"CORA-MBAD FOR ZYNQ 7000: MODEL BASED DEFINITION AND IMPLEMENTATION OF RECONFIGURABLE COTS AVIONICS"

Tiago Jorge (*), Laura Gouveia (*), Rubén Domingo (*), Fernando Pousa (*), David Arjona (*), Elena Alaña (*), Fabrizio Ferrandi (+), Thanassis Tsiodras (‡), Christophe Honvault (‡)

(*) GMV Aerospace and Defence
(+) Politecnico di Milano
(‡) European Space Agency

1. INTRODUCTION

The CoRA-MBAD activity ("*Compact Reconfigurable Avionics - Model Based Avionics Design*") was aimed at developing a HW/SW co-design toolchain providing functionality to easily deploy functional blocks in either HW or SW implementations, from identical source models. The toolchain developed for CoRA-MBAD was based on the TASTE toolset [1] and targeted a GR740 general-purpose processor coupled to a BRAVE reconfigurable FPGA. In this follow up activity, "*CoRA-MBAD for ZynQ 7000*", we adapt said toolchain to a ZynQ 7000 SoC target, motivated by low-cost missions that will use platforms based on COTS components such as this Xilinx SoC – which includes a dual-core ARM processor and a large reconfigurable FPGA.

2. OVERVIEW

To switch between HW and SW forms, the toolchain implements the automatic transformation of C source code (*whether manually written or generated by a code generator like those in Matlab/Simulink*) into Hardware (VHDL) source files. It additionally performs an automatic generation of the needed consistent communication interfaces supporting the exchange of commands and data between functional blocks executed on the processing system (PS) and on the programmable logic (PL) sides of the Xilinx SoC. This required adding support for the ARM Cortex A9 development toolchain (*RTEMS ARM support*), the Xilinx FPGA development toolchain (*Vivado*), and for the Advanced eXtensible Interface (*AXI*) communication interface for on-chip communication.

The toolchain was adapted to leverage the latest TASTE enhancements. The TASTE's Kazoo tool [2] was adapted to build the modeled systems with significantly increased build performance, especially in rebuilds. It efficiently produces derived models, code and scripts using AdaCore's "*templates-parser*" for templates processing and files generation.

For Matlab/Simulink models, the MBAD System relies on model-to-code transformation performed by MathWorks Embedded Coder [3] and on high-level synthesis of C code performed by Bambu [4]. Note that both TASTE and Bambu are open-source SW tools, so subsystems built in pure C can be synthesized and executed on the FPGA with no external dependencies. Bambu is FPGA vendor independent, hence it can be used with minor adaptations needed for each FPGA specific component.

The demonstrator use case is based on a computer vision algorithm that is used for vision-based navigation in the HERA project.

3. MODEL-BASED APPROACH

The complete automation and resulting cost effective extensibility made possible by the toolchain is not an easy feat to achieve since several elements need to come together, namely (see also Figure 1):

1) Basic SW and HW reusable elements: a) (SW) The RTEMS 5.1 custom built cross-compiler for ARM Cortex A9, equipped with the needed BSPs and validated on target. Additionally, some verification efforts necessarily have to target low level real time concerns such as CPU and task management,

clock frequency configuration, etc.; b) (SW) An AXI IP core control driver providing the necessarily interface configuration, initialization and read/write access to the SW applications. c) (HW) An AXI interconnect IP Core to manage and connect the AXI ports in the PS with the AXI ports of each of the modules/IPs implemented in the PL. At the same time, an AXI DMA IP controller to manage stream data transmissions between PS and PL.

2) Extensible model-based environments with high degree of automatism: a) TASTE provides heterogeneous application level modeling and implementation facilities, and importantly transparent and robust middleware level automation capabilities, in particular for communication aspects. TASTE's Kazoo allows for simple expansion and update of the supported targets, while improving code generation and build time; b) Vivado is an EDA (Electronic Design Automation) tool for FPGA and SoC, developed by Xilinx, with capabilities for low and high level synthesis, bitstream generation, timing analysis, simulation, etc. c) Matlab Simulink commonly used by domain engineers to design dynamic systems, e.g. the control and guidance of satellites designed by a GNC team, producing cyclically actuation data from sensor data, are best modelled with mathematics, data flows or functional models. This environment is extensible to e.g. incorporate as well autocoding facilities such as Embedded Coder.

