TOSHIBA

THE ROLE OF THE CCM IN AN END TO END RECONFIGURABLE SYSTEM

END-TO-END RECONFIGURABILITY

Craig Dolwin – Toshiba Research Europe Ltd
Jorg Brakensiek, Stefan Mende – Nokia Research Center

- Objectives
- Constraints
- Approach
- Hardware Abstraction
- Configuration Control Module
- Issues
- Summary

TOSHIBA

Objectives

- Reconfigurable physical layer and signal processing applications
- Dynamic reconfiguration
- A generic configuration interface that hides implementation detail
- Scalable
  - With improvements in equipment technology
  - With enhanced Radio Access Technologies (RAT)

Constraints

- Power consumption
  - Power consumed due to reconfigurability overhead should be minimised
  - Customer will not sacrifice functionality or battery life for SDR !
- Reliable and deterministic operation
  - Hard timing deadlines
- Support for proprietary hardware architectures
  - Manufacturer must be allowed to develop novel architectures. i.e. not a fixed architecture such as a PC

TOSHIBA

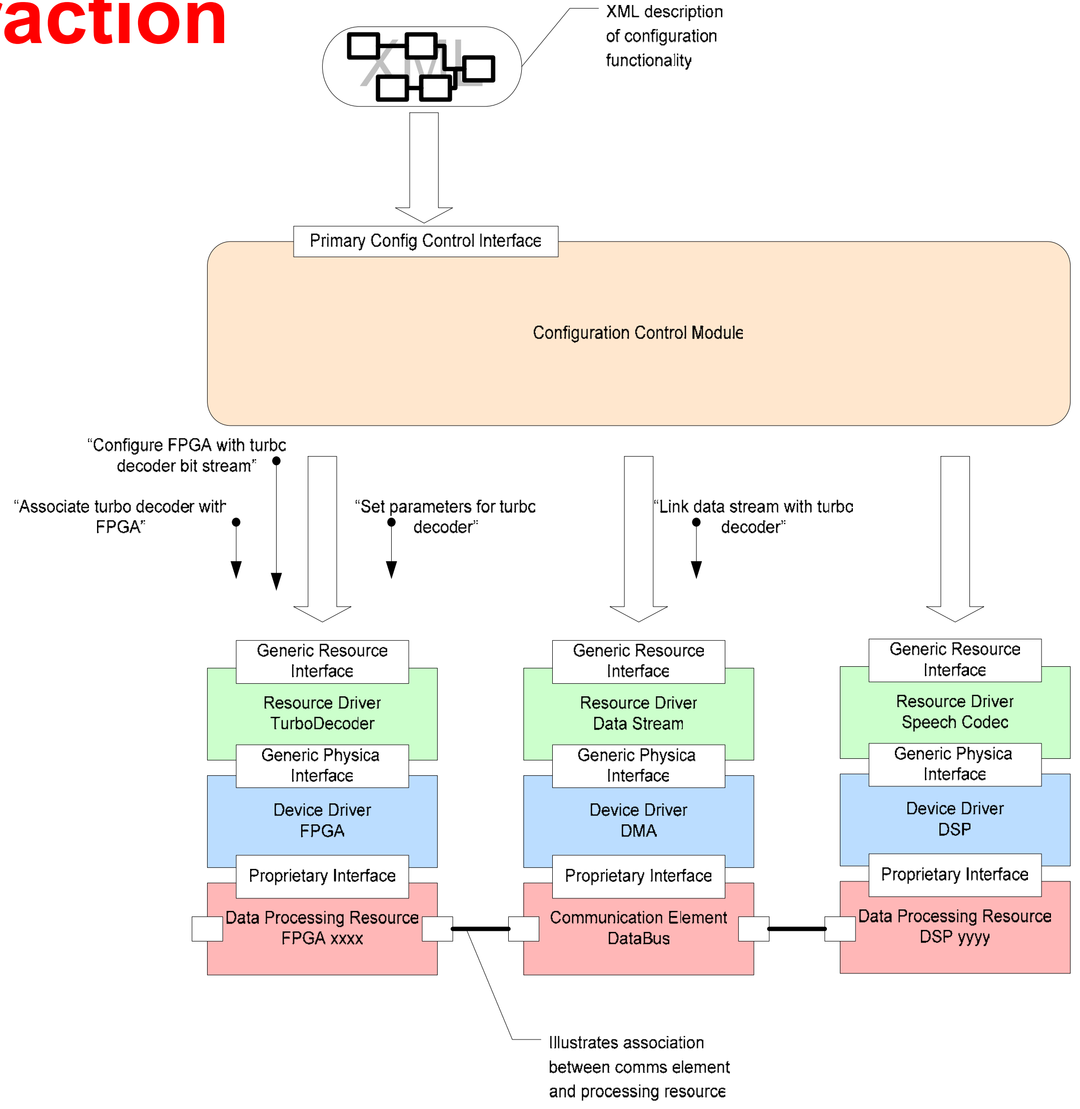
Approach

- Encapsulate data processing functions into modules
  - Configuration Execution Modules (CEM)
  - Allow data processing functions to be implemented using most power efficient methods available e.g. Assembler, Hardware Accelerators etc.
- A Configuration is then constructed by linking each module using the available communication fabric
- The Configuration Control Module (CCM) is used to coordinate all resources
- XML used to describe configurations

TOSHIBA

Hardware Abstraction

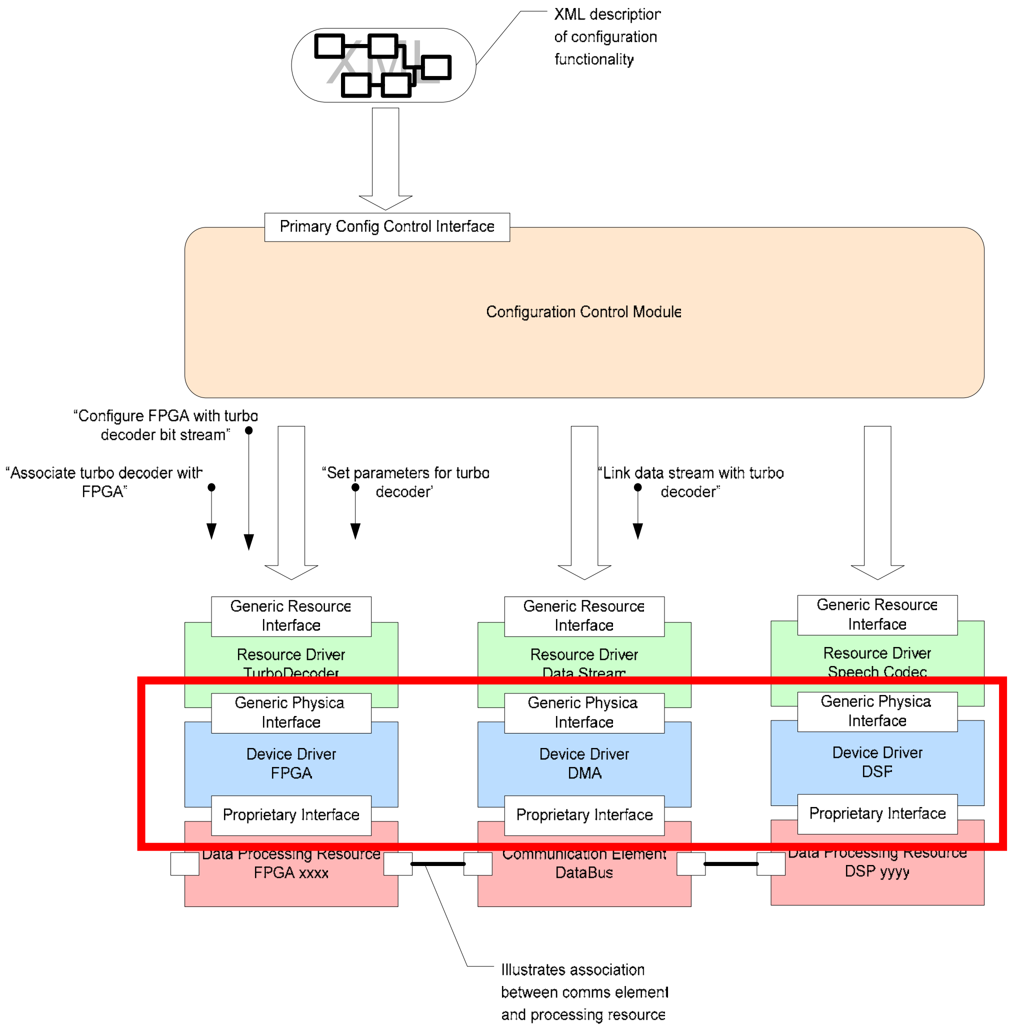
- Configuration Plane Only
- Two layers
  - Generic Physical Interface and Generic Resource (or Function) Interface


