Communications Research Centre Canada
An Agency of Industry Canada

Centre de recherches sur les communications Canada
Un organisme d'Industrie Canada

Deployment of Software Components into Heterogeneous SCA Platforms

Using the SCA with DSPs and FPGAs

Steve Bernier
Project Leader, Advanced Radio Systems
Communications Research Centre (CRC)
Agency of Industry Canada, Government of Canada

Outline

- Reusability through Portability
- SCA Deployment Basics
- SCA Devices
- Computational Elements
- Mapping the SCA to a Software Defined Radio

Reusability through Portability

Reusability Alternatives

- **Implementation Reuse**
  - interpreters or virtual machines
  - binary-compatible family of processors

- **Source Code Portability** **SCA**
  - Operating Systems (POSIX AEP)
  - Middleware (CORBA, MHAL)
  - Platform Devices (APIs)

- **Installation, Instantiation and Connection**
  - Core Framework

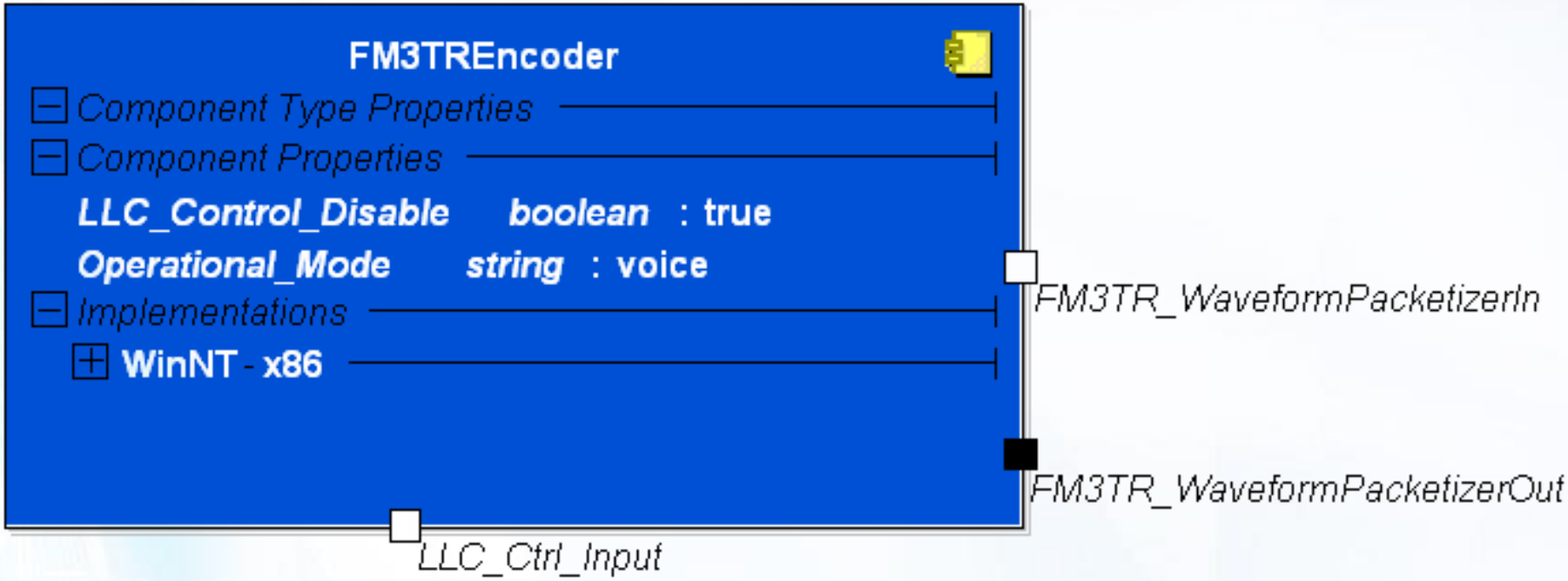
SCA Deployment Basics

Software Components

- SCA component is reusable binary code that performs a well defined function
- SCA Components are modeled as having ports to allow data flow and/or control
- SCA Component are modeled as having properties that can change their behaviour

