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

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

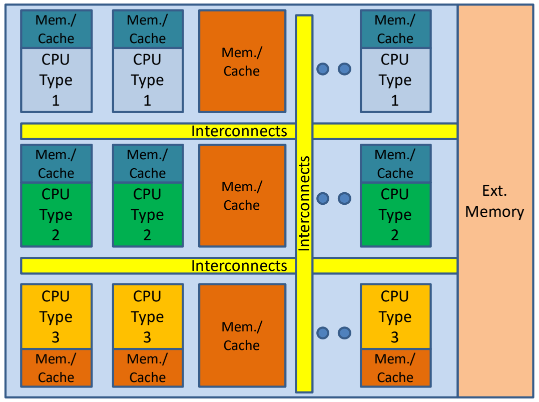


Figure 1: Target system architecture

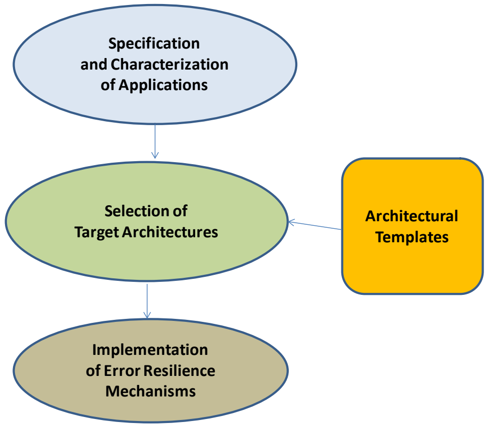
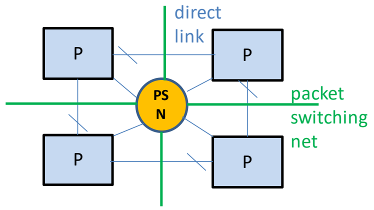
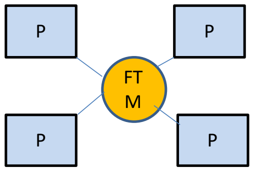

Specification and Characterization of Applications
Selection of Target Architectures
Architectural Templates
Implementation of Error Resilience Mechanisms