3) A pivot open toolchain gluing all elements together: TASTE, being an open framework targeting heterogeneous systems, is particular suitable to integrate and orchestrate all the other necessary elements. E.g. from a common ASN.1 data model and an AADL minimalistic component interface model it consistently and automatically exports: a) interface definition in the target language of choice with consistent inputs and outputs (in our demonstrator a Simulink model); b) SW and HW wrapper interface code that transparently guarantees the correct communication between the target's functions. Importantly these interface wrappers automatically grow or shrink according to the number and type of inputs and outputs; c) SW device driver to provide SW-HW communication with the HW implementation of the target function; d) "Bridge" code with the necessary adaptations and extra inputs needed in the transition between two autocoding tools, in this case between Embedded Coder and Bambu. Additionally, TASTE e) integrates the custom cross-compiler (1-a) as part of a new deployment target, f) links with the necessary bus drivers (1-b), g) maps with the needed HW BSP exported from (1-c, 2-b), h) orchestrates the calls to all needed autocode and compilation tooling - e.g. forwarding the Embedded Coder output as a Bambu input together with the generated consistent bridge (3-d) and finally calling the synthesis facilities of Vivado (2-b).

4) Multifaceted team in co-engineering: The high technical degree of the activity required the diverse skills and close collaboration of a: a) SW engineer (1-a/b, 3-c/f), HW engineer (1-c, 2-b, 3-b/g), design environment engineer (2-a, 3-a/b/c/d/e/h), and domain engineer (2-c).

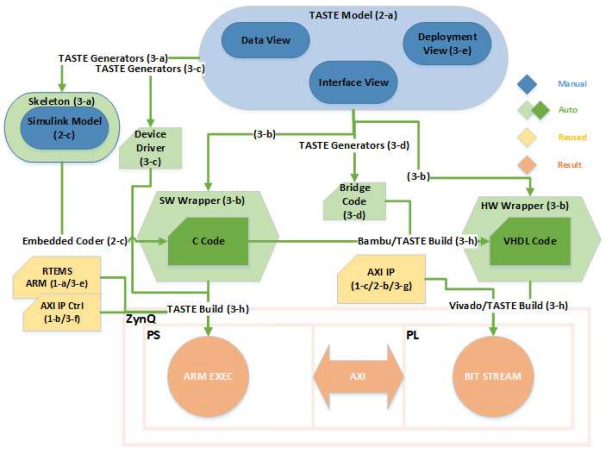

Figure 1 Toolchain overview

4. HW PROCESSING CAPABILITIES

CoRA-ZynQ makes use of AXI bridges of Zynq-7000 architecture to connect PS with PL. Three independent interfaces are implemented in order to provide different capabilities: One AXI interface used to write and read configuration registers, one AXI interface fully devoted to write and read large blocks of memory inside FPGA, and finally, one AXI

stream interface to support stream data processing. The number of registers or memories addressed through AXI interfaces can be configured to optimize the resource allocation of the FPGA. In addition, AXI stream transmission can be directed to achieve up to 32 different destinations through the same interface.

5. USE CASES

The use cases implemented have as prime objectives to 1) demonstrate the toolchain new target support and to 2) support a space representative application. Objective 1 was fully achieved with simple use cases. Objective 2 is presently partially achieved with work still ongoing. A preliminary version of the HERA mission computer-vision Lambertian sphere matching of asteroid body algorithm was re-used in this context. The Matlab design reused is not tailored for a HW implementation (e.g. no parallel nor fixed-point design) which naturally represents some challenges to the autocoding facilities and HW resource usage. Such tailoring was not yet performed due to project scope and time availability. Targeting prototyping activities, the present approach is instead leveraging to the maximum possible extent the configurability and autocoding strengths of the toolchain, avoiding any manual work, e.g. by exploring the rich Embedded Coder and Bambu options, types of possible SW-HW interfaces generated (e.g. external memory access, streaming type parameters) and resulting HW resource allocation.

[1] "TASTE," European Space Agency, [Online]. Available: https://taste.tools.

[2] [Online]. Available: http://taste.tuxfamily.org/wiki/index.php?title=Kazoo.

[3] [Online]. Available: https://www.mathworks.com/products/embedded-coder.html.

[4] P. d. Milano, "PandA," [Online]. Available: https://panda.dei.polimi.it/.