DEVELOPMENT OF A SCA 3.1 COMPLIANT W-CDMA WAVEFORM ON DSP/FPGA SPECIALIZED HARDWARE

Maxime Dumas, Lyrtech Inc., Quebec, Canada maxime.dumas@lyrtech.com
Louis Bélanger, Lyrtech Inc., Quebec, Canada louis.belanger@lyrtech.com
Sébastien Roy, Laval University, Quebec, Canada sebasroy@gel.ulaval.ca
Jean-Yves Chouinard, Laval University, Quebec, Canada chouinar@gel.ulaval.ca

ABSTRACT

This paper is a summary of the design method used to implement an SCA-compliant waveform on specialized hardware. As a proof-of-concept, the 3GPP W-CDMA waveform was selected, where focus was placed on the PHY layer. The specifications are implemented following the "platform-independent model to platform-specific model" (or PIM-to-PSM) design flow guidelines and targeted to a GPP/DSP/FPGA development platform. The paper also contains a discussion on what must be done to make specialized hardware SCA-compliant on heterogeneous platforms (blend of CORBA-enabled and non-CORBA-enabled processors). Difficulties associated with FPGA SCA compliance are presented, and solutions for creating location-transparent components are described.

1. INTRODUCTION

As the digital radio world is rapidly evolving, there is an increasing need for radio manufacturers and designers to seamlessly integrate different waveforms into their systems, and be able to reuse existing IP modules without putting too much effort on integration. Also, wireless service providers need to easily update or upgrade their hardware at minimum costs. The purpose of the SCA is to answer these exact needs in making waveform modules object-oriented, following OMG's OO-design guidelines. Although OO-design is well-suited for general-purpose processors that support high-level languages, modules that need to run on specialized hardware, such as DSPs and FPGAs cannot readily benefit from the same abstraction level. The JTRS specifies in its SCA SHS [2] how DSP/FPGA devices should be adapted and how software components should be developed, using an abstraction layer (HAL-C) to ease waveform portability and maximize design flexibility.

This paper opens with a brief summary of the SCA, especially for specialized hardware processors (SHP), and states SCA requirements for these devices. A description of the suggested design flow follows, emphasizing the challenges of implementing location-independent, relocatable, FPGA-targeted components and their possible solutions. Follows a description of the design methodology used to implement the W-CDMA waveform on the platform, including how and when in the design flow the SCA-compliance is integrated. Finally, a scenario of waveform reconfiguration is given as a case-study, and a discussion follows.

2. SCA/SCA SHS FRAMEWORK

The Software Communication Architecture (SCA) was developed as part of the Joint Tactical Radio System (JTRS) program to creating an abstraction layer between hardware platforms and waveforms to ease portability. This abstraction requires that waveform components be distributed and loaded on available processors by the Core Framework (CF), so that each component of a waveform is developed using an object-oriented modular design flow. This approach has the advantage of leading to generic, reusable, platform-independent waveform components.

Waveform components are truly implementation-independent, thanks to CORBA distributed processing middleware. While it is relatively straightforward to implement CORBA on a general-purpose processor (GPP), it does not lend itself readily to implementation on specialized hardware processors such as DSPs and FPGAs. Hence, there are two ways of configuring a DSP as part of an SCA platform. First, a DSP that has enough features can be CORBA-enabled through the implementation of a POSIX-compliant OS and a "C" ORB. Second, device-specific proxies are implemented to communicate with non-CORBA-enabled processing elements, as shown in figure 1. Because of hardware limitations or performance reasons, a DSP can also be part of an SCA platform as a non-CORBA-enabled device. In this particular case, the DSP environment must at least support the HAL-C API described in [2], which consists of four functions: HalcGetEndpoint(), HalcRegisterCallback(), HalcSend(), and HalcReceive(). For the CF to communicate with non-CORBA-enabled DSPs, logical devices representing them need to translate CORBA requests to HAL-C requests through a proxy.

Because waveform components are built as independent modules, they need to communicate through a common, standard interface on RTL-programmable logic (RPL) processors, designated to be Open Core Protocol (OCP). This interface makes complete abstraction of the bus configuration and arbitration, as implementation of these is left to the platform provider. On RPL processors (*i.e.* FPGA), components physically occupy static resources on the device (as opposed to instructions being loaded to data/program memory for DSPs). Hence, for components to be completely independent of the processors, they must be relocatable so that the CF can dynamically assign them to available resources (or areas) in the RPL processor's fabric. Moreover, RPL processors need to support partial reconfiguration to independently load components to it. Relocating RPL-targeted components is not a trivial task, especially if it must be done at runtime. The following section briefly presents the PIM-to-PSM design flow for creating SCA-compliant components. It then describes currently achievable ways to do so on Xilinx FPGAs.

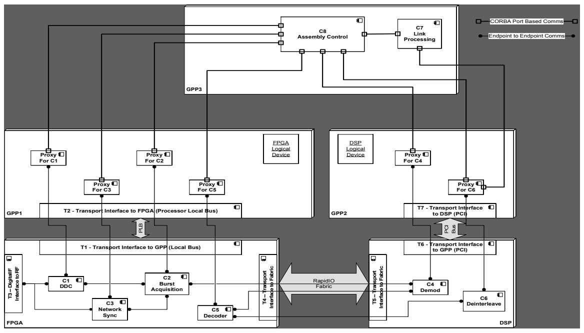

CORBA Port Based Comms
Endpoint to Endpoint Comms
C8 Assembly Control
C7 Link Processing
GPP3
FPGA Logical Device
DSP Logical Device
Proxy For C1
Proxy For C2
Proxy For C3
Proxy For C5
Proxy For C4
Proxy For C6
GPP1
T2 - Transport Interface to FPGA (Processor Local Bus)
GPP2
T7 - Transport Interface to DSP (PCI)
PLB
PCI Bus
T1 - Transport Interface to GPP (Local Bus)
T6 - Transport Interface to GPP (PCI)
T3 – DigitalIF Interface to RF
C1 DDC
C2 Burst Acquisition
C3 Network Sync
C5 Decoder
T4 – Transport Interface to Fabric
RapidIO Fabric
T5 – Transport Interface to Fabric
C4 Demod
C6 Deinterleave
FPGA
DSP

Figure 1: SCA abstraction layer on specialized hardware [3]

3. DESIGN FLOW FOR CREATING SCA-COMPLIANT COMPONENTS

All the design flows presented in this section refer to the PIM-to-PSM methodology defined by SCA, to ease waveform portability. This methodology is summarized here and starts with the definition of a platform-independent model (PIM), which contains what are called "golden sources", implementing waveform specifications without any platform-specific artifacts.

The first step towards the implementation-specific model is to derive targetable source code from the golden sources. This should be done for all potentially targeted SHP classes. For instance, a channel decoder could be derived from golden sources into C code as well as VHDL, so that it could be deployed either on a C-runtime processor (*i.e.* GPP or DSP) or an RPL processor (*i.e.* FPGA). The next step is to add the SCA interface that acts as a container to each worker, so that it can communicate on a SCA platform. These containers interact with the CF through proxies provided by the platform vendor. An optional step in the portability process is to optimize the code to target a specific device or device family.

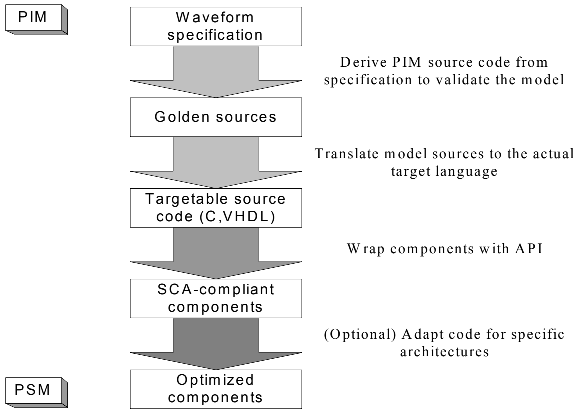

PIM
Waveform specification
Derive PIM source code from specification to validate the model
Golden sources
Translate model sources to the actual target language
Targetable source code (C,VHDL)
Wrap components with API
SCA-compliant components
(Optional) Adapt code for specific architectures
PSM
Optimized components

Figure 2: SCA PIM-to-PSM methodology

The standard design flow for creating relocatable components is to synthesize code, specify area constraints, then go through mapping, place and route, and finally bitstream generation. To relocate a component to a different area, the constraints file must be modified and the three last steps (above) be repeated each time. While this method is straightforward, it cannot be executed at runtime since mapping and especially place-and-route operations are time consuming. An adaptation of this method is to pre-generate several bitstream versions of the same component, each targeted at a different area in the FPGA, and store them in memory on the platform. However, this only partially fulfills the relocation requirement and consumes a lot of memory space.