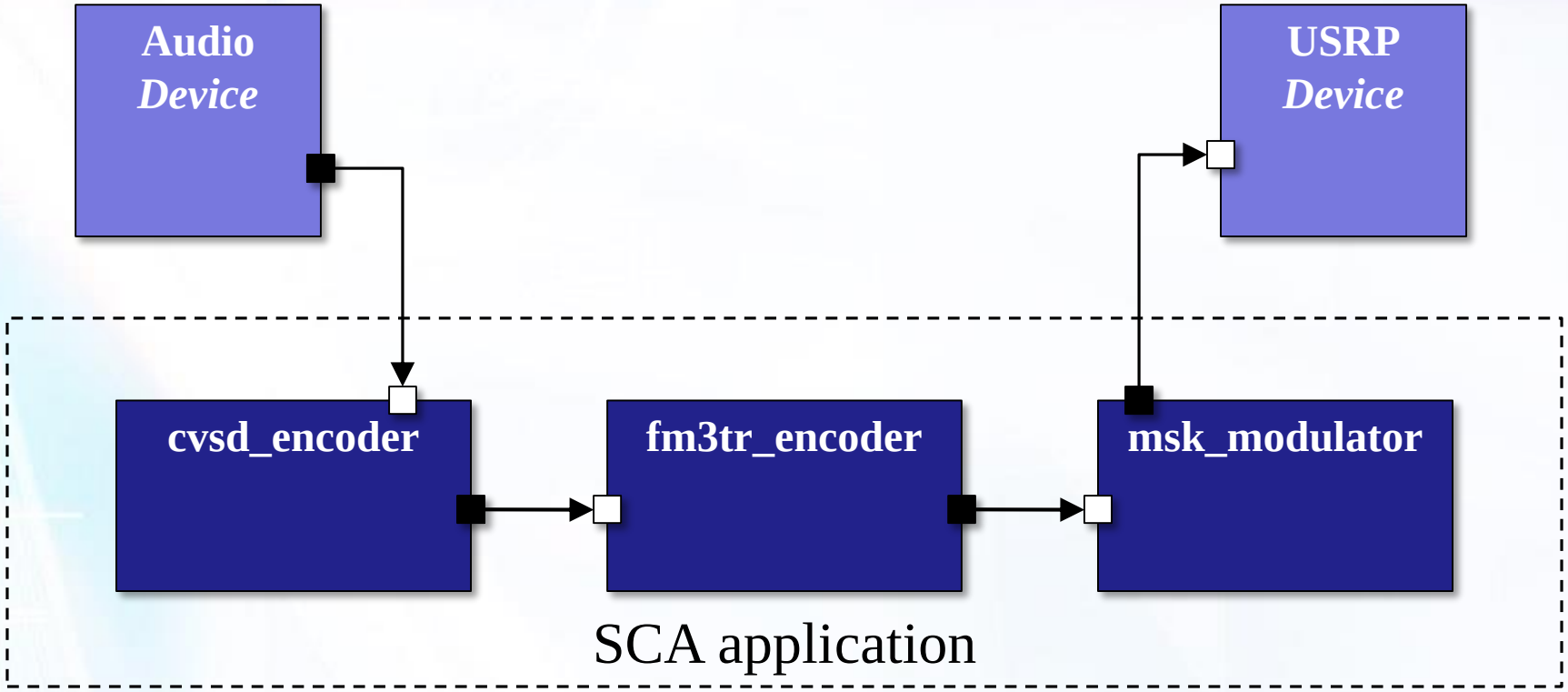
Software Components (cont.)

