



Herbert Wertheim
College of Engineering
UNIVERSITY *of* FLORIDA

POWERING THE NEW ENGINEER TO TRANSFORM THE FUTURE

Department of Electrical & Computer Engineering

MeXT: A Flow for Multiprocessor Exploration

Dr. Christophe Bobda, Harold Ishebabi, Philipp Mahr, Joel Mandebi Mbongue and Sujan Kumar Saha

Agenda

- Motivation
- The MeXT Design Flow
- MPI for Embedded Multiprocessor Systems
- Case Study
- Summary
- Future Work: Security and Reliability Extension

Motivation

- Increasing high computation demand
- Increasing Complexity due to multiple heterogeneous components in a system
- Heterogeneous systems are not optimized always for a set of applications
- An optimal communication among processing components is missing
- Parallelization required for some applications for performance increase

Motivation

- Approach:
 - Define a quality/tolerance factor for each component/task in the system
 - Reformulate original optimization problem
 - Provide an architecture that will enforce a high-level of reliability with minimum redundancy
 - Provide methods to optimize architectural resources at run-time

Motivation

- Approach – Problem Reformulation
 - Many problems are defined as on-line optimization problem, with the goal of producing a optimal output under environmental constraints
 - Reformulate the problem as the production of the closest output under the same environmental constraints.
 - Use current solver to devise an online solution
 - However, provide a combined time-space redundancy that will improve the quality of results and keep the output close to the optimal result
 - Use of reconfiguration logic, with extreme flexibility an performance to organize this work.

- Embedded Optimization to enforce rules at run-time

Target System

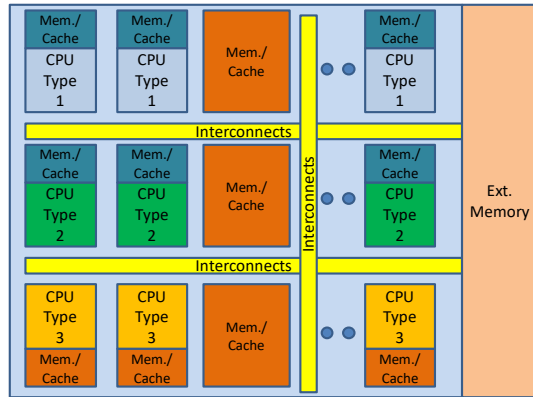


Figure 1: Target system architecture

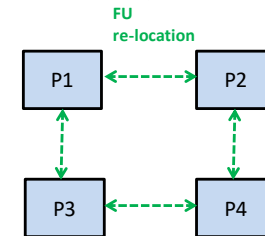
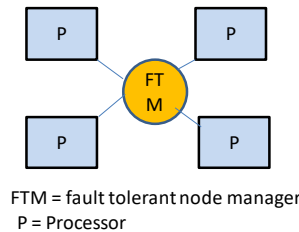
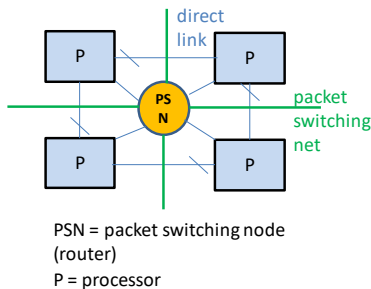