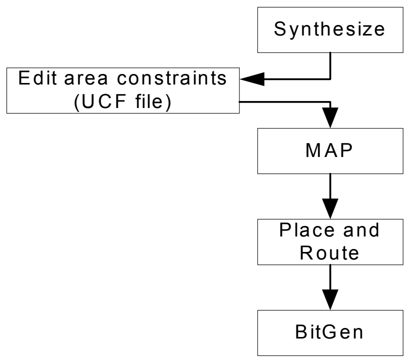


Figure 3: Standard methodology for relocating RPL components

With the design flow illustrated in figure 4, the placed-and-routed NCD file is modified so that the component is relocated to a different area of the FPGA. Because NCD is an internal and proprietary file format, it cannot be modified directly. Instead, the xdl tool must be used to convert it to the user-readable XDL file format so that it can be modified and converted back to generate the bitstream. Hence, only the BitGen and FPGA load steps need to be executed for relocation (aside from the modification itself).

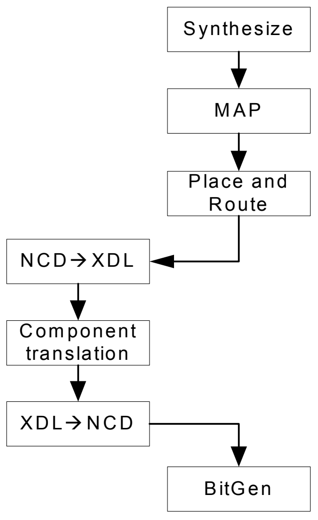


Figure 4: Advanced methodology for relocating RPL components

A placed-and-routed component can be relocated to a different area of the FPGA fabric by translating all resource addresses to target the new area. Of course, for the new, translated, placed-and-routed design to be valid, both areas have to be identical. Based on this restriction, two methods of dividing the FPGA in identical areas enable partial reconfiguration with component relocation. One is based on module-based partial reconfiguration, and the other relies on difference-based partial reconfiguration. (See [4] for details on partial reconfiguration methods.)

The first method consists in using columns as the base unit to divide the FPGA in identical, reconfigurable areas. The reason for this is that module-based partial reconfiguration is done by reloading complete columns (4-slice multiples). The problem with this method is that special resources (*i.e.* BRAM, MULT, *etc.*) are distributed column-wise, making it difficult to divide the FPGA in identical sub-areas. This method often results in coarse granularity, varying between 2 and 6 subdivisions, depending on the FPGA model and family.

The second method more efficiently exploits the distribution of special resources on FPGAs by dividing it into horizontal areas instead of vertical areas. To leverage partial reconfiguration with horizontal area division, difference-based partial reconfiguration needs to be used. Difference-based partial reconfiguration works by comparing two versions of a design at bitstream generation to create a bitstream that only contains the parts that are different from those of the already loaded bitstream. To be able to incrementally load individual components in the FPGA and obtain a sum of components forming part of a waveform and actually processing data, a base template bitstream must be generated. This template contains all the static parts of the FPGA design, such as the bus architecture, the HAL-C infrastructure, the HAL-C adapter, and the platform-specific I/O interface module (shown in figure 5). At design time, all components are synthesized, mapped, and placed and routed with area constraints. The constrained area must be a multiple of the "atomic relocatable unit", which is the smallest area that can be defined such that all resources of that area may be remapped to any other area of the FPGA through translation of addresses. This implementation results in a placed-and-routed NCD file for each component. A first waveform component is added to the FPGA by looking at which areas are available for new components. Once an available area is found, translation is performed from the area used at development time to the new area. After translation, the modified XDL content is added to the base design and converted back to the NCD format. The last step before loading the FPGA is to generate the bitstream with partial bit file option and specify the base design bitstream as the previous version to generate a difference-based partial bitstream. The subsequent components are added the same way. However, instead of referring to the base design, they refer to the last design created, thus leading to an incremental circuit build-up.

The main problem with the latter design flow is that XDL files can rapidly become space-consuming (in the order of 1 GBytes for a complete FPGA bitstream). While this method is better than the constraints-editing standard method, it may still take a long time to obtain a relocated bitstream and thus remains inadequate as a runtime solution.

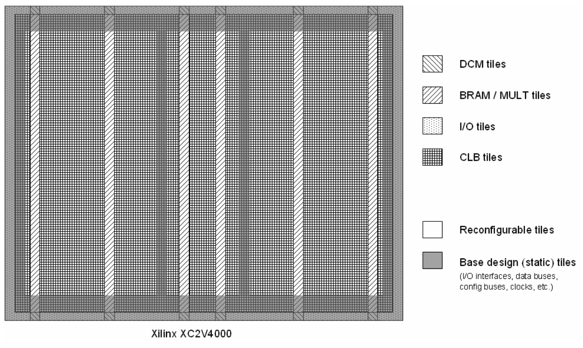

DCM tiles
BRAM / MULT tiles
I/O tiles
CLB tiles
Reconfigurable tiles
Base design (static) tiles
(I/O interfaces, data buses, config buses, clocks, etc.)
Xilinx XC2V4000

Figure 5: Floor plan of the base template design (static parts)

Waveform components impose one further constraint to be relocatable: their connection endpoints must fit with the base design bus endpoints. The "atomic relocatable unit" must contain fixed tiles that can only be used for interconnection with the interconnect bus.

While the methodology using difference-based partial reconfiguration may be applied to both Virtex-II and Virtex-4 families, Xilinx tool suite currently supports the module-based design flow only for the Virtex-II family. Partial loading is done by frame, and Virtex-II frames are complete, 4-slice wide columns, while Virtex-4 frames are 16-slice wide by 1-slice high, meaning that when module-based partial reconfiguration will be supported, it will be possible to use it with horizontally-divided areas.

As a final workaround, it is possible to relocate a complete design from one area to another by modifying the frame headers contained in the bitstream, provided, as for the earlier method, that both source and destination areas are identical. (This is not yet officially supported by Xilinx.) This mechanism can be operated with both partial loading methods. It is based on the principle that the smallest entity that can be configured is a frame and that, to be configured, each frame must be addressed. Thus, if a frame must be relocated to a different area, only the frame addresses in the bitstream need to be modified (and the checksum).

It is interesting to note that methodologies similar to the ones presented here have been explored, and that tools such as [5] and [6] were developed to facilitate partial reconfiguration, although none of them can be used in the present application.

4. SCA PLATFORM PRESENTATION

The platform on which the SCA-compliant waveform is implemented is the Lyrtech SignalMaster Quad, a member of the Lyrtech Signal Processing hybrid DSP/FPGA development platform family. Its cPCI 6U form factor is ideally suited for prototyping since it allows the developer to easily integrate additional hardware, such as A/D and D/A converters, host PC cards, or digital I/O. In the present case, a host PC card provides the GPP environment on which the CF will run. The SignalMaster Quad C6416/V4 consists of two clusters, each containing two TMS320C6416 DSP and a Virtex-4 FX FPGA. Each cluster can deliver up to 16,000 MIPS of DSP processing power and up to 70 GMACS of FPGA power, yielding a total of 32,000 MIPS of DSP power, 16-million gates system. The platform architecture is presented in figure 6. As can be seen, the DSPs are interconnected through FPGAs by high-speed parallel buses. The two FPGAs communicate through parallel LVDS buses while high-speed I/Os are connected with point-to-point parallel paths. The Lyrtech SCA Board Support Package consists of all interfaces and abstraction layers necessary to run SCA waveforms on it.

