The University of Auckland
Department of Electrical, Computer, and Software Engineering
Zoran Salcic & Morteza Biglari-Abhari

COMPSYS 701 – Advanced Digital Design, Semester 1 2025

Heterogeneous Multiprocessor System on Chip (HMPSoC) for Mixed-criticality Applications

(Research Based Course assessed through two labs, a Group Project (GP) and an Individual Project (IP))

1. Introduction

The course centres on the topic of multi-core systems for mixed-critical embedded and real-time applications that clearly separate critical processing in terms of time predictability and safety from non-critical parts. The system uses physical isolation of these two parts of the execution platform and enables the two parts/subsystems communicate via standard bus and/or more general interconnection of a Network-on-Chip. Non-critical subsystem uses standard bus to connect communicating components, while critical subsystem is based on the use of time predictable NoC and network nodes that all offer time-predictability. Designing the platform, which can be considered as Heterogeneous Multiprocessor System on Chip (HMPSoC), includes advanced digital design techniques when designing individual components and architecting them into a complex HMPSoC. Those components may include existing general-purpose processor cores, new general-purpose time predictable processor cores, application-specific processors and network interfaces, using existing components in the form of IP blocks and their integration into HMPSoC. The course contains significant research, as well as advanced design of hardware and software components.

2. Motivation and background

Increasing number of embedded and real-time systems have computation requirements difficult to meet with standard, off-the-shelf available execution platforms. Those requirements go beyond capabilities of a single processor core and require multi-core platforms with large number of cores that have become a feasible and attractive alternative [1]. Communication requirements of programs that run on multi-core systems have naturally led towards various types of interconnect, e.g. standard buses, and networking of the cores in network-on-chip (NoC) based execution platforms [2]. While technology is not a bottleneck to design NoCs there are three issues that attract the attention of researchers: (1) what type of interconnect/NoC is suitable for certain types of execution platforms for target applications, (2) how to effectively use the cores to achieve target goals and design constraints of the applications and (3) how to guarantee time-predictability of the execution platform when dealing with hard real-time and safety-critical parts of applications. These issues are challenging and without definite answers. The cores of single type, which lead towards homogeneous multi-core systems, are not sufficient to satisfy requirements of complex applications as the utilisation of cores and performance of the overall systems can be inadequate. The solutions could be achieved more effectively with the use of different types of cores, which together are able to achieve the goals of the target system in the best way in terms of targeted metrics such as response times, predictability, and energy consumption. Such systems, typically implemented as a Systems-on-Chip (SoC) are referred to be heterogeneous as they comprise different types of cores [1]. In addition, employing heterogeneous MPSoCs helps optimising the power/energy consumption of the system. One approach that chip designers can provide is an execution platform, and then the design of the embedded system applications relies on using primarily software tools, which are not well developed for such platforms. The other approach can be to start the application development in a given design tool (e.g. specification language) and then synthesise the customised platform capable of executing the application by using certain off-the-shelf components (IPs), but also allowing incorporating design specific components, and connecting them using some kind of standard interconnect, e.g. bus or/and NoC. In other words, the platform can be customised for specific application by using processing and storage components that need to be interconnected and enable transfer of data each to the other. This way, hardware/software partitioning for co-design, based on application requirements and design constraints, would also be supported. FPGAs are the best current practical technology that allow prototyping and experiments with this design approach.

In this project we are working on the development of an execution platform for mixed-criticality systems that is heterogeneous in nature by incorporating different types of processor cores and application-specific components (in the form of IPs) that are interconnected using a combination of a standard bus and NoC. As the starting point we assume that we target real-time and safety critical applications which require guarantees in terms of functional properties as well as timing constraints for at least certain parts of the application. We explore the options of integrating, customising and synthesising a target platform using different components such as our own simple time-predictable RISC processor called ReCOP, standard Intel's (Altera's) general purpose processor Nios II as well as custom-made application-specific cores that accelerate the execution of signal processing or other algorithms. A general architecture of HMPSoC is illustrated in Figure 1.