Figure 3: Communication Topologies

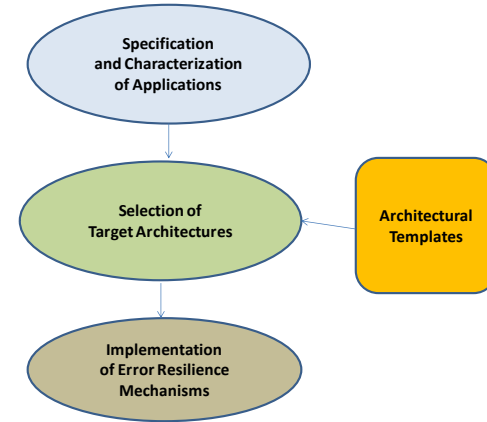


Figure 2: Overall Design Flow

Target System

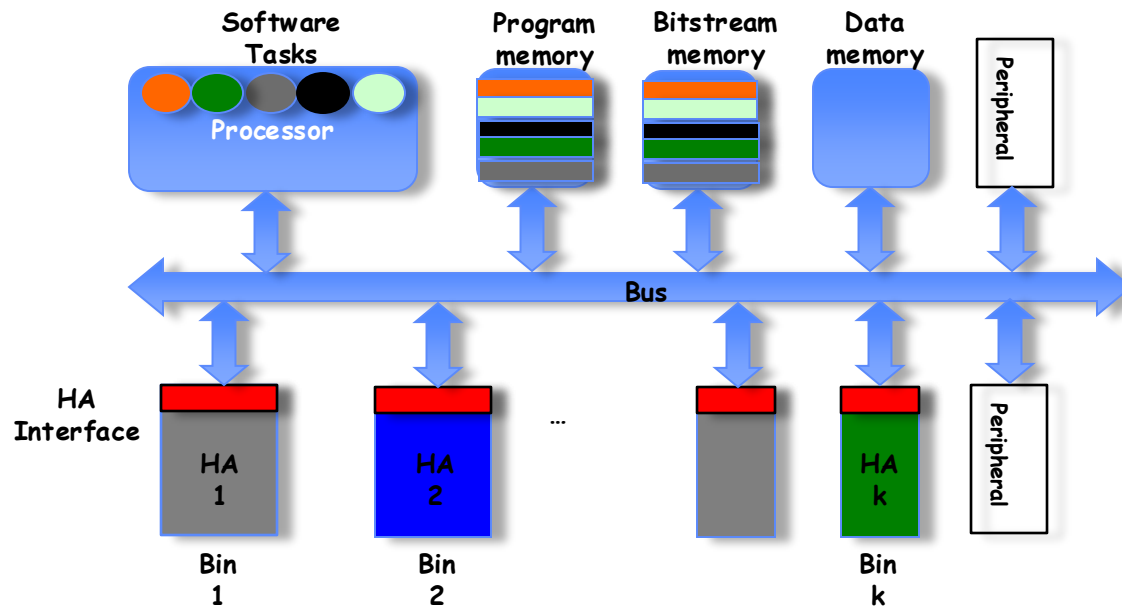


Figure 4: SoC Architecture with Hardware Accelerators

MPSoC Exploration Design Flow

Input: Parallel programs and sequential programs, real-time constraints, platform components

- Intermediate representation of programs for performance profiling
- Cost optimization and architecture exploration
- Transforming of an Abstract Specification into a Platform-dependent *Concrete Specification* (e.g.: Platform \rightarrow Xilinx ML310, CPU \rightarrow PowerPC, Memory \rightarrow BRAM, CommMedium \rightarrow PLB,...)
- Generation of the platform-dependent hardware description files (Concrete Component Description)

Output: FPGA configuration file

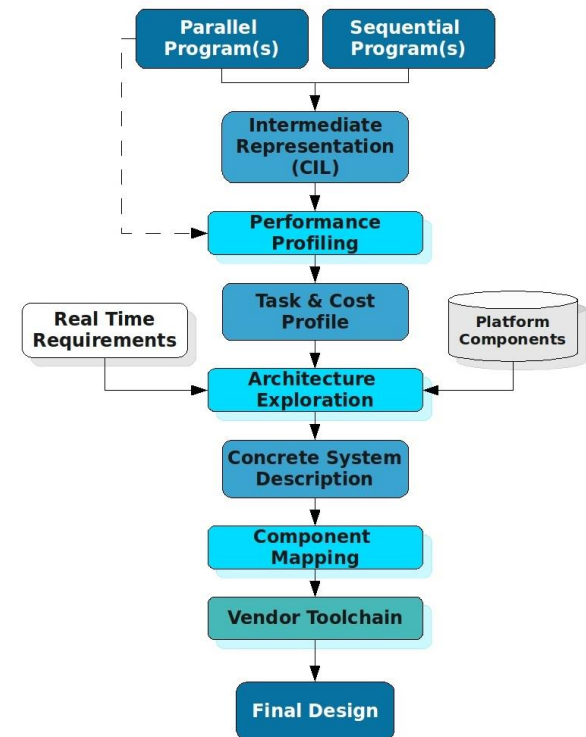


Figure 5: MeXT Design Flow

- The *Concrete Specification* can also be the result of the architectural synthesis