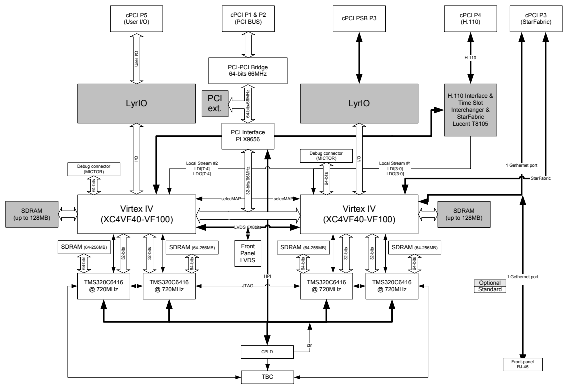


Figure 6: SignalMaster Quad architecture

5. 3G APPLICATION

Wideband Code Division Multiple Access is one of the most processing-intensive waveforms currently in use, mainly due to the fact that it consists of 5-MHz wide Direct Sequence Spread Spectrum (DSSS) QPSK modulated signals. Spreading the signal is achieved by OVSF sequences, which allow the multiplexing of several physical channels over the same frequency band. A scrambling sequence (Kasami code) is used to differentiate users over the same channel. The W-CDMA chip rate is 3.84 Mchip/s and has a bandwidth of 4.6848 MHz due to the root-raised cosine filter with a roll-off factor of 0.22. The chip period being 0.26 µs, resolution is good enough to account for multipath at the receiver. After being brought to baseband, the almost 5-MHz wide spread spectrum signal goes through despreading, multipath detection and combination, as well as symbol decoding. Next come deinterleaving, channel decoding (standard Viterbi decoding or turbo decoding), and block decoding (CRC).

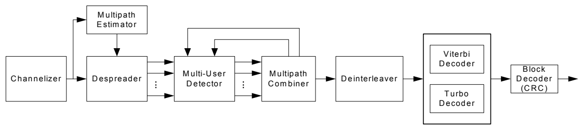


Figure 7: W-CDMA receiver block diagram

To demonstrate the functionality of the platform, the physical layer of 3G W-CDMA is being implemented. Most waveform components are developed in-house, except for the RAKE receiver, which was provided by École Nationale Supérieure des Sciences Appliquées et de Technologie (ENSSAT)[7], France, as VHDL IP cores. Following the PIM-to-PSM design flow, a platform-independent reference model was first created based on 3GPP specifications using a system-level approach with Simulink. As a next, Simulink blocks were replaced with C code for components targeted to DSP and with VHDL for components targeted to FPGA. Once the design is validated, wrapping must be performed for SCA-compliance. All modules were wrapped with a proper API, specified in SCA SHS 3.1 clause 2.4 or in SCACS 3.2, and then compiled/synthesized to the target devices, in this case Texas Instruments C64x family and Xilinx XC4VFX100. For VHDL components, constraints needed to be specified. The less space a component occupies on the FPGA, the better. It may thus be necessary to perform some iteration to find the right area constraints that will contain the design in the smallest possible area, while still respecting timing constraints. Also, since bus connections are placed at specific positions, the OCP interface must be constrained to these locations.

As a demonstration scenario, one could use the same waveform with two different front-end configurations, the first being that the RF signal is downconverted to baseband, digitized, and then sent for demodulation (in FPGA), in which case no DDC or filter is needed in the software waveform path. For the second configuration, several FDMA channels are provided in digital-IF and must be downconverted in software for channel selection before further processing. In this case, the channelizer component would be added in the FPGA since it is now required in the waveform path. Following the SCA methodology, the CF will look at the device's Device Package Descriptor file to find unallocated space where the new component may be loaded. After loading the component into the FPGA, the DCD file is updated and reconfiguration of endpoints performed. While there is a major difference between these two configurations, the only necessary modification is minor changes in the descriptor files (*i.e.* Software Assembly Descriptor), as the component and datapath changes are taken over by the CF.

6. CONCLUSION

SCA requires that waveform components have the ability to be individually loaded on devices and that their interconnections be specified afterwards, making them as modular as possible. As this can be achieved with CORBA middleware for some processors, SCA SHS details how it should be done for those that are not CORBA-enabled. Proxies on the CORBA-enabled processor side side translate requests so that non-CORBA-enabled processors can process them through a common API, while HAL-C (OCP) provides the common transport layer. In this context, location-independence of FPGA-targeted components is not a given, and is a thus problematic. Among all solutions presented, the most suitable uses bitstream frame address translation as a relocation method.

Although waveform implementation and platform adaptation is still in progress, everything is in place to construct a platform containing Specialized Hardware Processors SCA 3.1-compliant that will demonstrate dynamic reconfiguration of the W-CDMA waveform.

7. REFERENCES

[1] Joint Tactical Radio System (JTRS), Joint Program Office, “Software Communication Architecture (SCA) Specification”, JTRS-5000 SP V3.0.

[2] Joint Tactical Radio System (JTRS), Joint Program Office, “Specialized Hardware Supplement to the JTRS Software Communication Architecture (SCA) Specification”, JTRS-5000 SP V3.1.

[3] J. Kulp, M. Bicer, L. Pucker, G. Holt, Portable Waveform Components for Specialized Hardware, JPO Portability Workshop, January 2005.

[4] Xilinx Development System Reference Guide, 2005.

[5] Anup Kumar Raghavan, Peter Sutton, “JPG - A Partial Bitstream Generation Tool to Support Partial Reconfiguration in Virtex FPGAs”, Proceedings of the 16th International Parallel and Distributed Processing Symposium, p.192, April 15-19, 2002.

[6] Edson L. Horta, John W. Lockwood, Sérgio T. Kofuji, “Using PARBIT to Implement Partial Run-Time Reconfigurable Systems”, Proceedings of the Reconfigurable Computing Is Going Mainstream, 12th International Conference on Field-Programmable Logic and Applications, p.182-191, September 2-4, 2002.

[7] D. Menard, M. Guitton, S. Pillement, O. Sentieys, “Design and Implementation of WCDMA Platforms: Challenges and Trade-offs”, International Signal Processing Conference (ISPC'03), April, 2003.

Development of a SCA 3.1 compliant W-CDMA waveform on DSP/FPGA specialized hardware