In ideal case, our embedded and real-time system applications are developed in a new system-level language SystemGALS [3] that allows the use of a natural designer-specified parallelism and can exploit capabilities of multi-core execution platforms, including application-specific processors, and uses C as its host language. Although all tools for SystemGALS compilation and scheduling of executable code have not been developed yet, the paradigm of SystemGALS can be used to manually develop applications and generate partial executables of the overall systems as it was shown over years with its predecessor SystemJ GALS language [4] with fully automated tools. SystemGALS is a powerful concurrent programming and design language based on formal Globally Asynchronous Locally Synchronous (GALS) model of computation (MoC) that allows the specification and composition of very complex concurrent systems that target embedded and real-time applications [3, 4]. The language naturally supports explicit partitioning of the designed system on two types of computational units that correspond to two types of processing in complex embedded real-time systems:

(1) control-driven processing based on GALS MoC that incorporates synchronous-reactive concurrent MoC for describing complex concurrent control flows and globally asynchronous concurrent MoC for composition of synchronous reactive components, and
(2) data-driven processing where the computations require processing of data that arrives most often in streams and with high data rates. Additional motivation for using SystemGALS for the application design is that a designer can use it in different types of applications, from embedded non-real time systems to hard real-time and safety critical systems and the systems that are combination of both, we refer to them as mixed-criticality systems.

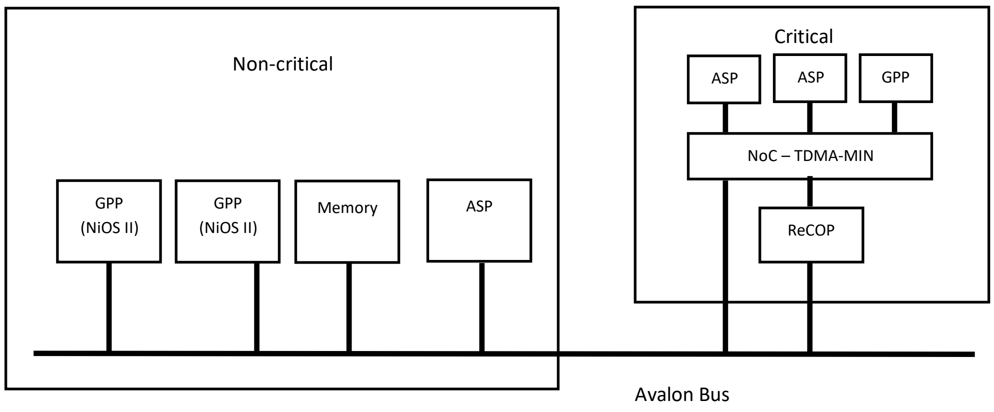

Figure 1 - General architecture of HMPSoC

3. HMPSoC – Heterogeneous Multiprocessor SoC

The overarching goal of the project is to investigate the architectural options and develop a prototype of the HMPSoC, which allows system designers to combine programming and hardware solutions and hardware/software partitioning and design. As an initial assumption, HMPSoC uses two major interconnect mechanisms (see Figure 1), Avalon bus (Intel/Altera) that allows interconnection of various components designed as master or slave components, and network-on-chip (NoC) developed specifically for time predictability necessary in critical part of the system [5] illustrated in Figure 2. Time-predictability is a pre-condition for static timing analysis of the execution time of the program or hardware component and estimation of worst-case execution times (WCET), as well as more general response times of the designed system. Here under “program” we generalise design specifications that are implemented in combination (composition) of software and hardware design units and corresponding behaviours when executing.

The HMPSoC’s critical part comprises different types of cores connected/networked using Time Division Multiple Access (TDMA) Multistage Interconnection Network (MIN), thus called TDMA-MIN [5], developed within Embedded Systems Research Group. A more detailed diagram that presents the TDMA-MIN interconnect fabric is shown in Figure 2.

TDMA-MIN allows connecting any pair of nodes (processor cores, for example) via NxN switch-type interconnect, where N is maximal number of connected nodes. TDMA-MIN can be synthesised for any N=2^k (k=1, 2, 3,…). In case of HMPSoC, it comprises at least one ReCOP processor core and the remaining cores are typically ASPs, but also can be general purpose processors (GPP) such as a Nios II processor core. The NoC provides data transfers between pairs of nodes without collision [6] and guarantees bounded latency (in terms of clock cycles) between any two nodes in NoC. In order to ease