Semi-Automatic System Generation

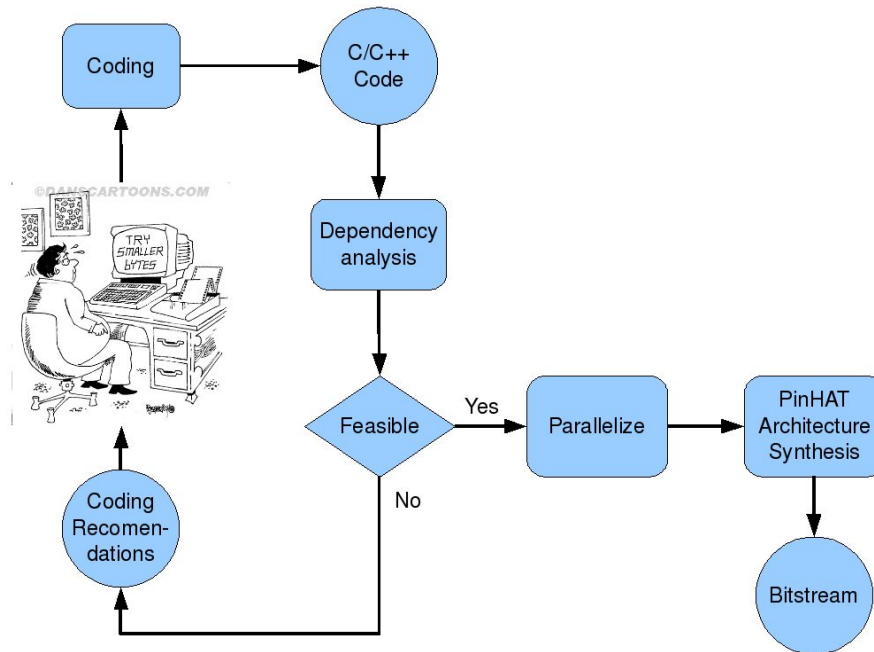


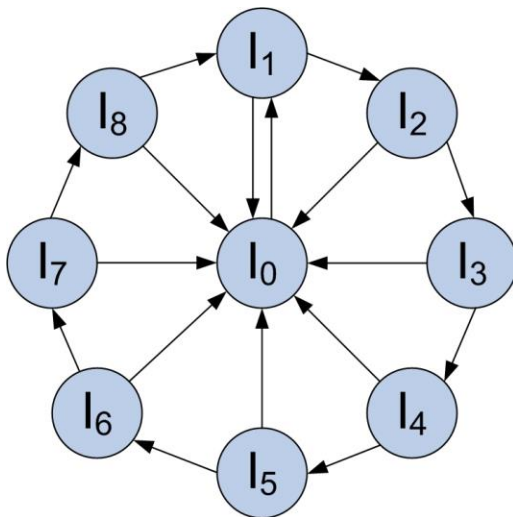
Figure 6: A Semi-Automatic System Generation Design Flow

Architecture Synthesis - Scope

- Modeling goal:
 - Allow selection of the best configuration (application-specific)
 - ➔ Enable (runtime) optimization
 - Lead to (automatic) generation of HW infrastructure
 - ➔ Cost models of HW components
 - ➔ System model/description (PEs, networks, mapping, ...)
 - ➔ SW configuration
 - ➔ Synthesis and device configuration
- Approach:
 - Synthesize an application-specific optimum system from parallel programs (high-level synthesis) using ILP

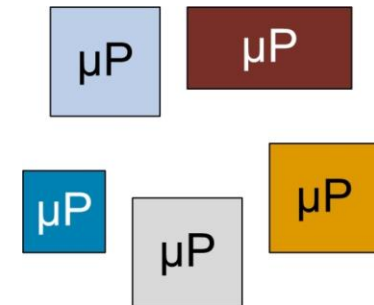
Architecture Synthesis - Problems

Parallel Program:

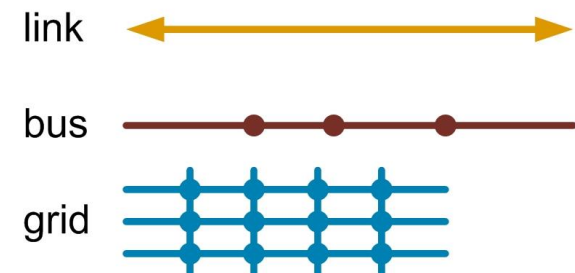


Problem: Simultaneous Task-Mapping and Selection of Resources

Processors:



Comm. Networks:



MPSoC Design Objectives

- Construct an implementation with the desired functionality
- More challenging: simultaneously optimize numerous design metrics
- Metrics
 - Unit cost, NRE
 - Size, Power, a Weight
 - Performance
 - Flexibility, maintainability
 - Time-to-market, time-to-prototype

ILP Model

Map all tasks

$$\sum_{j=0}^m x_{ij} = 1, \forall I_i$$

Max # tasks per PE

$$\sum_{i=0}^n x_{ij} \cdot s_{ij} \leq s_j, \forall I_i$$

Max. usable FPGA area for PEs

$$v_j \leq \sum_{i=0}^n x_{ij}, \forall J_j$$

$$A_{PE} \geq \sum_{j=0}^m v_{ij} \cdot a_j$$

Switching time for all tasks

$$T_{SWITCH} = \sum_{j=0}^m \sum_{i=0}^n x_{ij} \cdot t_j$$

ILP Model

Maximum Number of processes using a communication topology

$$\lambda_{i_1, i_2} = \frac{x_{i_1 j_1} + x_{i_2 j_2}}{2} \quad y_k + \sum_{I_{i_1}, I_{i_2} | I_{i_1} \leq I_{i_2}} \lambda_{i_1, i_2} \leq M_k, \forall C_k$$

Area Cost of Communication Network

$$A_{NET} \geq \sum_{j=0}^m A_k \cdot y_k$$

Total Area Cost

$$A \geq A_{PE} + A_{NET}$$

ILP Model

Computation Time Cost for network topology

$$z_{ki_1i_2} \geq \lambda_{i_1,i_2} + y_k - 1$$

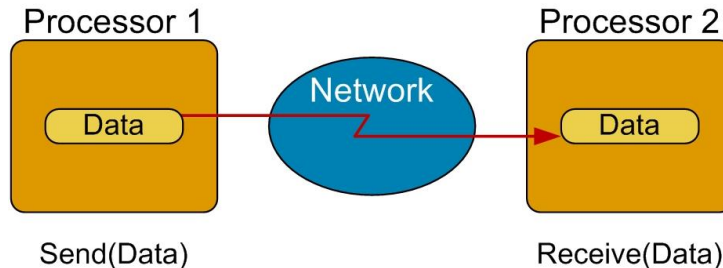
$$T_{NET} = \sum_{I_{i_1}, I_{i_2} | I_{i_1} < I_{i_2}} \left(\sum_{k=0}^K (D_{i_1i_2} + \tau_k \cdot p_k) z_{ki_1i_2} \right)$$

Total Computation Time Cost

$$T = \min \left(\sum_{j=0}^m \sum_{i=0}^n x_{ij} \cdot T_{ij} + T_{NET} + T_{SWITCH} \right)$$

SocMPI

- Embedded Systems lack an efficient Message Passing Interface (MPI) communication library



- SocMPI - on-Chip Communication library
- Compatible with a subset of the MPI Standard
- Hardware supported MPI functions (BCast, WTime)
- Configurable (MPI functions and networks)
- Memory requirements: 11 - 16 KByte

SocMPI

- Two Layers: network dependent and network independent
- Currently supported : Direct Link, Star, Bus and Ring topology
- Easily extendable to other networks