M. Dumas, L. Bélanger, S. Roy and J.-Y. Chouinard

www.lyrtech.com

Outline

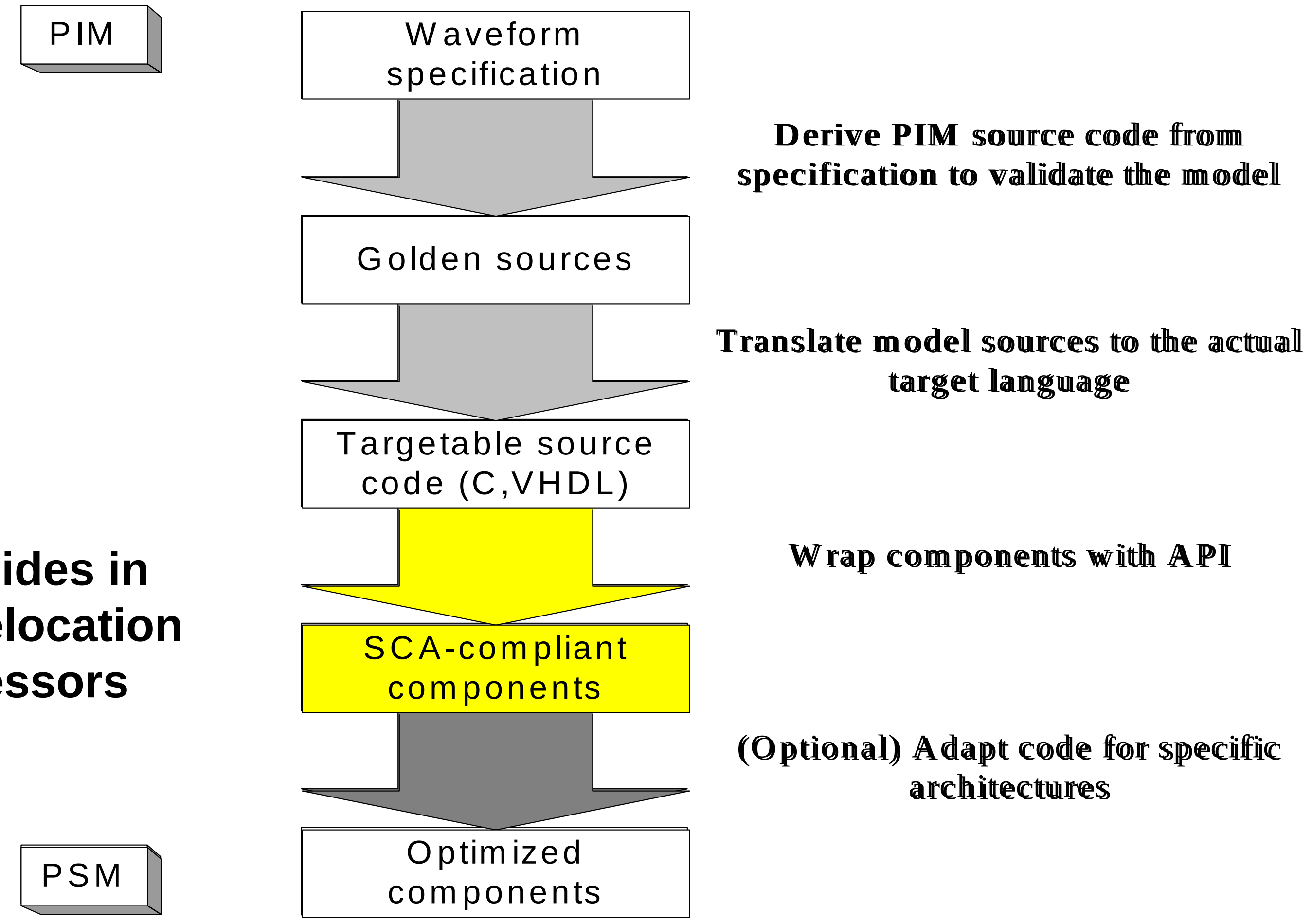
- Software Communications Architecture
- Design Flow and main challenge
- FPGA Component Relocation Methods
- Lyrtech Platform and Board Support Package
- W-CDMA waveform reconfiguration
- Conclusion

SCA / SCA SHS

Summary of objectives:

- Waveform portability
- Location-independence
- Waveform modularity
- On-the-fly waveform reconfiguration

Design Flow


PIM
Waveform specification
Derive PIM source code from specification to validate the model
Golden sources
Translate model sources to the actual target language
Targetable source code (C,VHDL)
Wrap components with API
SCA-compliant components
(Optional) Adapt code for specific architectures
PSM
Optimized components

Challenge resides in component relocation for RPL processors

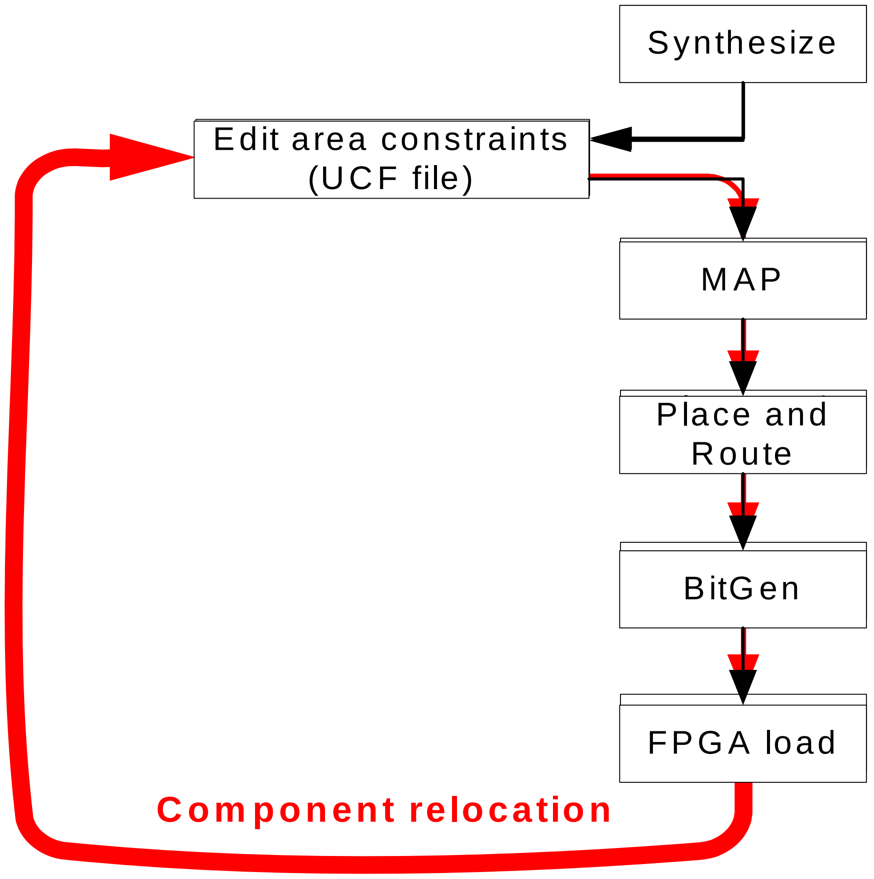
LYRTECH
LYRTECH SIGNAL PROCESSING

Component relocation

Standard relocation flow

• Time consuming

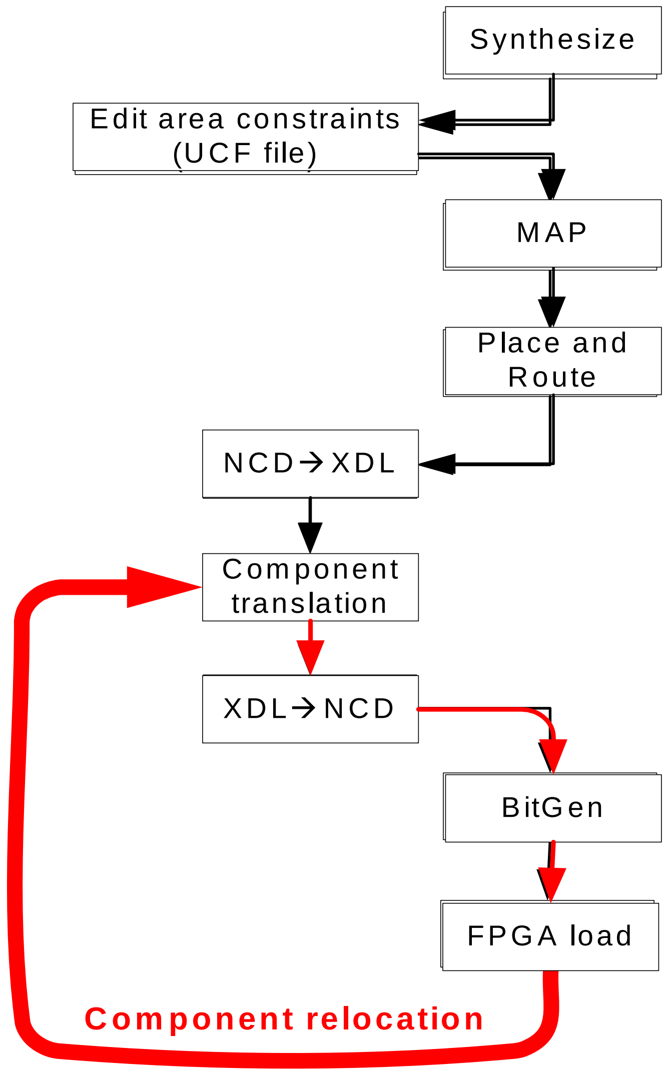
• Not for run-time applications



Component relocation

Post place-and-route relocation flow

- XDL files quickly become large
- Could be used at run-time
- Requires a large amount of resources (memory and processing)
- Translation is not trivial