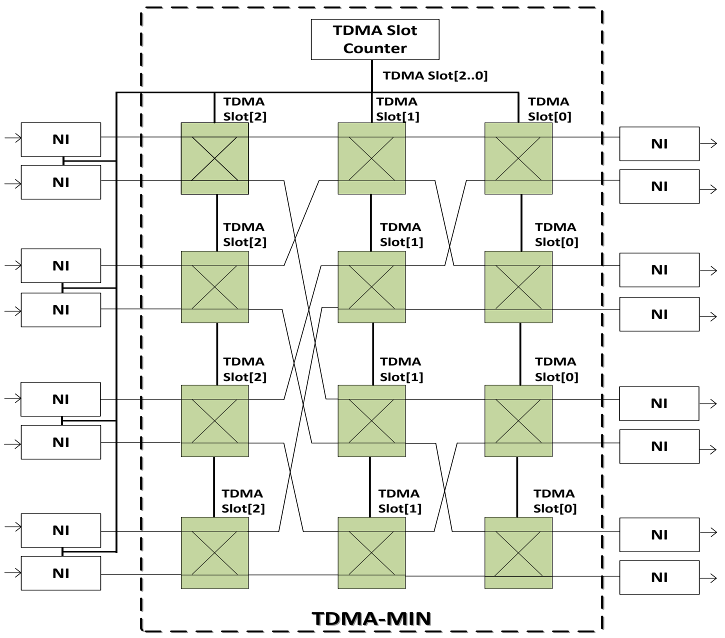

Figure 2 - TDMA-MIN Interconnect with the network interfaces (8x8 nodes)

applicability of the interconnect fabric, network interface (NI) can be uniform for all nodes, although some specific nodes can have their own NI.

4. Major goals (milestones) and constraints

The major goals of the GRP and IRP projects are:

1. Full understanding of the overall architecture of HMPSoC and its aims to enable design and implementation of instances of architecture customised to specific application or application domain; understanding System-on-Chip (SoC) and Network-on-Chip (NoC) concepts generally and of TDMA-MIN NoC, specifically; understanding of all interconnects used to make the SoC (TDMA-MIN and Avalon-bus), and how interfaces of individual nodes are made to provide connectivity.
2. Ability to critically read, analyse and assess existing research and other literature including existing provided designs.
3. Ability to design fully operational ReCOP RISC processor with necessary customisation including the assembler that supports the addition of new instructions.
4. Design an application specific processor (ASP) that can be interfaced with TDMA-MIN NoC and/or Avalon bus; functionality of the ASP will be specified separately within Individual Research Project (IP); the ASP is in the domain of performing digital signal processing algorithms in custom digital hardware implemented in an FPGA.
5. Integration of components into SoC using Intel FPGA Platform Designer.
6. Use of VHDL to design the required components of the system at RTL level and to simulate before synthesis using ModelSim as a part of system design flow.

7. Prototyping with Intel/Altera FPGA on DE1-SoC prototyping board.
8. Testing parts of the system with software developed using SystemGALS style specifications; those specifications will be manually mapped on software in C and ReCOP assembly language as well as on ASPs designed using VHDL.
9. Working in a team with understanding own and other members' responsibilities and roles and respecting the contributions of all members of the team.

In order to be manageable and achievable, the project will assume the following design constraints as a starting point:

1. A single type of new general purpose RISC processor called Reactive and Concurrency-Processor (ReCOP) [7, 8, 9] that allows easy customisation to satisfy the requirements of the applications by simple internal changes, and Nios II as given mature general purpose processor (GPP). While Nios II processor is primarily used in implementation of non-critical part of the system, it is also used as support for configuration and initialisation of critical part. Also, it can be used for the implementation of SystemGALS concurrent software behaviours and data modules [3].
2. Different types of Application-Specific Processor (ASP) cores, some of them are specified further in this brief that target specific application of power system frequency relay, and further detailed in a separate individual project (IP) brief.
3. Network-on-Chip based on TDMA-MIN interconnect [5, 6].

Due to the project's time constraint, our goal is to develop an experimental platform that has limited capacity which includes all major elements ready for up-scaling. For the critical part, it comprises at least one ReCOP, minimum 4 ASPs, at least one Nios II core and provides 8 ports in the TDMA-MIN NoC, which allows connecting up to 8 nodes/cores. Nios II, which uses one NoC port, extends the connectivity to Avalon bus and non-critical part of the system.

As one of the sub-goals is to master full, yet simple, processor design; the provided ReCOP core initial specification, given in the form of high-level programming model of the processor that includes Instruction Set Architecture (ISA) needs to be expanded to full design and implemented as register transfer level (RTL) design.