Figure 2: Overall Design Flow



PSN = packet switching node (router)
P = processor



FTM = fault tolerant node manager
P = Processor

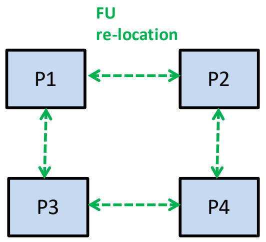


Figure 3: Communication Topologies

Target System

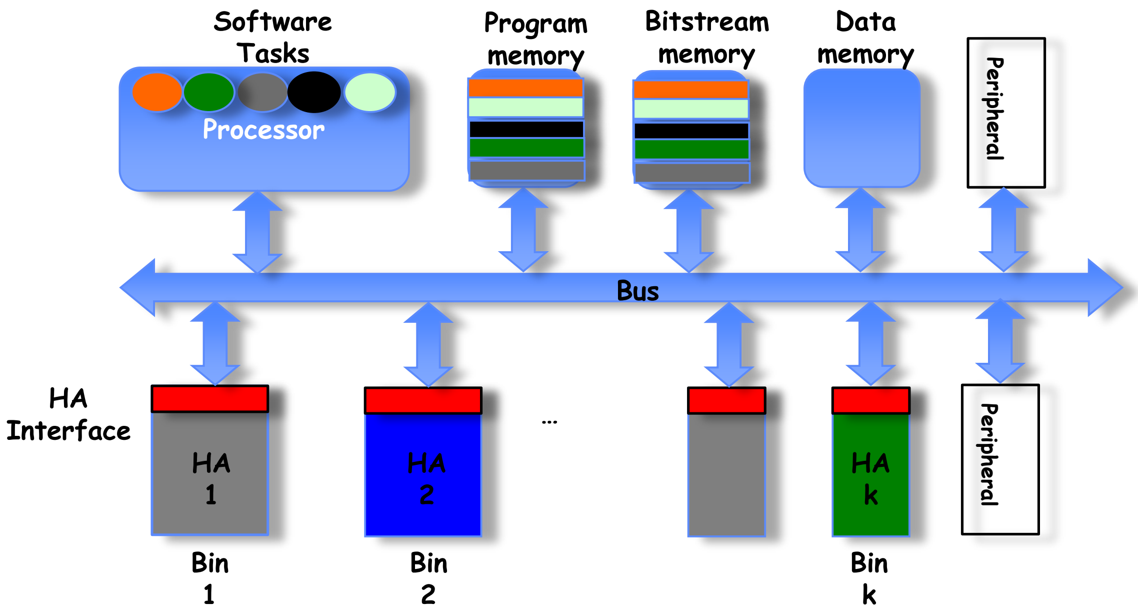


Figure 4: SoC Architecture with Hardware Accelerators

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* ⍰ Xilinx ML310, CPU ⍰ PowerPC, Memory ⍰ BRAM, CommMedium ⍰ PLB,...)
- Generation of the platform-dependent hardware description files (Concrete Component Description)