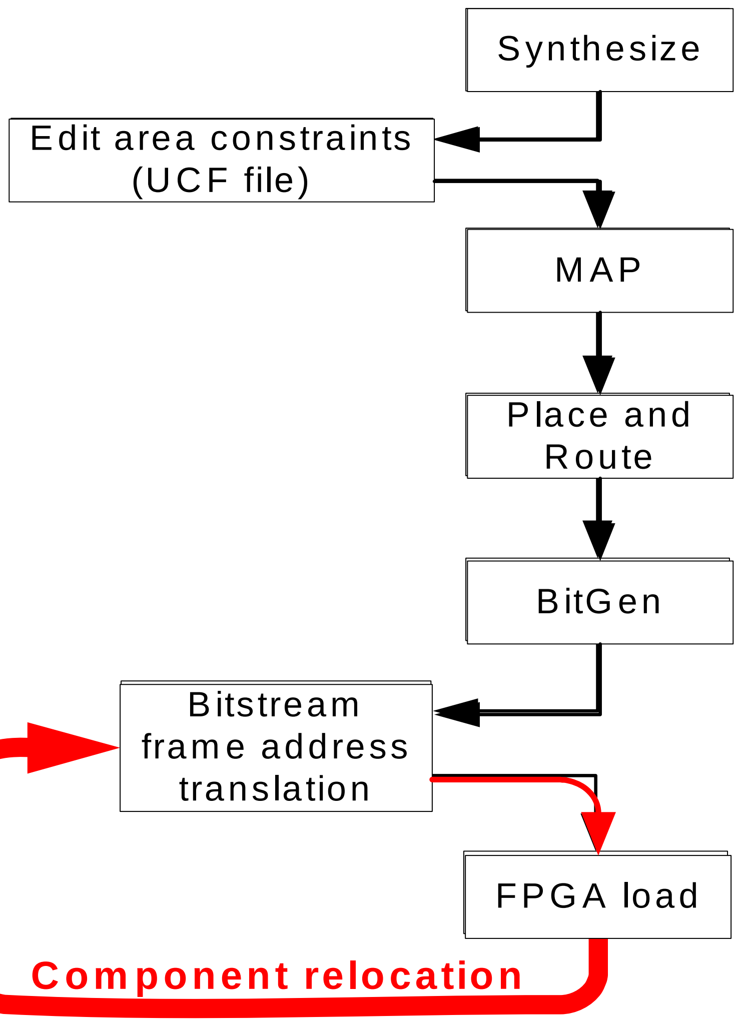
LYRTECH
LYRTECH SIGNAL PROCESSING

Component relocation

Bitstream relocation flow

• Only one operation : address translation

• Biggest file size around 1 Mbytes


Synthesize
Edit area constraints (UCF file)
MAP
Place and Route
BitGen
Bitstream frame address translation
FPGA load
Component relocation

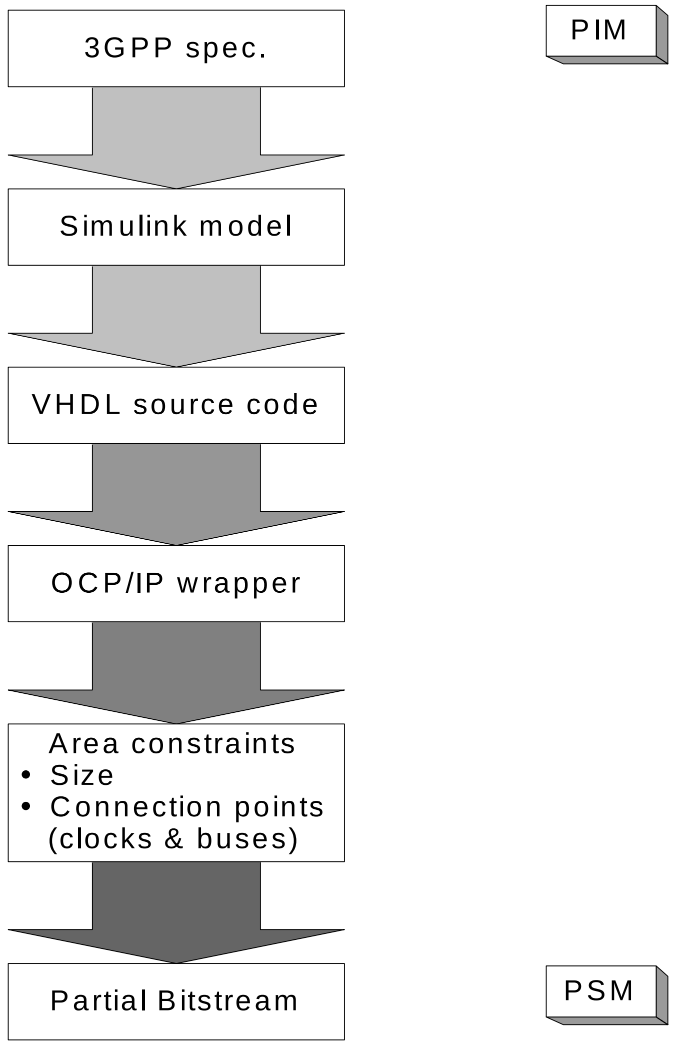
Partial reconfiguration

- Module-based
- Difference-based

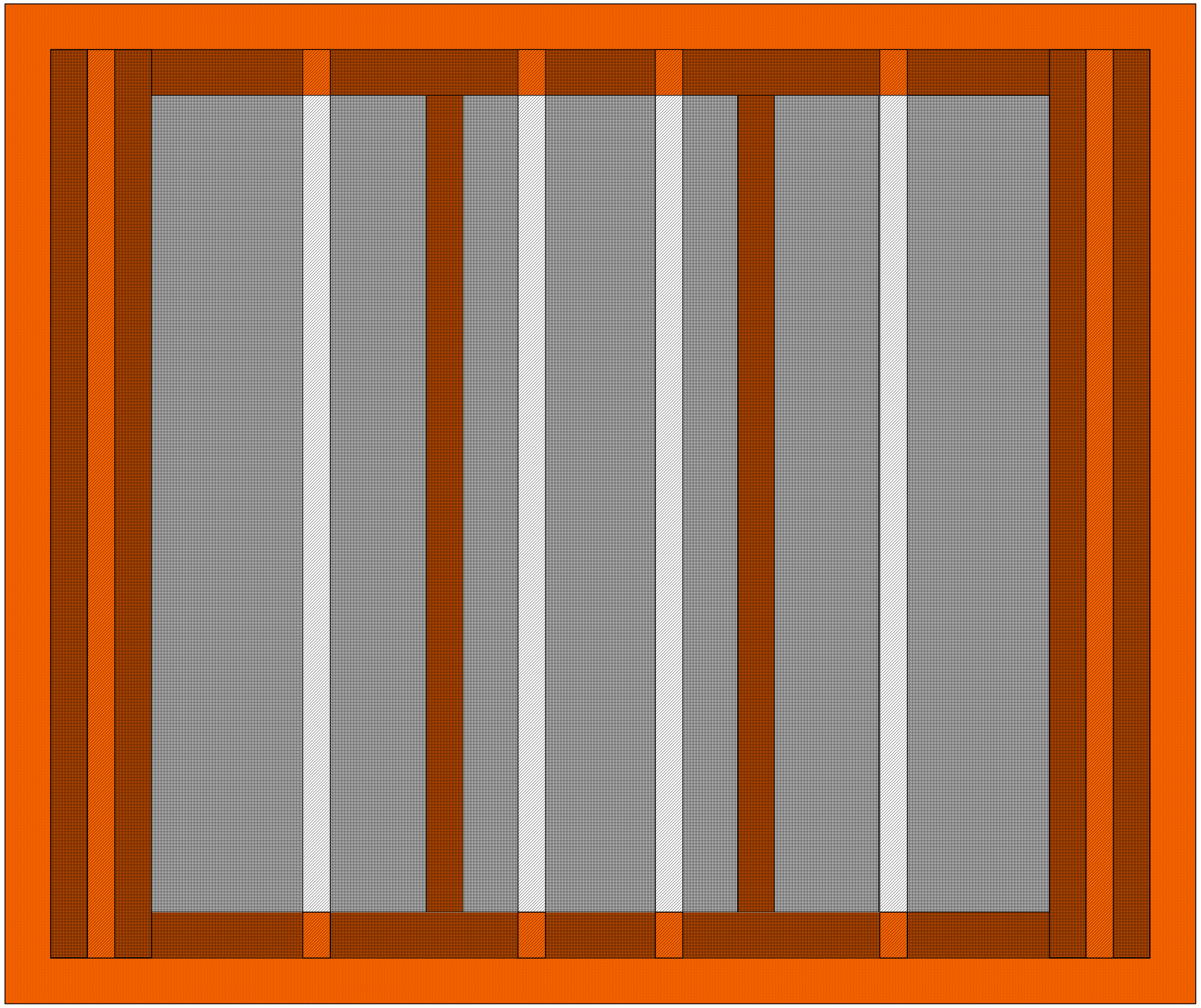
- Custom reconfiguration
  - Elaborate base design
  - Specify interconnection points for modules to connect
  - Build modular components with area constraints (size & connection points to common resources)