- **Typical graphical representation of a SCA component:**



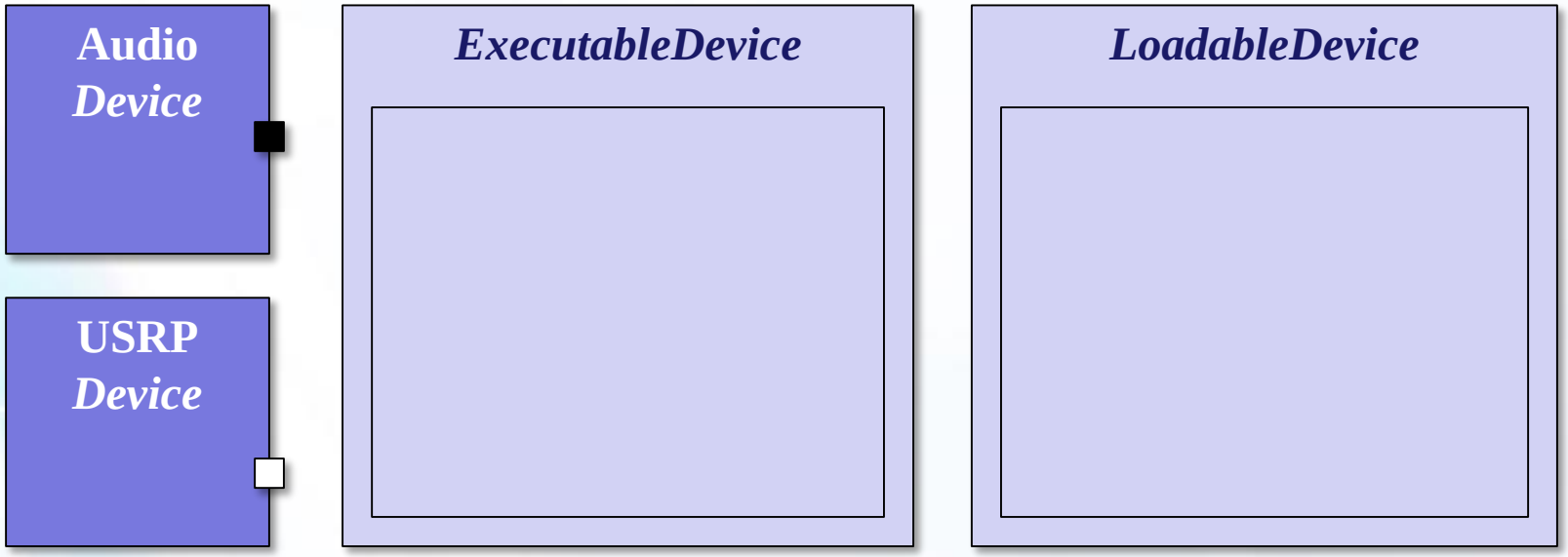
SCA applications

- SCA applications are composed of software components and connections



SCA Platforms

- **SCA platforms are made of software components called SCA *Devices* used as proxies to hardware components**