Output: FPGA configuration file

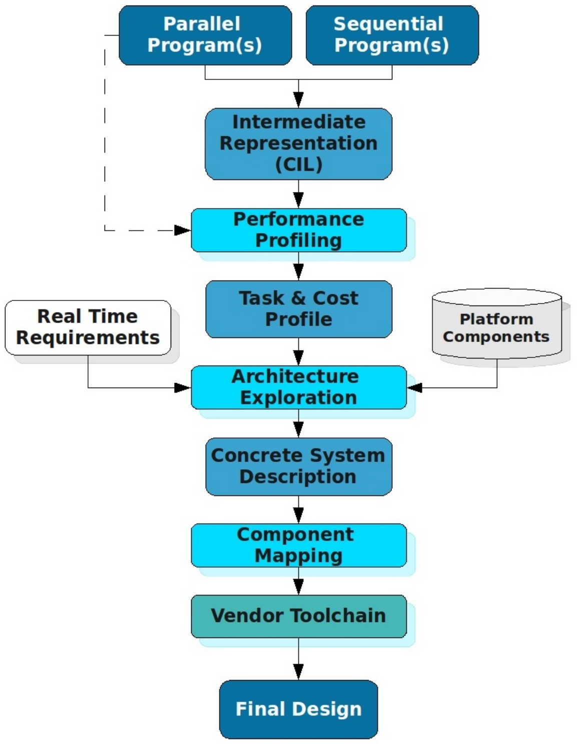


Figure 5: MeXT Design Flow

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

Semi-Automatic System Generation

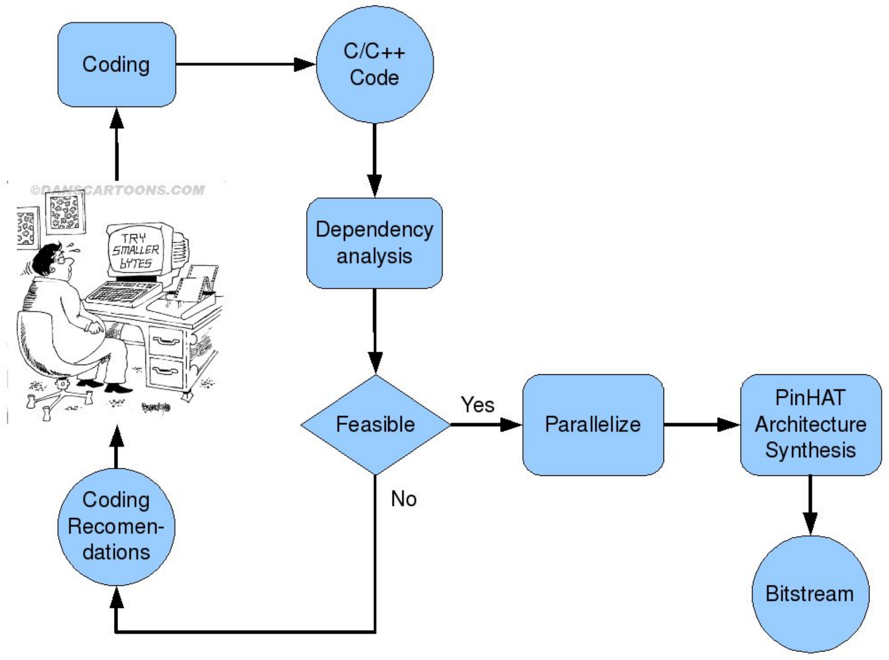


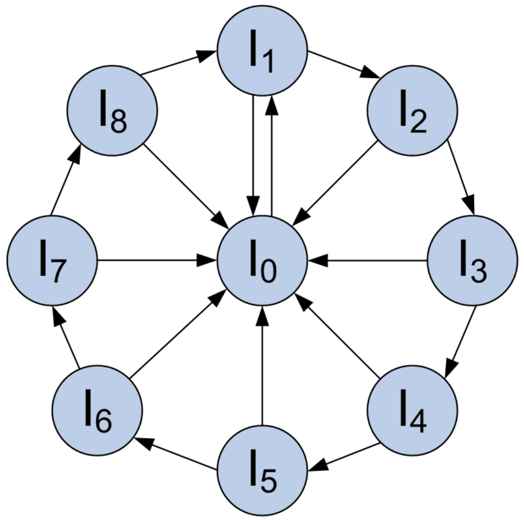
Figure 6: A Semi-Automatic System Generation Design Flow

Architecture Synthesis - Scope

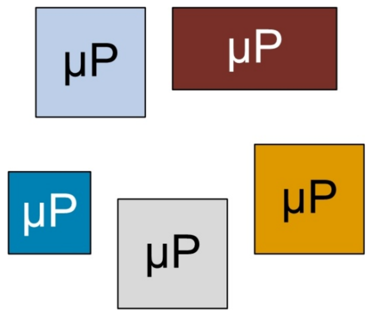
- Modeling goal:
  - Allow selection of the best configuration (application-specific)
    ➔ Enable (runtime) optimization
  - Lead to (automatic) generation of HW infrastructure
    ➔ Cost models of HW components
    ➔ System model/description (PEs, networks, mapping, …)
    ➔ SW configuration
    ➔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

Architecture Synthesis - Problems

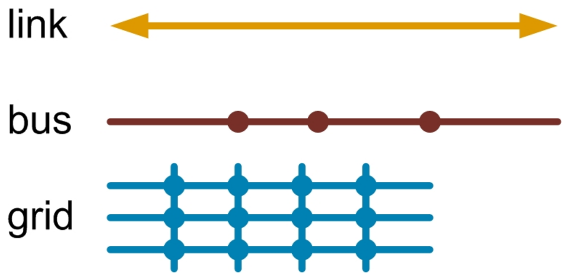
Parallel Program:



Processors:



Comm. Networks:



Problem: Simultaneous Task-Mapping and Selection of Resources

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

$$\lambda_{i1,i2} = \frac{x_{i_1 j_1} + x_{i_2 j_2}}{2} \qquad y_k + \sum_{I_{i_1}, I_{i_2} | I_{i_1} < I_{i_2}} \lambda_{i1,i2} \leq M_k, \forall C_k$$

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

$$z_{ki_1i_2} \geq \lambda_{i1,i2} + y_k - 1$$

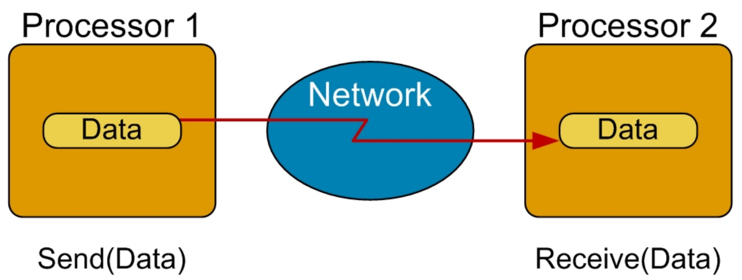
$$T_{NET} = \sum_{I_{i_1}, I_{i_2} | I_{i_1} < I_{i_2}} \left(\sum_{k=0}^{K} (D_{i_1i_2} + \tau_k \cdot p_k) z_{ki_1i_2} \right)$$