XML
XML description of configuration functionality
Primary Config Control Interface
Configuration Control Module
"Configure FPGA with turbo decoder bit stream"
"Associate turbo decoder with FPGA"
"Set parameters for turbo decoder"
"Link data stream with turbo decoder"
Generic Resource Interface
Resource Driver TurboDecoder
Generic Physical Interface
Device Driver FPGA
Proprietary Interface
Data Processing Resource FPGA xxxx
Generic Resource Interface
Resource Driver Data Stream
Generic Physical Interface
Device Driver DMA
Proprietary Interface
Communication Element DataBus
Generic Resource Interface
Resource Driver Speech Codec
Generic Physical Interface
Device Driver DSP
Proprietary Interface
Data Processing Resource DSP yyyy
Illustrates association between comms element and processing resource

TOSHIBA

Hardware Abstraction

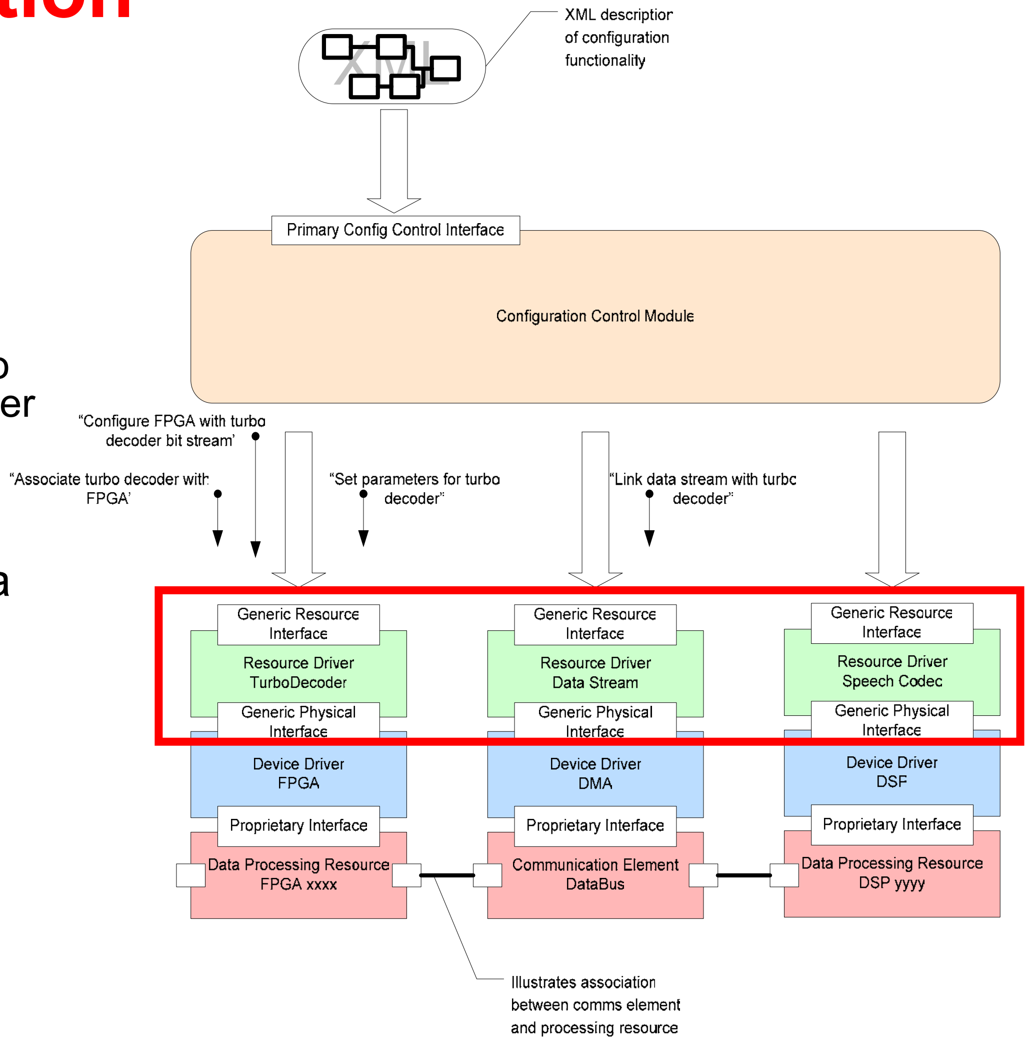
- Device Drivers
  - Hide physical method for configuring or starting resources
  - Implements Generic Physical Layer Interface
  - Located on control processor with CCM
  - Device drivers exist for data processing functions and communication elements
  - Device driver supplied by equipment manufacturer