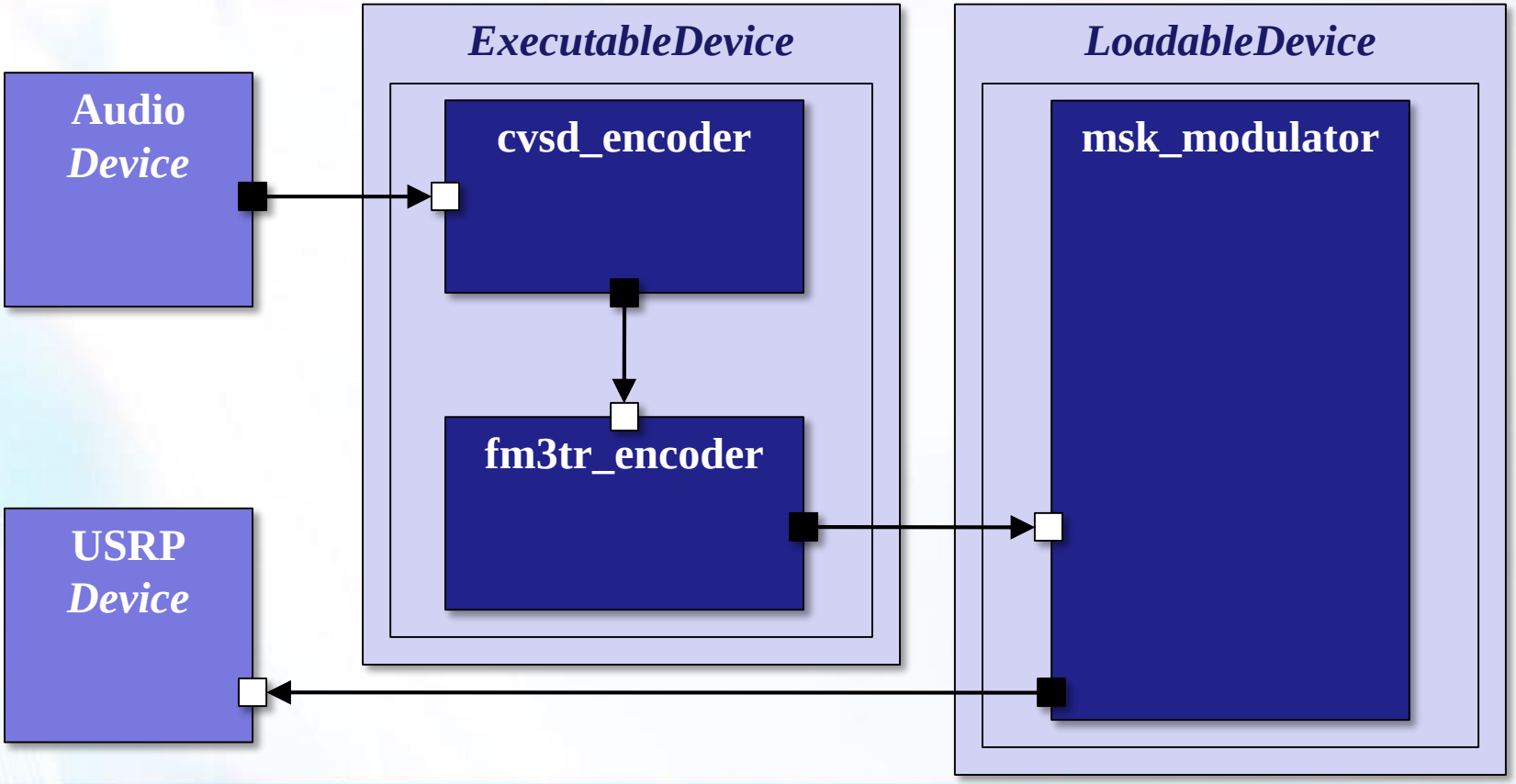


SCA Application Deployment

- ***LoadableDevices* and *ExecutableDevices* are containers for software deployment**
  - Used for loading and/or running software
- ***Devices* advertise properties while application components specify requirements**
  - Capacity properties (MIPS, RAM, etc.)
  - Capability properties (OS, processor, etc.)
- **Deployment of an SCA application is a matching process**
  - Requirements versus Advertisements

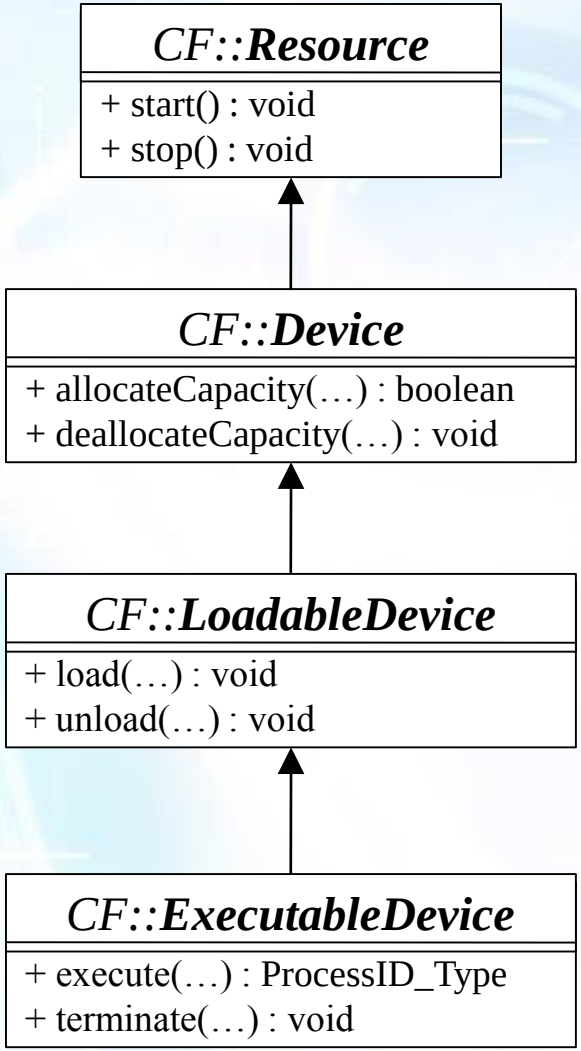
SCA Application Deployment

- SCA application components are deployed to *LoadableDevices* or *ExecutableDevices*



SCA Devices

SCA Devices



- ***Device***:
  - Capacity model
  - Cannot load anything

SCA Containers

- ***LoadableDevice***:
  - Can be used to load files (bit streams, executable code, etc)
- ***ExecutableDevice***:
  - Can be used to create multiple tasks