Total Computation Time Cost

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

Soc*MPI*

- 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

Soc*MPI*

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

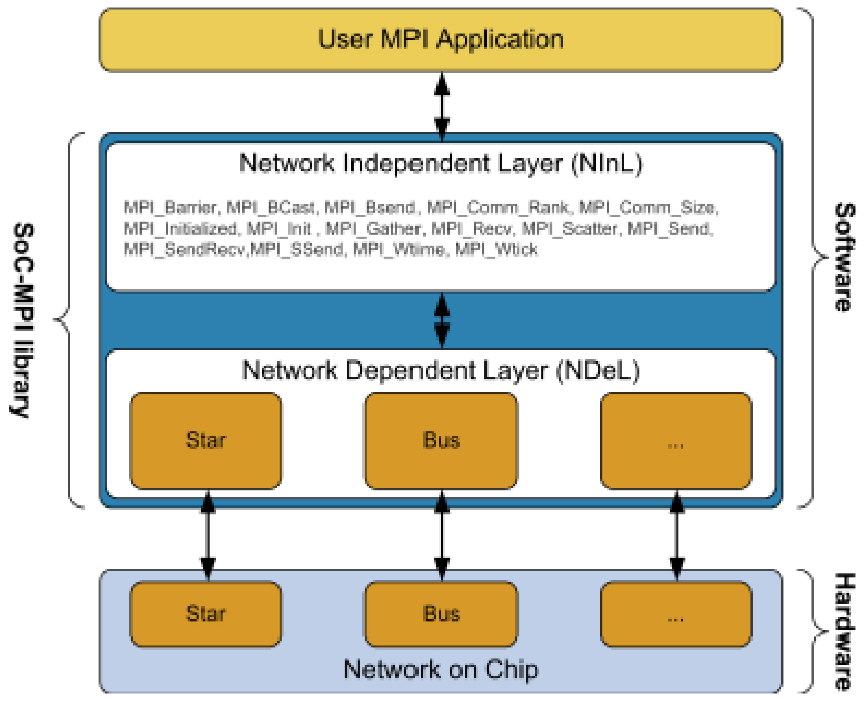

User MPI Application
Network Independent Layer (NInL)
MPI_Barrier, MPI_BCast, MPI_Bsend, MPI_Comm_Rank, MPI_Comm_Size, MPI_Initialized, MPI_Init, MPI_Gather, MPI_Recv, MPI_Scatter, MPI_Send, MPI_SendRecv, MPI_SSend, MPI_Wtime, MPI_Wtick
Network Dependent Layer (NDeL)
Star
Bus
...
Star
Bus
...
Network on Chip
SoC-MPI library
Software
Hardware

