN or created separately
3) QEMU defines HW from a System Device Tree
System Device Tree (1/2)

```dts
/dts-v1/
{
  compatible = "slnx,slnx-hdmi", "slnx,slnx-display";
  ...,
  cpu0 {
    reg = <0x0>
  }
}
```

System Device Tree (2/2)

```dts
memory {
  device_type = "memory";
  reg = <0x0 0x80000000 0x0 0x80000000>
}
```

Domains

```dts
reserved-memory {
  @address-cells = <0x2>
}
```

Linux Device Tree (A53)

```dts
/dts-v1/
{
  compatible = "slnx,slnx-hdmi", "slnx,slnx-display";
  ...,
  cpu0 {
    reg = <0x0>
  }
}
```
Domains and Resource Groups

- **A domain** is a set of nodes defining an address space
  - Used to create a traditional DT to an OS like Linux
  - Domains can both refer to physical objects (CPUs, memory, devices) as well as other resource groups and domains
  - Specified in the chosen section
  - Using the compatible = "openamp, domain-v1" attribute

- **A resource group** is a set of shared nodes grouped together
  - Devices that can be used by multiple domains
  - Memory areas that can be shared by multiple domains
  - Specified in the chosen section
  - Using the compatible = "openamp, group-v1" attribute

- **Examples of domains:**
  - A list of devices that could be shared between different domains
  - Shared pages and other resources used by VirtIO/rproc/rpmsg
  - A Linux domain running on the A53s including some memory and devices
  - A trusted FW/TZ accessing A53 domain plus more
Open Source Tool (Lopper) to Manipulate S-DT

- Built on top of existing Device Tree tooling and libraries (DTC, libfdt)

- Manipulates S-DTs and DTS by
  - Merging/adding information like new nodes and new attributes (already exists in DTC)
    - Merge S-DTs from different places (SoC, board, FPGA, ...)
    - Includes suppressing nodes that are not pinned out to the board
  - Prunes System Device Trees to traditional/partitional Device Trees
    - Removes nodes not used, resolves address mapping issues
    - Other transformation (clocks, phandles, ...)
    - Use a domain configuration in a S-DT chosen section to select what to be kept/pruned

- Additional features
  - Extensible to add SoC vendor specific information, such as remoteproc info
  - Keeps as much as possible of original information, including labels and comments
  - Can create source output as well as binary output (dts and dtb)
Interested?

- System Device Tree meeting at Connect 9:00 AM on Thursday
- Send me an email: tomase@Xilinx.com
Thank you

Join Linaro to accelerate deployment of your Arm-based solutions through collaboration

contactus@linaro.org