LoadableDevice

- **Used to represent single-load devices**
  - Used by the deployment engine to load artifacts of software components.
  - Capable of loading new artifacts, but not capable of launching new tasks (also known as processes)
    - Not capable of instantiating a component.
  - Loading the device will change the device's behaviour
    - Ex: loading an 'image' on a DSP
    - Cannot be used by two applications to load different functionality at the same time

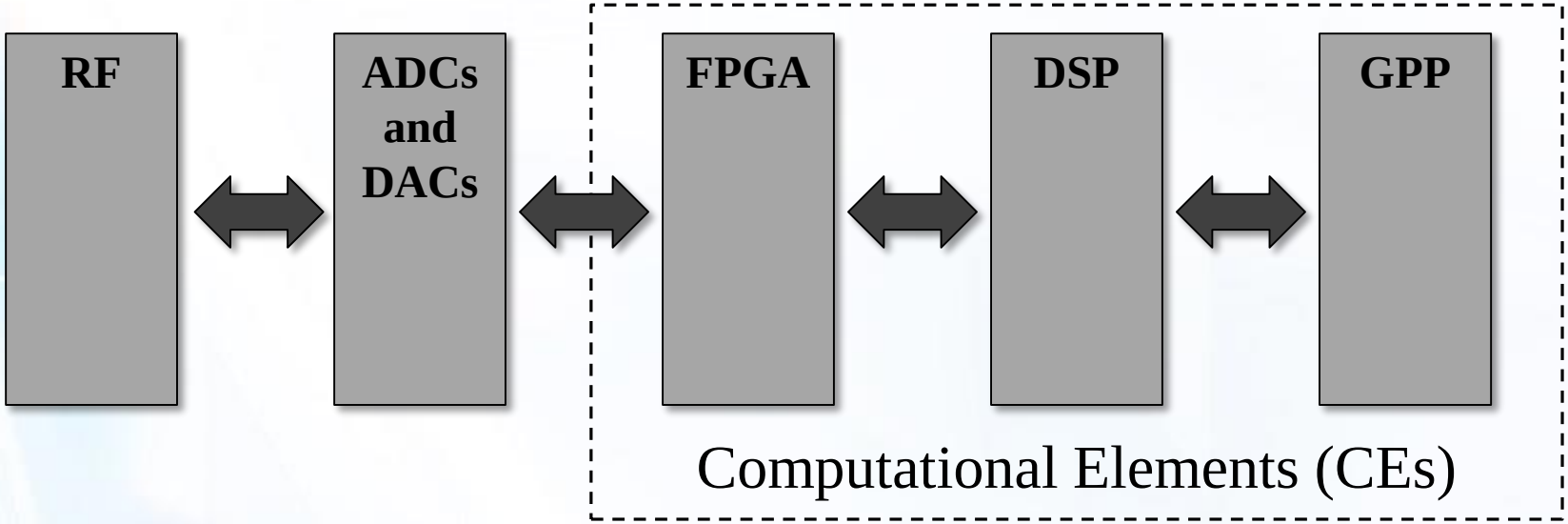
ExecutableDevice

- **Used to represent a Computational Element capable of running several software components concurrently**
  - Used by the deployment engine to load and execute artifacts of potentially many software components.
  - Capable of creating several tasks at the same time without having to reboot to launch a newly loaded task
    - Capable of instantiating a component.

Computational Elements

SDR Platforms

- **SDR platforms can provide three types of Computational Elements (CEs):**
  - General Purpose Processor (GPP)
  - Digital Signal Processor (DSP)
  - Field-Programmable Gate Array (FPGA)



Computational Elements

- **Field Programmable Gate Array**
  - Special purpose device used to implement complex logical circuits evaluated in parallel
  - SDR: Used for very fast and highly specialized RF/IF signals processing
  - Popular FPGAs:
    - Xilinx's Virtex family
    - Altera's Stratix family

Computational Elements (cont.)

- **Digital Signal Processor**
  - Special purpose processing unit designed for high speed arithmetic and high data throughput
  - SDR: typically used for baseband/IF signals processing
  - Popular DSPs:
    - Texas Instrument's C6000 family
    - Freescale's StarCore family

Computational Elements (cont.)

- **General Purpose Processor**
  - The “Jack of all trades, master of none” processor
  - SDR: GPP typically used for implementing MAC/networking layers
  - Popular GPPs:
    - Intel’s x86 family
    - AMD’s Kx family
    - FreeScale’s PowerQuicc family

Key Features

- **Incremental Loading:**
  - Allows new binary code to be loaded into execution memory during runtime without rebooting the CE.
  - Most operating systems that provide this feature do so through what is called a loader (usually requires a file system).
  - For instance, the POSIX standard provides access to a loader through the ‘exec()’ system calls.