Design flow for FPGA

PIM-to-PSM design flow applied to FPGA



Base design floorplan



▨ BRAM / MULT tiles

▨ I/O tiles

▦ CLB tiles

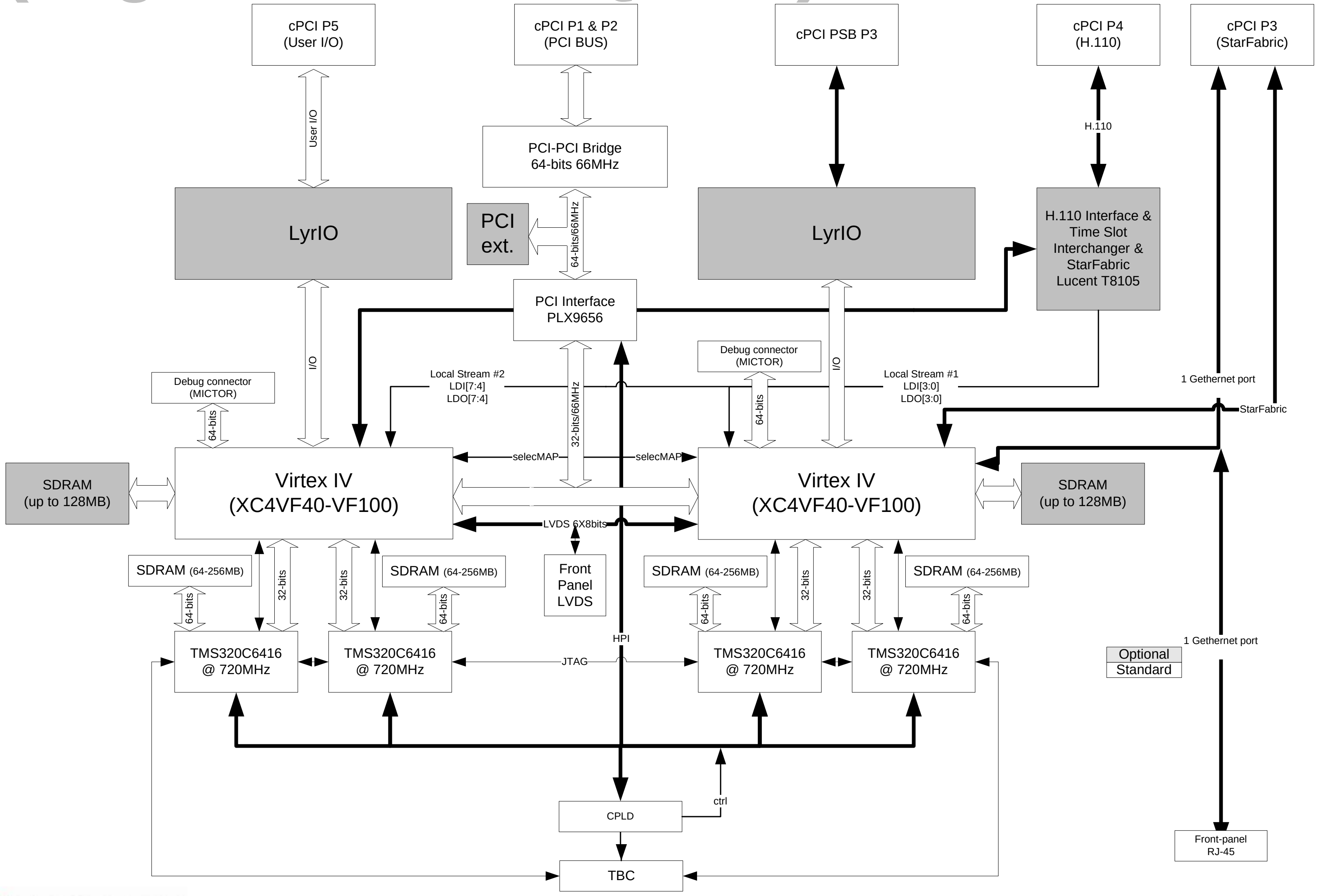
□ Reconfigurable tiles

🟧 Base design (static) tiles
(I/O interfaces, data buses, config buses, clocks, etc.)