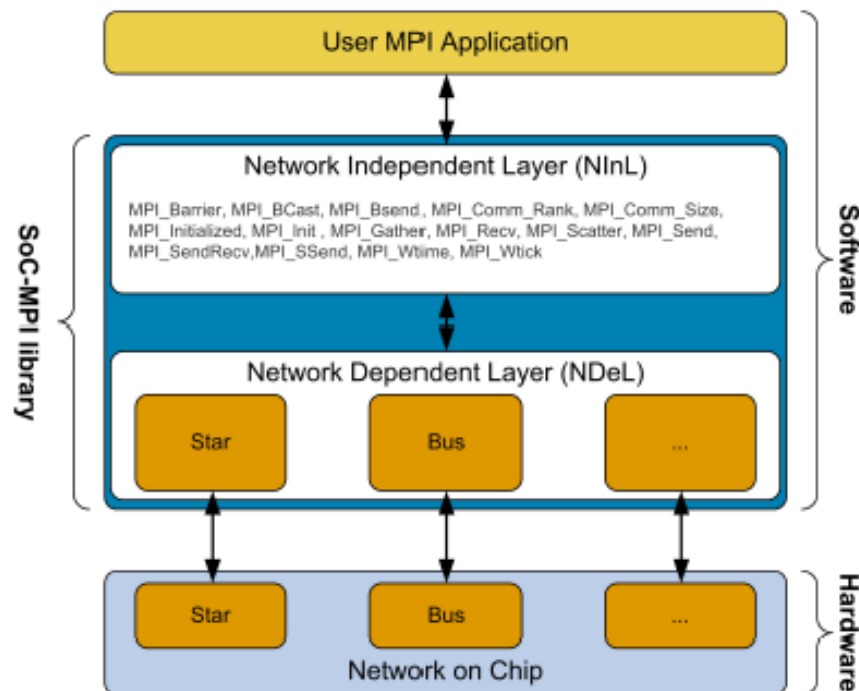


Figure 7: Layers of SoC-MPI

Case Study 1: Mandelbrot Fractal

- Parallel implementation of Mandelbrot is used to analyze for generating profile.
- Three MPI tasks were allowed
- FSL direct-link, PLB bus and FSL-based bus used for exploration
- Simplified architecture is suggested in figure
- Table I shows parameter for available networks

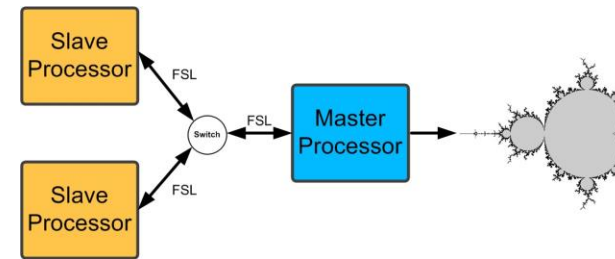


Figure 8: Simplified System

TABLE I
PARAMETERS OF THE AVAILABLE NETWORKS

Name	Capacity	Area	Transfer Latency [μ s]	Arbitration Probability	Probability Bound	Type
PLB	8	242	0.04	0.08	0.000016	bus
Star	8	213	0.04	0	0	star
FSL	1	21	0.02	0	0	link

Case Study 2: WLAN 802.11g

- Parallel implementation of signal processing chain for IEEE 802.11g WLAN is used to analyze for generating profile.
- DAG shows communication pattern, task deadline and execution time
- Optimization is performed using MeXT and scheduling is analyzed
- Figure shows the resulting system architecture