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

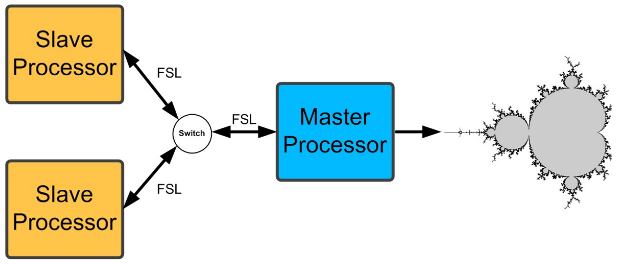


Figure 8: Simplified System

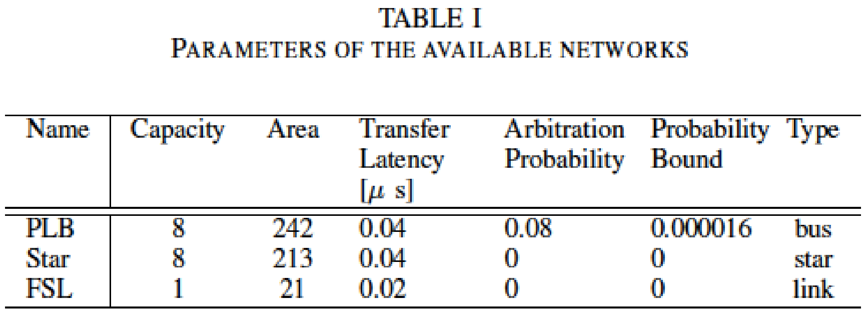
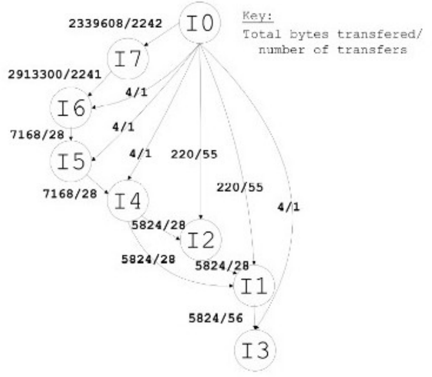


TABLE I
PARAMETERS OF THE AVAILABLE NETWORKS

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



Task	Deadline (ns)	P1 (ns)	P2 (ns)	P3 (ns)
I0	19	6032	1034	11
I1	988	3504	601	601
I2	988	909	156	156
I3	19	33781	5791	17
I4	1216	598	102	102
I5	1216	534	534	534
I6	19	11	11	11
I7	19	13	13	13

Figure 9: DAG of the program and task deadlines

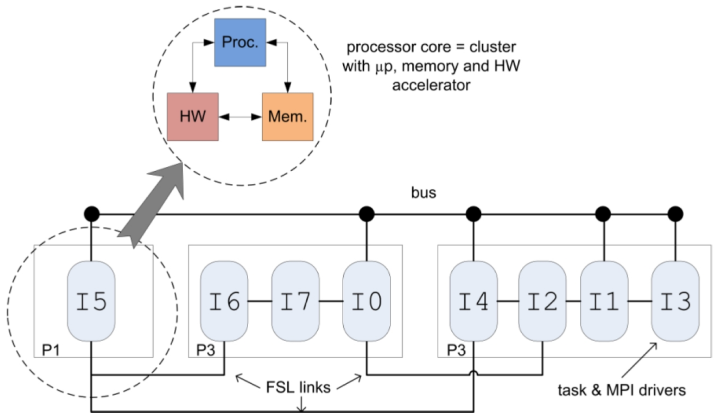


Figure 10: Architecture for WLAN program

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

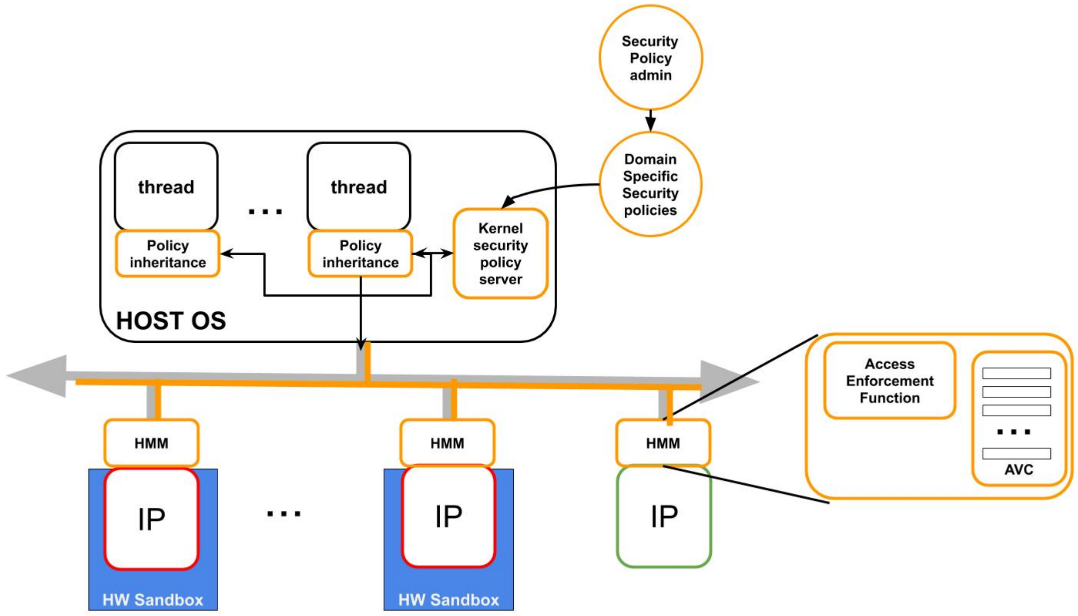
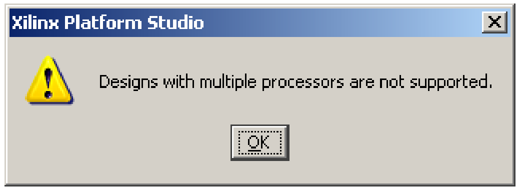