Xilinx XC2V4000

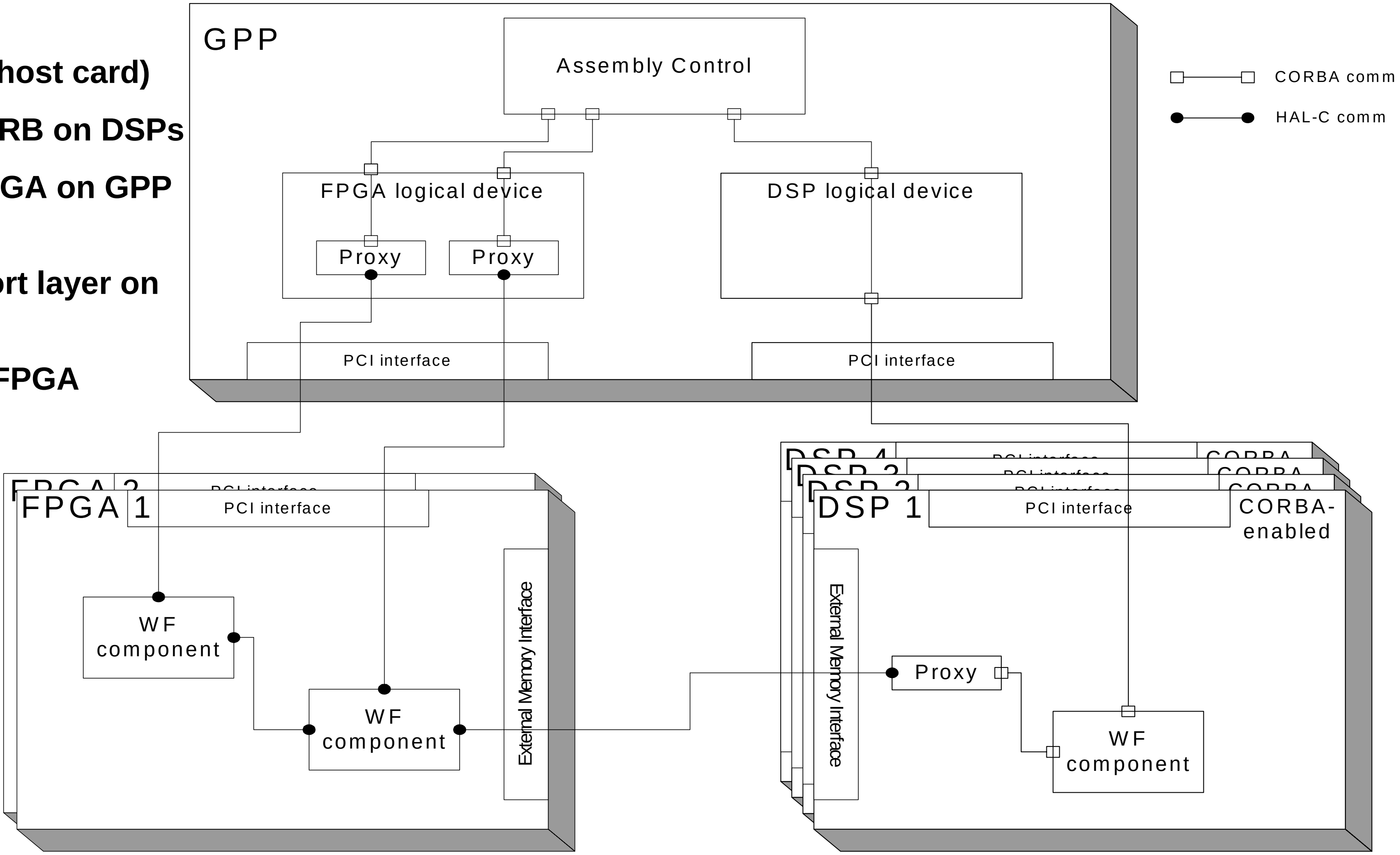
LYRTECH
LYRTECH SIGNAL PROCESSING

Lyrtech platform architecture (SignalMaster Quad)

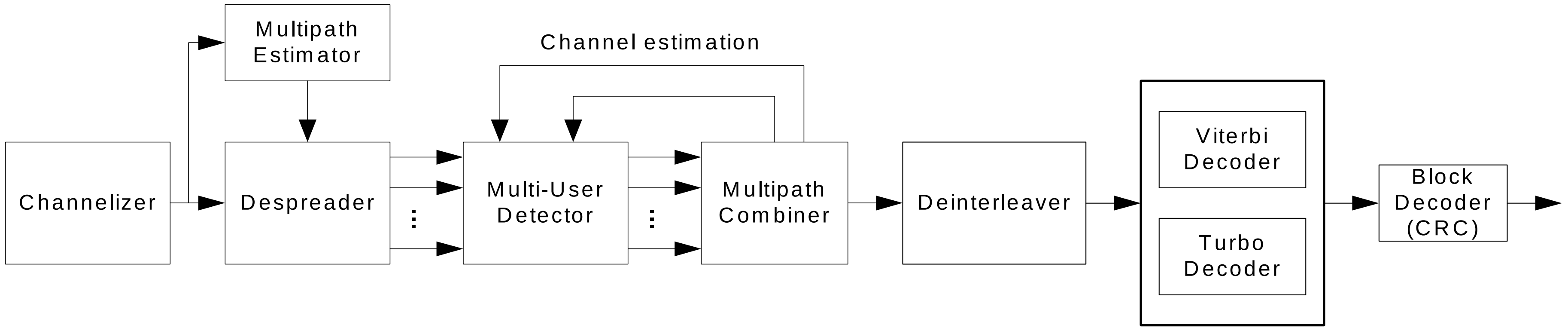


SCA Mapping to Platform (SCA Board Support Package)

- CF on GPP (host card)
- Prismtech ORB on DSPs
- Proxy for FPGA on GPP and DSPs
- OCP transport layer on FPGA
- common on FPGA



3G application (W-CDMA)



3G application (W-CDMA)

Why?

- Complex, processor-intensive waveform.
- Different versions of the waveform. (front-end, MIMO)

3G application (W-CDMA)