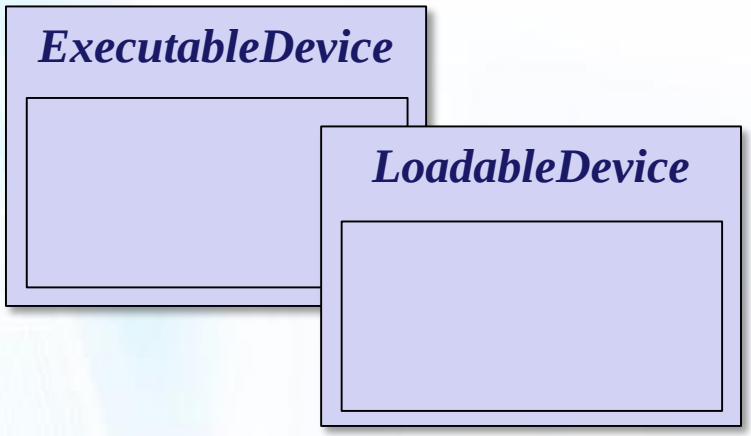
Key Features (cont.)

- **Multi-tasking:**
  - Allows multiple tasks (also known as processes) to share a single CE.
  - Allows several tasks to run concurrently without requiring a reboot every time a new task is launched.
  - Most operating systems implement this feature through what is called a task scheduler.

Mapping the SCA to a Software Defined Radio

Mapping the SCA to a SDR

- **How can the SCA concepts be mapped to a Software Defined Radio platform?**
  - SCA models in terms of application components being deployed to SCA Devices
  - While SDR platforms are made of Computational Elements that can be programmed



ExecutableDevice

ExecutableDevice

- **GPP: can be mapped as an *ExecutableDevice***
  - GPP Operating Systems always provide a loader
  - CRC's *ExecutableDevice* has been used as a proxy to processors such as x86, PPCs, ARM9, and Xscale using INTEGRITY, VxWorks, LynxOS, Linux, and soon Windows
- **DSP: can be mapped as an *ExecutableDevice***
  - Some DSP Operating Systems provide a loader along with multi-task support
  - Unaware of any SDR platform mapping a DSP as an *ExecutableDevice*

ExecutableDevice (cont.)

- **FPGA: can be mapped as an *ExecutableDevice***
  - An FPGA is in fact a parallel processing Computational Element
    - Provides multitasking without the need for a scheduler
  - Can provide support for incremental loading through ‘partial reconfiguration’
    - Can load new components into a ‘running’ FPGA without rebooting
  - CRC helped ISR Technologies support a Xilinx Virtex FPGA with partial reconfiguration using an *ExecutableDevice* (IDP-100 platform)

LoadableDevice

LoadableDevice

- **DSP: are typically used without an operating system**
  - This generally means no support for multi-tasking and no support for a loader
  - Once the DSP has been programmed, no new code can be injected without rebooting
  - In such a case, the DSP cannot be mapped as an ExecutableDevice
- ***Note:***
  - DSP/BIOS supports multi-tasking but does not provide a loader (and no file system)

LoadableDevice (cont.)

- **FGPA: is quite often mapped as an SCA LoadableDevice**
  - Loads one single bit stream
  - No new code can be injected without rebooting
- **GPP: is always used with an operating system that provides multi-tasking and a loader**
  - Always mapped as an *ExecutableDevice*

Conclusion

- **Most current SDR platforms provide all three types of Computational Elements**
  - FPGA, DSP, and GPP
- **Mapping a Computational Element as an SCA *ExecutableDevice* requires the support of multi-tasking and incremental loading**
  - Many RTOS provide a scheduler for multi-tasking and provide a loader for incremental loading
  - Even some DSP RTOS provide the two characteristics

Conclusion

- **Currently, DSPs and FPGAs are generally mapped as a *LoadableDevices***
  - Added complexity for post-manufacturing technology insertion
- **For fully flexible Software Defined Radios, Computational Elements should be mapped as *ExecutableDevices***
  - This does not necessarily require the use of a GPP!

Questions?

Steve Bernier, Project Leader
Advanced Radio Systems
steve.bernier@crc.ca
www.crc.ca/sdr