Figure 11: Flask Security Architecture with Hardware Management Module (HMM)

PinHaT (Plattform-independent Hardware Generation Tool)

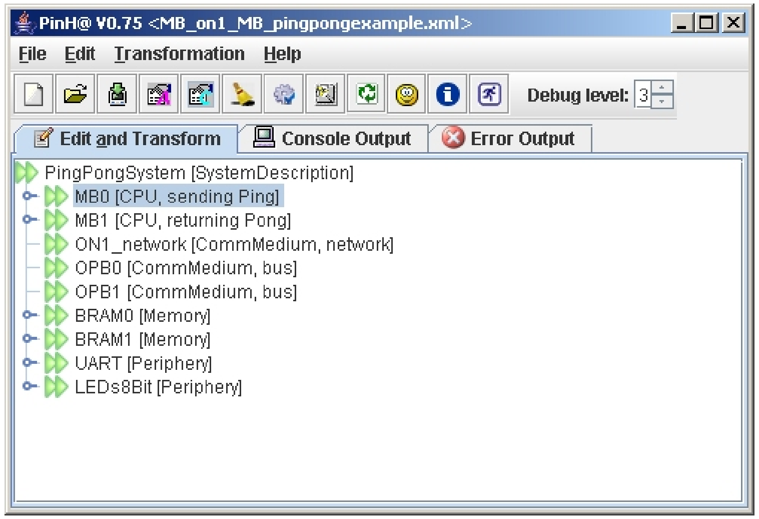
- Funded By The German Research Foundation (DFG) 2005 - 2010
  - Simplify the Design of MPSoCs
  - Vendor independent framework based on Java and XML
  - Abstract System Specification → Platform-dependent description files



Xilinx Platform Studio

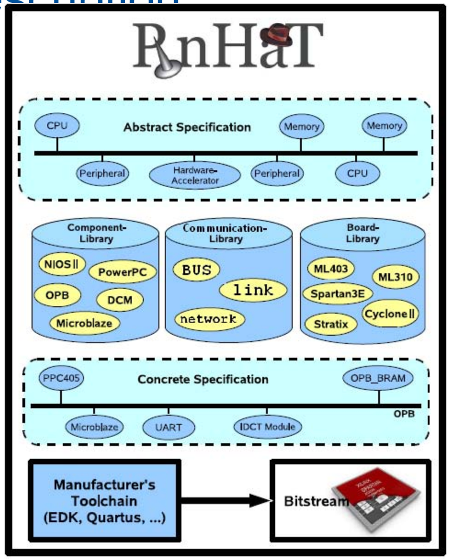
Designs with multiple processors are not supported.