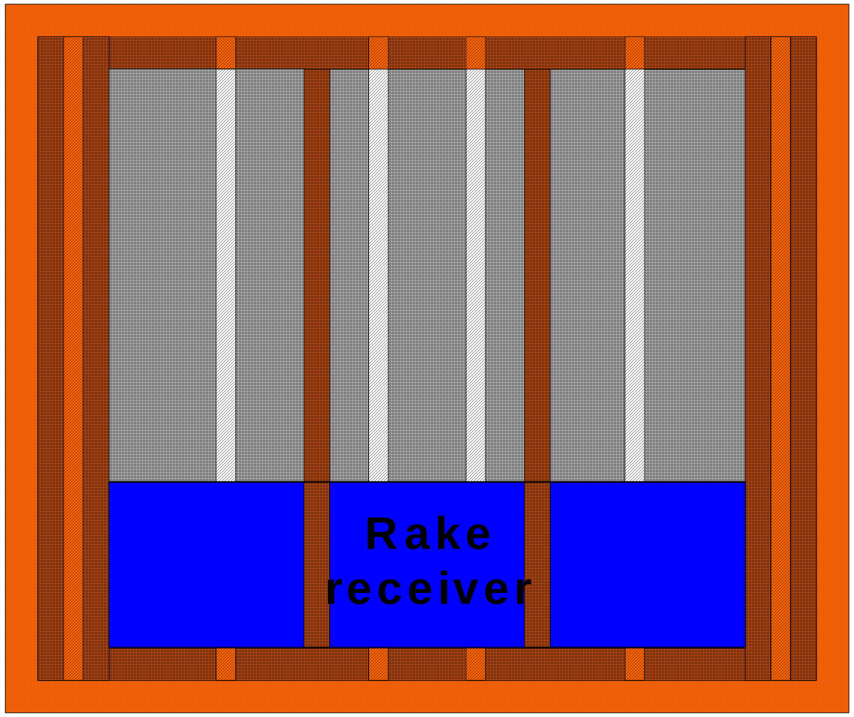
Scenarios

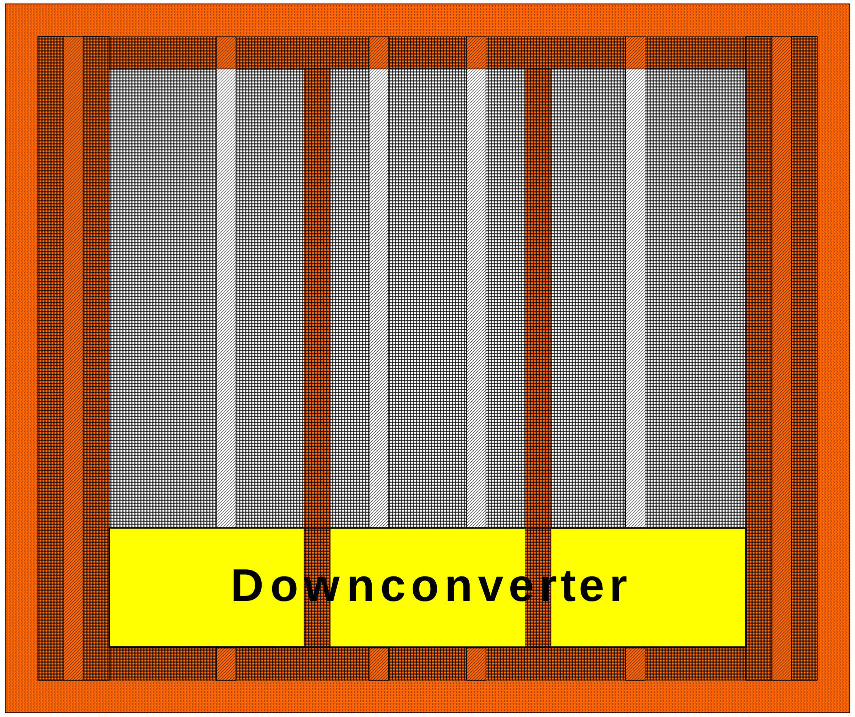
1. Baseband input signal

2. IF input signal

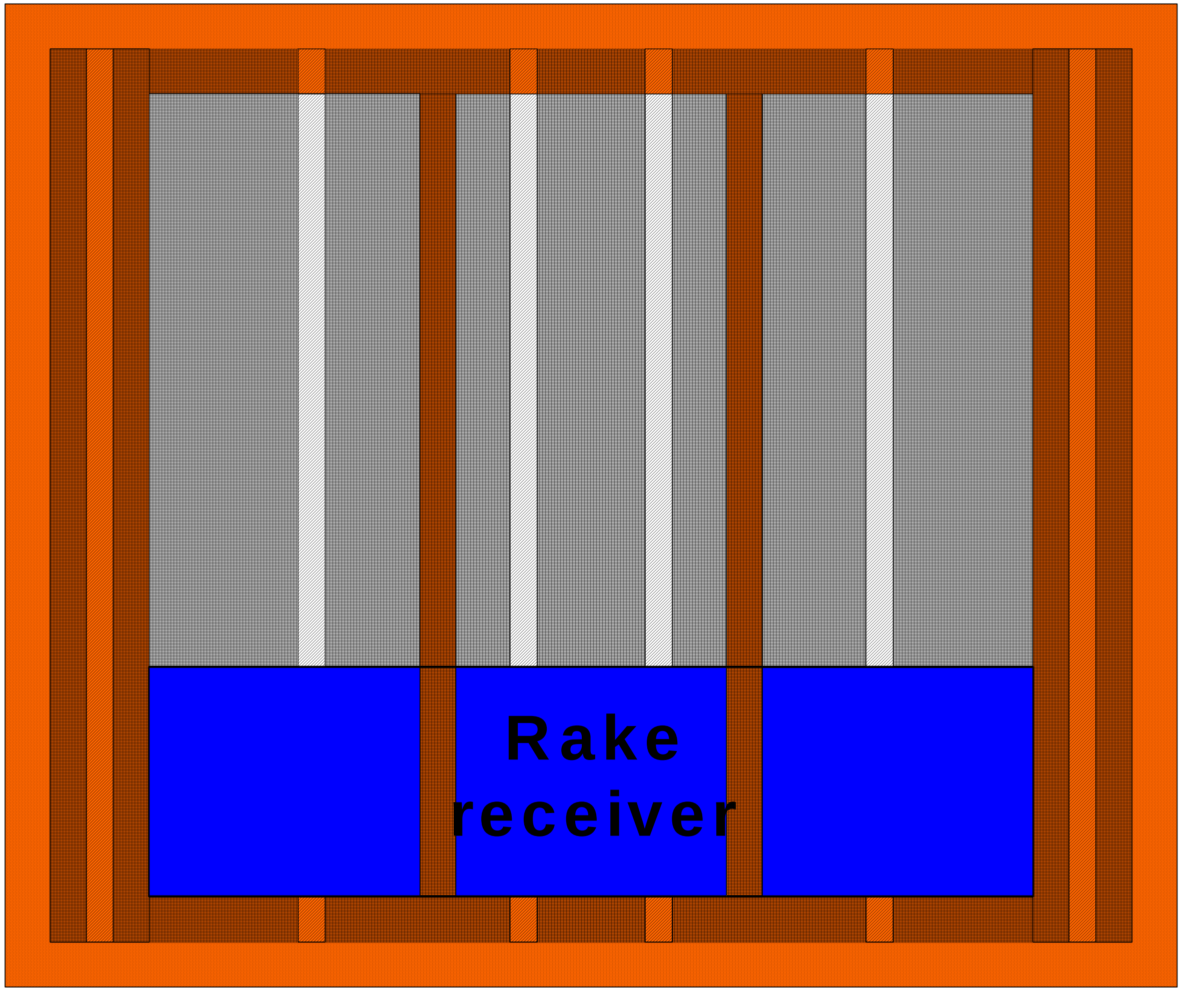
3G application (W-CDMA)

FPGA modules required





3G application (W-CDMA)



Scenario 1

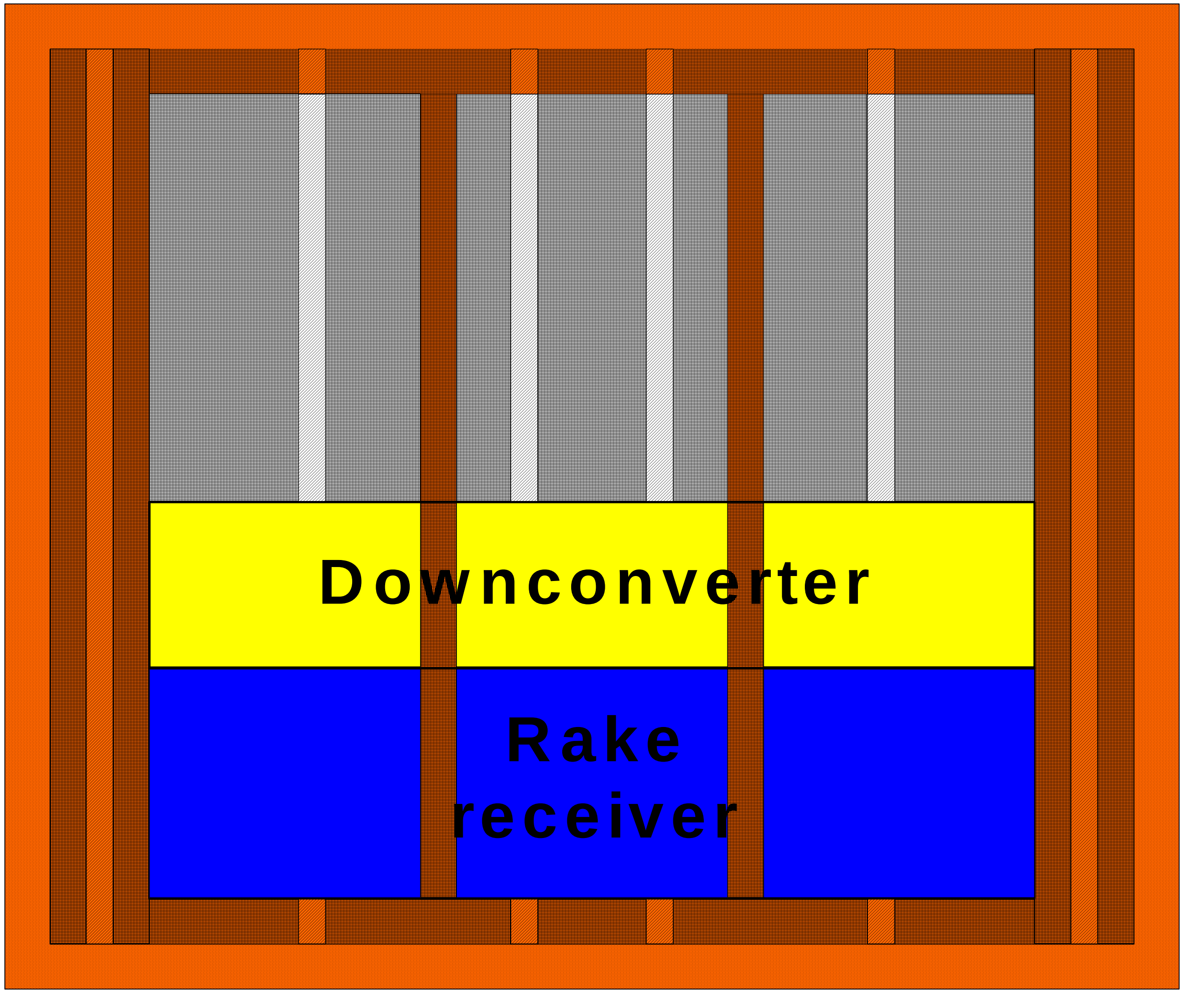
- Front-end feeds baseband signal to FPGA.
- Demodulated signal is sent to DSP for further processing

LYRTECH
LYRTECH SIGNAL PROCESSING

3G application (W-CDMA)

Scenario 2

- Multiplexed signal is received in IF
- Sent to downconverter module, then fed to the rake receiver
- Goes to DSP



LYRTECH
LYRTECH SIGNAL PROCESSING

Conclusion

SCA requires that:

- They can be individually loaded on devices
- Their interconnections be specified afterwards

What makes it possible:

- CORBA middleware
- SCA SHS for non CORBA-enabled processors
  - Common API
  - Common transport layer (OCP)

LYRTECH SIGNAL PROCESSING

Conclusion

Challenge:

Location independence on FPGA

Solutions:

- Standard relocation flow (post-synthesis)
- Post place-and-route relocation flow
- **Bitstream relocation flow**

Run-time reconfiguration as presented will be demonstrated with W-CDMA