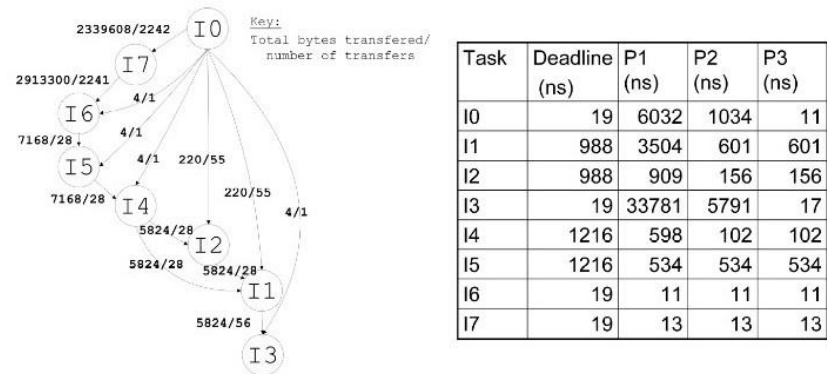


Figure 9: DAG of the program and task deadlines

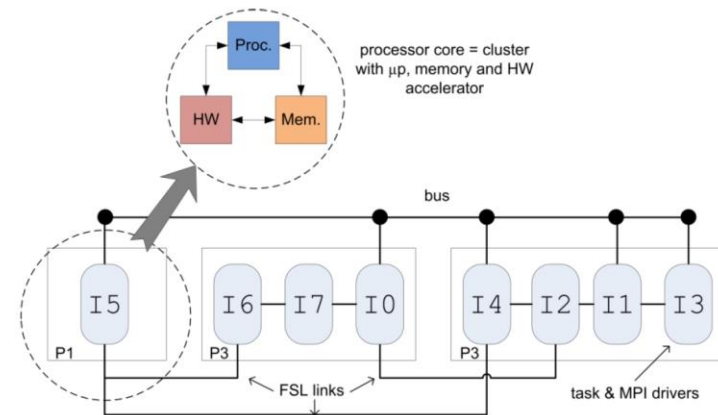


Figure 10: Architecture for WLAN program

Summary

- Design Systems Able to Cope with Probabilistic Behavior
- Framework for MPSoC Design Exploration
- Optimization for suitable Design
- Message Passing protocol for SoC-MPI

Future Work: Security and Resiliency Integration

- Secured Execution of Hardware/Software Threads in System on Chips
- Extending separation kernel policies within hardware components, and
- Enforcing corresponding access decision rules directly in hardware