OK

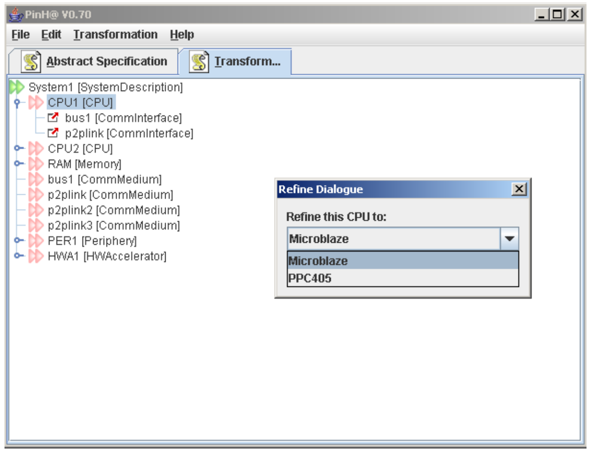


PinH@ V0.75 <MB_on1_MB_pingpongexample.xml>

```
PingPongSystem [SystemDescription]
  MB0 [CPU, sending Ping]
  MB1 [CPU, returning Pong]
  ON1_network [CommMedium, network]
  OPB0 [CommMedium, bus]
  OPB1 [CommMedium, bus]
  BRAM0 [Memory]
  BRAM1 [Memory]
  UART [Periphery]
  LEDs8Bit [Periphery]
```

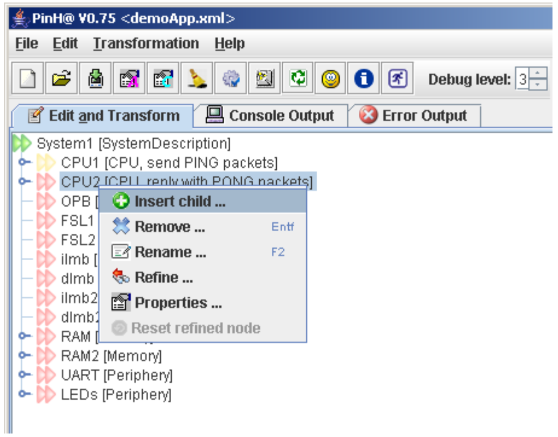


Security and Resiliency Integration



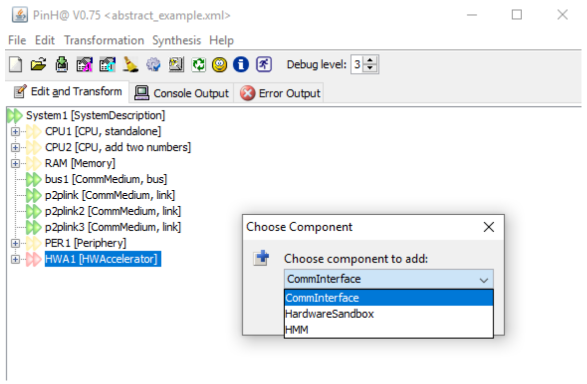
```
PinH@ V0.70
File  Edit  Transformation  Help
Abstract Specification | Transform...

System1 [SystemDescription]
  CPU1 [CPU]
    bus1 [CommInterface]
    p2plink [CommInterface]
  CPU2 [CPU]
  RAM [Memory]
  bus1 [CommMedium]
  p2plink [CommMedium]
  p2plink2 [CommMedium]
  p2plink3 [CommMedium]
  PER1 [Periphery]
  HWA1 [HWAccelerator]

Refine Dialogue
Refine this CPU to:
Microblaze
  Microblaze
  PPC405
```

Configure Component



```
PinH@ V0.75 <demoApp.xml>
File  Edit  Transformation  Help
Debug level: 3
Edit and Transform | Console Output | Error Output

System1 [SystemDescription]
  CPU1 [CPU, send PING packets]
  CPU2 [CPU, reply with PONG packets]
  OPB [
  FSL1
  FSL2
  ilmb [
  dlmb
  ilmb2
  dlmb2
  RAM [
  RAM2 [Memory]
  UART [Periphery]
  LEDs [Periphery]

Insert child ...
Remove ...          Entf
Rename ...          F2
Refine ...
Properties ...
Reset refined node
```

