A Methodology for the Design and Deployment of Reliable Systems on Heterogeneous Platforms

Hugo A. Andrade†, Arkadeb Ghosal*, Kaushik Ravindran* and Brian L. Evans†*

*National Instruments Corporation, Berkeley, CA 94704, USA, †Dept. of Electrical and Computer Engineering, The University of Texas at Austin, Austin, TX 78712, USA

E-mail: {hugo.andrade, arkadeb.ghosal, kaushik.ravindran}@ni.com, bevans@ece.utexas.edu

Motivation

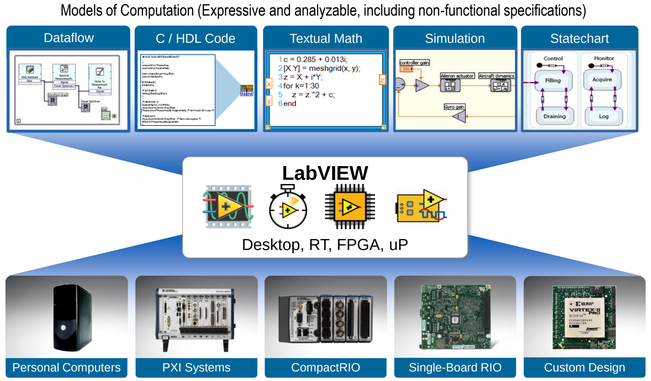
- *Heterogeneous programmable multi-target platforms are an established trend in mainstream and embedded computing*
  - Processors, FPGAs, GPUs
  - Application specific processing units
  - Distributed memories, and specialized interconnection networks
  - Intelligent I/O
  - Global Timing and GALS
- *Reasons*
  - Continuous increase in performance requirements of embedded applications
  - Need to adapt products to rapid market changes has made programmability a key criterion
  - Dominant platforms in a variety of markets including digital signal processing, communications, networking, graphics, and gaming.
  - Recent surge of applications in medicine, finance, and security
- *Challenge: Reliability an increasing design consideration*
  - Embedded systems need to ensure high reliability/availability in safety critical environments
  - Systems are often large and complex and downtime is very expensive
  - New technologies are more susceptible to hard errors due to external effects like radiation and deterioration over time.
  - Due to the harsh physical conditions in which they are deployed, systems are more susceptible to catastrophic (physical) failures or malicious attacks.
  - The complexity of the design solution makes them increasingly prone to manufacturing (lower yield) errors, design errors, and run-time (soft) errors.
  - The typical platform is therefore a collection of unreliable computing devices, communicating over an unreliable network, and surrounded by I/O that may be unreliable as well.
  - The application components deployed on this platform may also be subject to design and run time errors, or even contain maliciously created programs.
- *Opportunity*
  - The larger number and relative independence of multiple processing components motivates a design approach that exploits redundancy to improve reliability and availability.
  - The system can be viewed as a collection of possibly redundant computation, communication, and I/O components, which can be replaced in the field while the system is in operation.
  - The redundancy of these field replaceable units (FRUs) are exploited to maintain the mean time between failure (MTBF) within reasonable boundaries and create a dynamic fault-tolerant system.
  - FRUs are typically low cost parts, composed of few components, as opposed to an entire chassis, which helps minimize the cost of repair and replacement.

Approach

- *Requirement to build a reliable, efficient, and cost-effective system of FRUs*
  - The application specification must be extended to include non-functional requirements related to reliability and availability.
  - The platform model should capture the reliability of individual components and define how the reliability of the combined system is evaluated.
  - In most embedded applications, the nature and affinity of I/O is an important consideration.
  - FRUs usually have direct interaction with the I/O of the system. The processing components of these FRUs are complemented by reconfigurable devices that serve as interfaces to the I/O and guarantee control and timing requirements.
  - This introduces additional constraints that regulate how these FRUs can be reconfigured and computation can be re-distributed to exploit redundancy.
  - The challenge then is to develop models, tools, and run-time environments for the design and deployment of reliable fault-tolerant systems on heterogeneous multi-target platforms.

- *Design Methodology & Deployment/Run-time Environment*
  - Elevate reliability concerns to the system level, and propose a framework that enables reliable system level design by application domain experts.
    - Specification of non-functional requirements in terms of intuitive and well-defined models of computation and application design patterns
    - Modeling (reconfigurable) platform elements that can be automatically composed into systems to provide a reliable architecture for deployment
    - Segmenting (in space and time) the deployment and run-time environment such that the system captures independent end-user provided reliability criteria.
    - Hardware-software co-design environment that uses dynamic mapping to adapt to different operational scenarios.