Future Work: The Flask Security Architecture Model

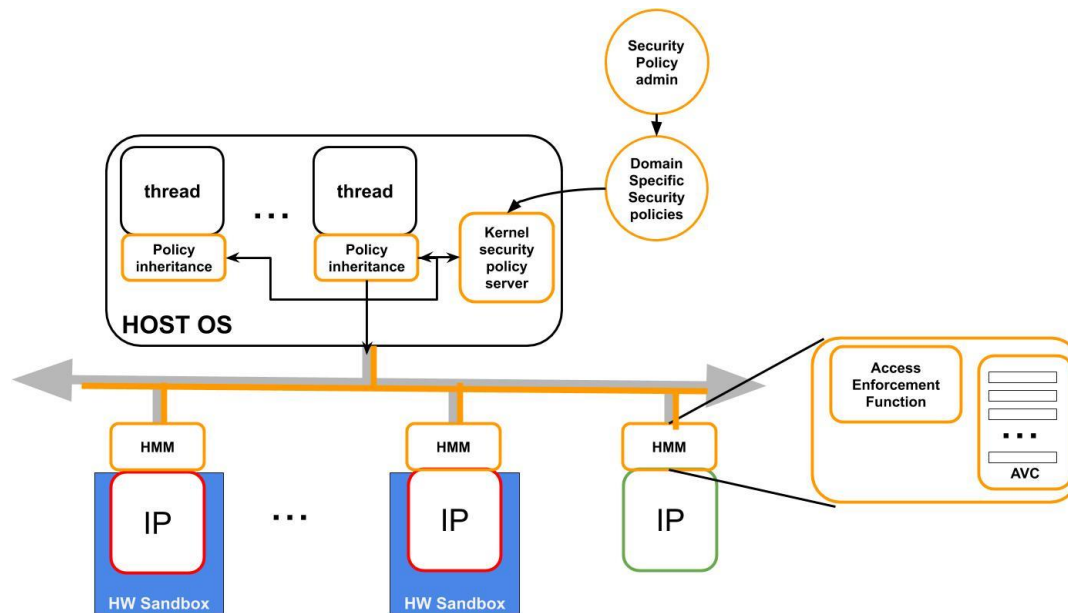
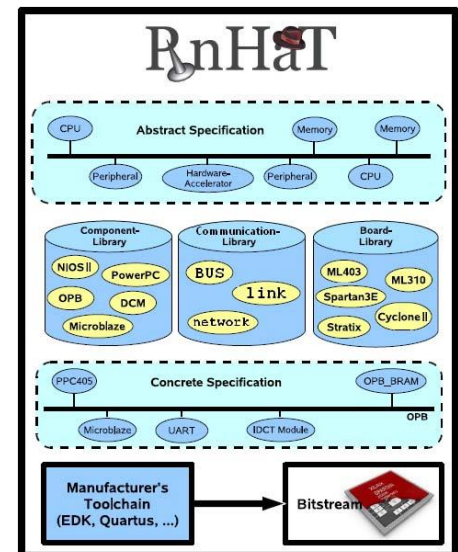
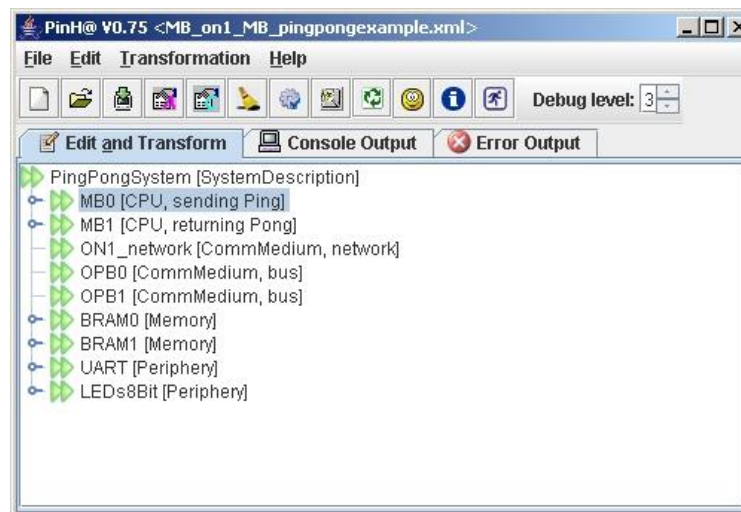
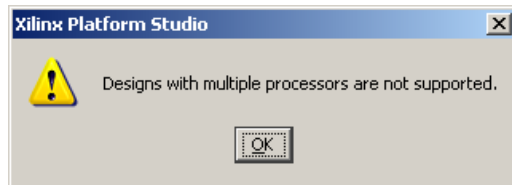


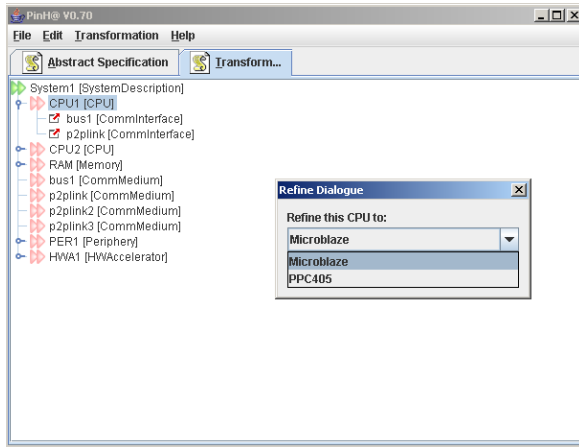
Figure 11: Flask Security Architecture with Hardware Management Module (HMM)

PinHaT (Platform-independent Hardware Generation Tool)

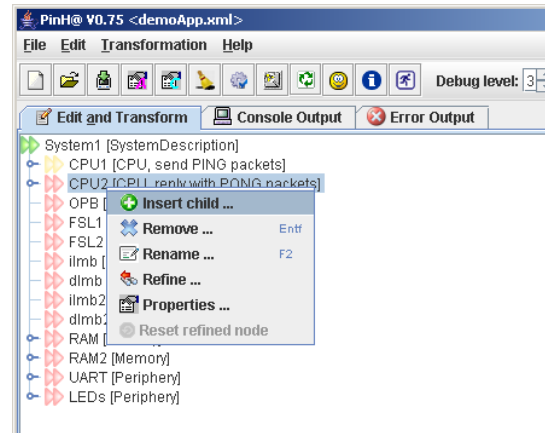
- Funded By The German Research Foundation (DFG) 2005 - 2010
 - Simplify the Design of MPSoCs
 - Vendor independent framework based on Java and XML
 - Abstract System Specification → Platform-dependent description files