Interface Integration

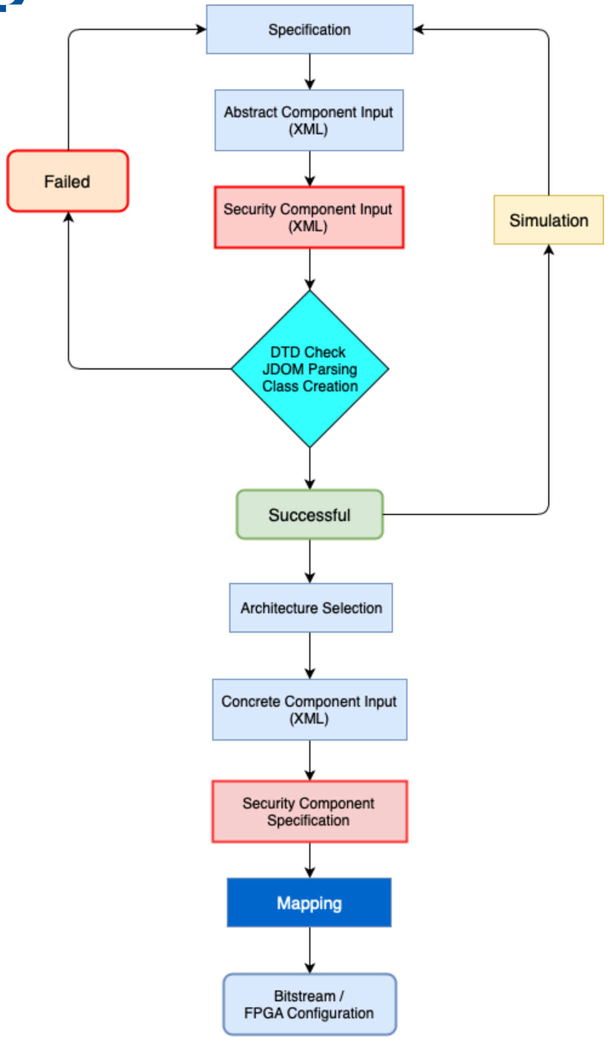


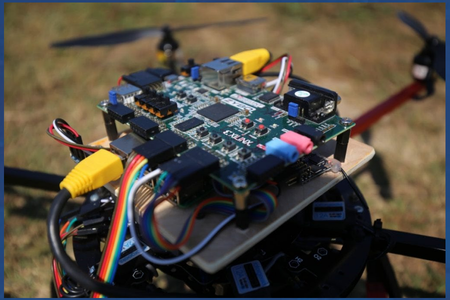
```
PinH@ V0.75 <abstract_example.xml>
File  Edit  Transformation  Synthesis  Help
Debug level: 3
Edit and Transform | Console Output | Error Output

System1 [SystemDescription]
  CPU1 [CPU, standalone]
  CPU2 [CPU, add two numbers]
  RAM [Memory]
  bus1 [CommMedium, bus]
  p2plink [CommMedium, link]
  p2plink2 [CommMedium, link]
  p2plink3 [CommMedium, link]
  PER1 [Periphery]
  HWA1 [HWAccelerator]

Choose Component
Choose component to add:
CommInterface
  CommInterface
  HardwareSandbox
  HMM
```

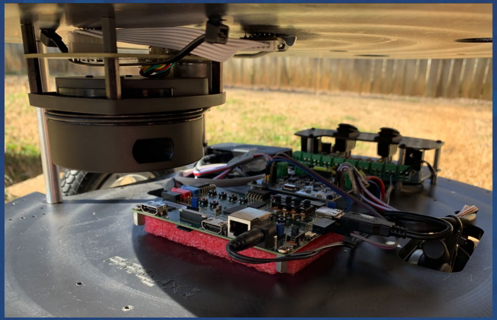
Security/Reliability Extension

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


Specification
Abstract Component Input (XML)
Failed
Security Component Input (XML)
Simulation
DTD Check
JDOM Parsing
Class Creation
Successful
Architecture Selection
Concrete Component Input (XML)
Security Component Specification
Mapping
Bitstream / FPGA Configuration





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

smartsystems.ece.ufl.edu

POWERING THE NEW ENGINEER TO TRANSFORM THE FUTURE