XML description of configuration functionality
XML
Primary Config Control Interface
Configuration Control Module
"Configure FPGA with turbo decoder bit stream"
"Associate turbo decoder with FPGA"
"Set parameters for turbo decoder"
"Link data stream with turbo decoder"
Generic Resource Interface
Resource Driver TurboDecoder
Generic Physical Interface
Device Driver FPGA
Proprietary Interface
Data Processing Resource FPGA xxxx
Generic Resource Interface
Resource Driver Data Stream
Generic Physical Interface
Device Driver DMA
Proprietary Interface
Communication Element DataBus
Generic Resource Interface
Resource Driver Speech Codec
Generic Physical Interface
Device Driver DSP
Proprietary Interface
Data Processing Resource DSP yyyy
Illustrates association between comms element and processing resource

TOSHIBA

Hardware Abstraction

- Resource Driver
  - Allows reconfigurable resources to be programmed without detailed knowledge of implementation
    - e.g. setting the polynomial for a turbo decoder will use same method whether implemented on a DSP or hardware accelerator
  - Uses services supplied by the associated device driver to access a resource
    - e.g. Resource driver will supply the offset address for the control register used to set the polynomial value and the device driver will supply the absolute address for all control registers used by the turbo decoder


XML description of configuration functionality
XML
Primary Config Control Interface
Configuration Control Module
"Configure FPGA with turbo decoder bit stream"
"Associate turbo decoder with FPGA"
"Set parameters for turbo decoder"
"Link data stream with turbo decoder"
Generic Resource Interface
Resource Driver TurboDecoder
Generic Physical Interface
Device Driver FPGA
Proprietary Interface
Data Processing Resource FPGA xxxx
Generic Resource Interface
Resource Driver Data Stream
Generic Physical Interface
Device Driver DMA
Proprietary Interface
Communication Element DataBus
Generic Resource Interface
Resource Driver Speech Codec
Generic Physical Interface
Device Driver DSP
Proprietary Interface
Data Processing Resource DSP yyyy
Illustrates association between comms element and processing resource

Configuration Control Module (CCM)

- Responsible for coordinating resources to implement a requested configuration
- Takes as its input a platform independent description for a configuration.
  - Looking at using XML for this description
- Maintains a local database of object code, device drivers, resource drivers and metrics.
- CCM defines:
  - Which resource implements a function (Spatial Scheduling)
  - When a resource executes the function (Temporal Scheduling)

Configuration Control Module

- Service API
  - A common interface to CCM
  - Uses a simple Message Passing Interface (MPI)
  - Made up of several sub-interfaces:
  - Primary Configuration Control
    - Used by Configuration Management Module (CMM) to request a new RAT
  - Secondary Configuration Control
    - Allows resources to be reconfigured to implement different operating modes within a RAT or adapt to match channel environment e.g. Additional transport channels or a change in the number of rake fingers
    - Changes could be requested by elements such as the RRC/RLC in a WCDMA protocol stack or a proprietary adaptation control unit.

Configuration Control Module

- Capability Interface
  - In a truly configurable platform defining precisely what functionality the equipment will support is problematic
  - This is due to the sharing of resources between functions e.g. a common data bus.
  - Example:
    - Will a platform executing WCDMA at 384Kbs support the data processing and timing constraints for monitoring GSM or 802.11a WLAN in dual mode operation ?
    - What would the answer be if we added an AMR speech codec ?
  - Simple solution is to constrain configuration to a known set.
  - A more sophisticated approach could be to dynamically calculate if a configuration can be supported

Configuration Control Module

- Database Interface
  - CCM contains local cache of executables used by CEM
    - Resource and Device Drivers
    - Object code
    - Metrics for use in configuration plane
  - Only items regularly used by the CCM are stored local to the equipment other items are referenced using a *handle*

Issues

- Granularity of configuration description
  - Low Granularity
    - Module corresponds to a complete RAT
    - Each module verified by supplier
    - Low flexibility
    - Small changes will require a large object code to be downloaded
    - Based on traditional approach except ROM, ASIC etc replaced with RAM and reconfigurable logic
    - Capability Description easier to implement. Can be based on Class Mark
  - High Granularity
    - Modules correspond to components in a RAT
      - e.g. Rake receiver, Channel decoder etc
    - Role of CCM is to combine modules to implement a complete RAT
    - Therefore operation of CCM must be verified to ensure compliance with RAT standard ?
    - High flexibility

TOSHIBA

Summary

- In E^2R we are developing a framework that will support the following:
  - Low power consumption
  - Support proprietary hardware architectures
  - Evolution from low granularity to high granularity
- Work is ongoing
- We welcome any comments !

TOSHIBA

Acknowledgment

This work is being performed within the framework of the EU funded project E^{2}R (http://www.e2r.motlabs.com). The authors would like to acknowledge the contributions of their colleagues from the E^{2}R consortium