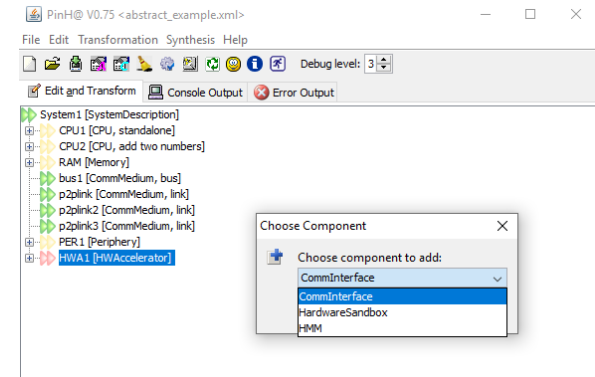
Security and Resiliency Integration



Configure Component



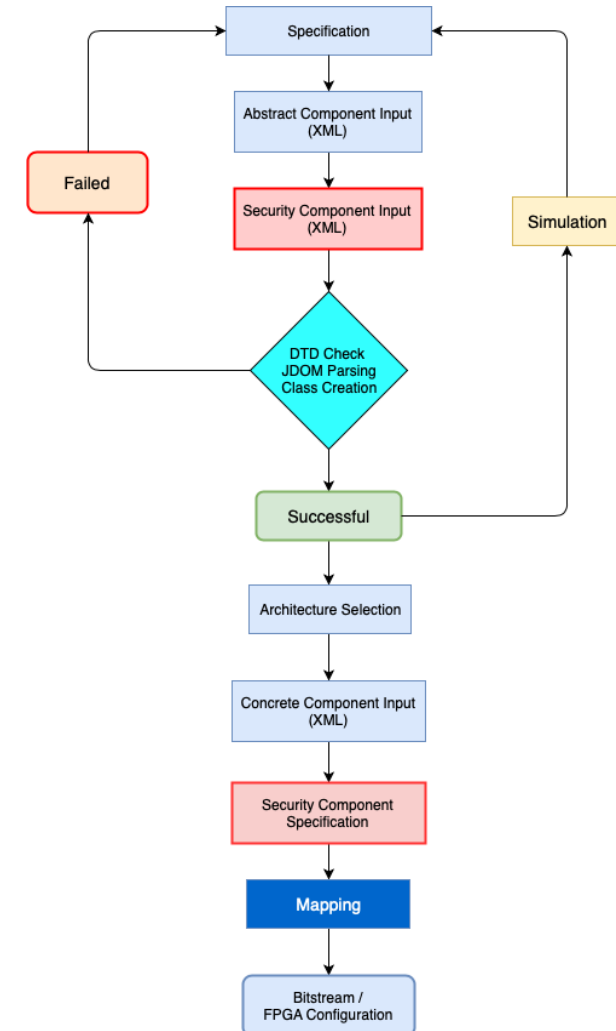
Interface Integration

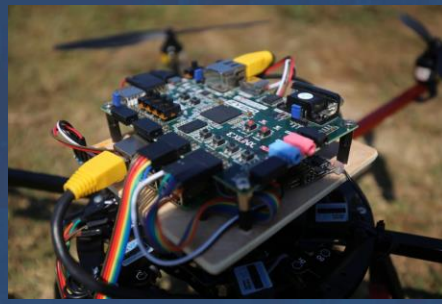


Security/Reliability Extension

Security and Resiliency Integration

- The following extension has been added with PinHat
- Abstract security components (Hardware Sandbox and HMM) can be added now with existing abstract component input.
- While adding concrete component input, security components specification (e.g. size of HMM Access Vector Cache, I/O etc.) can be added.
- The communication interface between security components and bus can be specified





UF | Herbert Wertheim
College of Engineering
UNIVERSITY *of* FLORIDA

smartsystems.ece.ufl.edu