Project will be conducted through a number of phases and tasks in order to make it easier to manage and reduce the complexity of design process. After the analysis and understanding of the overall goals, a careful plan and alignment with each phase is the key for the successful completion of the tasks. Besides the team effort, which constitutes 60% of the marks, the project contains an individual component (IP) of exploring and designing an ASP, which constitutes 20% of the final grade of each student. The lab assignments are in groups of two students, but treated as individual work, and individual work on an ASP is a choice of each student. Creation of groups for the assignments and main project will be explained in lectures. Small group consists of two students, large group of four, but a large group incorporates two small group members. All issues in the group/team function, operation and collaboration have to be indicated immediately upon they arise to the academic staff and course coordinator by either the team or individual members of the team.

5. Background on processor design

In order to prepare for processor design, within the lectures we are going to present full design of an early pioneering reactive microprocessor called ReMIC, including rationale for its design, description of its programming model, ISA, datapath and control unit, as well as potential for user and application-specific customisation. First, it will include the design of non-reactive core of the processor [10], followed by the rough description of the design and customisation for reactive systems [11] in multi-cycle version. A pipelined version will not be discussed in 2025. This will be a good background for the design of ReCOP processor, as the first phase of the GP.

6. Application domain for ASP

We envisage as a target application which requires certain level of

(1) **Reactivity** on external inputs and events (such as pressing of push buttons and/or sudden changes on external inputs, which indicate the need for change of control flow within the application and internally generated events that mean sudden change of signal values detected during system operation).
(2) **Processing of streams of data**, which come to the system (sourced from analogue signal and an ADC converter) presented as time series of samples, processing of those samples using signal processing algorithms as specified below or cascade of those algorithms, and conversion of the digital results into the output stream/time series ready to be converted into analogue form (via DAC converter).
(3) **Support and non-critical functions** that will be used in system operation that execute on Nios II processor (such as configuration of critical part of the platform, preparation of data, change of program object code of critical part, debugging, customisation of instruction set, communication with the designer team etc).

For the purposes of this course you will design a simple tool for power system signal analysis and measuring instantaneous frequency, and then reacting in real-time if the frequency or its rate of change breach some thresholds. For this purpose, the power system signal is converted in to a sequence of samples which are processed in three main ASPs. Additional ASP(s) are created to enable emulation of the real connection to the power system network. ASPs will implement algorithms that are not suitable for software implementation due to inability of processors (ReCOP, Nios II) to guarantee (hard) real-time requirements. Concrete design specification will have to be made by each group/team.

In individual project (IP) each student will have to implement or two ASPs with a set of features described in IP brief. All students in a team/group will design different ASPs consistent with the overall goal that they have to be cascaded into a chain (pipeline) that satisfies application requirements without software intervention. Examples of typical computations on the data streams are:

- Passthrough of data
- Filters (e.g. Linear, Finite Impulse Response (FIR), Infinite Impulse Response (IIR), Kalman)
- Peak (local) minimum / maximum detection in the time series
- Average over a moving/sliding window (sample based or block based)
- Signal autocorrelation and correlation
- Digital signal generation/synthesis,

For these refer to the individual project (IP) brief.

An Input ASP (emulating an ADC) can be made with a set of virtual channels and sampling rates, as well as an Output ASP that provides an analogue output from the prototyping board; they will be also referred to as IO-ASPs. ASPs that perform data processing (computations) will be referred to as DP-ASP. If IO-ASPs have more specific functionality, for example emulating ADC and DAC operation, they can be referred to as ADC-ASP and DAC-ASP, respectively.

For the initial phases in the design process, you may use a customised state machine ASP (instead of ReCOP) to send configuration messages over the TDMA-MIN interface and configure other ASPs; this will allow teams to work on the NoC and ASPs in parallel with the development of ReCOP. For this you can also refer to Lab 2.

The functional application will be expected to take/read configuration values from switches on the development board to:

- Select the input channel and sampling rate
- Enable a combination of each of the ASP provided filters for each channel:
    - Direct passthrough
    - Linear filtering
    - Averaging
    - Peak Detection

  Data formats and type of expected arithmetic will be determined separately and kept at fixed point or integer arithmetic only. There will be no need to design or use any dividers. Division operation will be limited to the numbers that are 2^k, where k is limited to 0, 1, 2 and 3.
- As an example, to configure an averaging filter mode the ReCOP would send a DAC-ASP configure command to enable the appropriate DAC output, then a Data Processing ASP (DP-ASP) command to configure it to execute an averaging filter over the provided data with the DAC as an output destination, then finally an ADC-ASP configuration command to select the ADC channel, sample rate, and set the output destination to the DP-ASP. At this point the ADC-ASP will begin sending data to the DP-ASP, then the DP-ASP will perform the averaging filtering function and (when data is available) this will send the data to the DAC-ASP. Similarly, you can create different datapath that process the data stream in cascade that comprises different processing blocks (DP-ASPs). For this purpose we provide a specific initial design, called **reference design**, that integrates the nodes (ASPs) into a NoC and demonstrates it on an example similar to what described here.

7. Timeline-deliverables-assessment

Major tasks and milestones, and tentative dynamics

Week **Task/milestone**	**2**	**3**	**4**	**5**	**6**	**SB1**	**SB2**	**7**	**8**	**9**	**10**	**11**	**12**
Lab-1	Assess (10%)	Assess (10%)											
Lab-2					Rel			Assess	Assess				
IP (ASP with NI)				Rel							Canvas Submis.		
GP1 (ReCOP)		Rel			IV	Canvas Submiss							
GP2 (HMPSoC integration , IP IW)				Rel	Rel								GP2, IV. Canvas subiss.(30%) IP (IV 10%)

*SB- Study break

IV – Interview (lab time TBA)

8. Resources and tools

- Terasic DE1-SoC FPGA Development Kit for Altera/Intel Cyclone V based systems
- Intel Quartus Development Environment including Platform Designer, version 18.1
- ModelSim
- Other resources will be provided via Canvas

9. References and readings

1. Singh, A.K., Shafique, M., Kumar, A. and Henkel, J., 2013, May. Mapping on multi/many-core systems: survey of current and emerging trends. In *Proceedings of the 50th Annual Design Automation Conference* (p. 1). ACM.
2. Benini, L. and De Micheli, G. 2002. Networks on chips: a new SoC paradigm. *Computer 35*, 70-78.
3. Salcic, Z., Park, H., Biglari-Abhari, M., and Teich, J., 2019, SystemGALS – A language for the design of GALS software systems, Embedded Systems Research Group, University of Auckland, Internal document, Embedded Systems Research Group, available to C701 class
4. Malik, A., Salcic, Z., Roop, P.S. and Girault, A. 2010. SystemJ: A GALS language for system level design. Computer Languages, Systems & Structures 36, 317-344
5. Salcic, Z., Nadeem, M. and Striebing, B, 2016, A Time Predictable Heterogeneous Multicore Processor for Hard Real-time GALS Programs. *ARCS 2016*
6. Salcic Z, Park H, Teich J, Malik A, Nadeem M , 2017, NoC-HMP: A Heterogeneous Multicore Processor for Embedded Systems Designed in SystemJ, ACM Transactions on Design Automation of Embedded Systems, 2017, Volume 22 Issue 4
7. Salcic, Z., Nadeem, M., Park, H. and Teich, J, 2016, Optimizing Latencies and Customizing NoC of Time-Predictable Heterogeneous Multi-Core Processor. In *2016 IEEE 10th International Symposium on Embedded Multicore/Many-core Systems-on-Chip*, Lyon
8. Salcic, Z & Lorigan, H, 2020, ReCOP – A Processor Core for Control-dominated Reactive Applications, University of Auckland, Department of Electrical and Computer Engineering, Embedded Systems Research Group, a modified version of 2015 document, document available to the C701 class
9. Salcic, Z. and Malik, A. 2013. GALS-HMP: A heterogeneous multiprocessor for embedded applications. *ACM Transactions on Embedded Computing Systems (TECS) 12*, 58.
10. Hui, D. and Salcic, Z., 2004, MiCORE- A Customisable Microprocessor Core, University of Auckland, Department of Electrical and Computer Engineering, Embedded Systems Research Group Internal Report
11. Hui, D. and Salcic, Z., 2004, ReMIC- A Customisable Reactive Microprocessor Core, University of Auckland, Department of Electrical and Computer Engineering, Embedded Systems Research Group Internal Report