Platform-based Design and MoCs

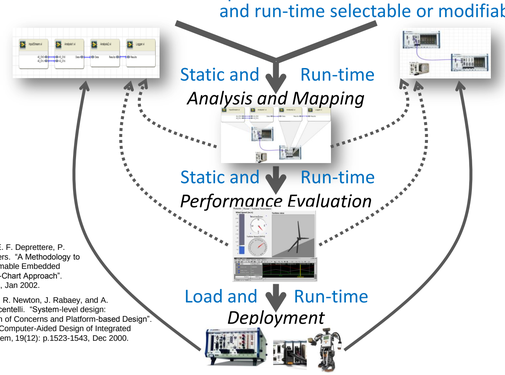


Ref: Models of Computations,. Prof. Edward Lee, Platform Based Design, Prof. Alberto Sangiovanni-Vincentelli

Y-Chart Disciplined System Design Methodology with non-functional Requirements

Application Model (and Constraints)
...with functional and non-functional requirements and run-time selectable or modifiable behaviors

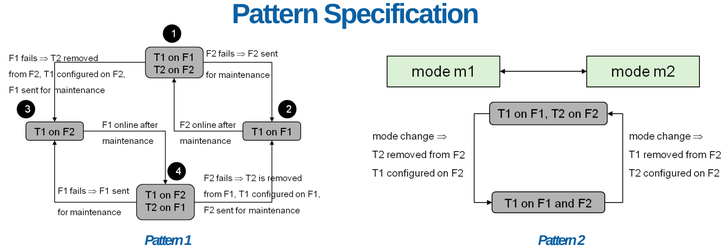
Platform Model (and Constraints)
... with reliability metrics



[1] B. Kienhuis, E. F. Deprettere, P. Wolf, K. A. Vissers. "A Methodology to Design Programmable Embedded Systems - The Y-Chart Approach". SAMOS, p.18-37, Jan 2002.

[2] K. Keutzer, A. R. Newton, J. Rabaey, and A. Sangiovanni-Vincentelli. "System-level design: Orthogonalization of Concerns and Platform-based Design". IEEE Trans. on Computer-Aided Design of Integrated Circuits and System, 19(12): p.1523-1543, Dec 2000.

Pattern Specification



Pattern 1 *Pattern 2*

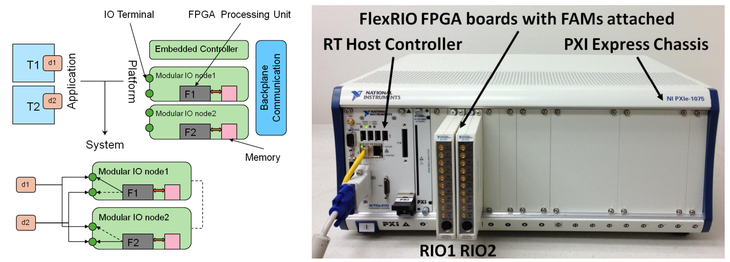
Pattern 1: Increasing Availability without Explicit Backup Redundancy

- Reconfigurability is used to meet availability requirements. Two tasks T1 & T2 are running on two reconfigurable processors F1 & F2, respectively. T1 has higher priority than T2 in terms of availability, and requirement for the design states that if at least one of the processors is operational, then T1 must be executing.

Patter 2: On-demand Reliability based on Modal Requirements

- Two tasks (T1, T2) and two reconfigurable processors (F1, F2). There are two operation modes m1 and m2. In mode m1, tasks T1 and T2 execute on F1 and F2, respectively. In mode m2,task T2 is removed and task T1 is replicated on F1 and F2 such that T1 runs in DMR.

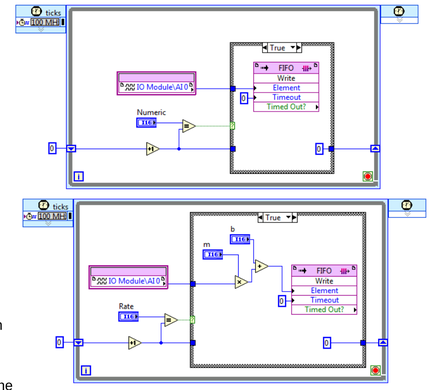
Case Study



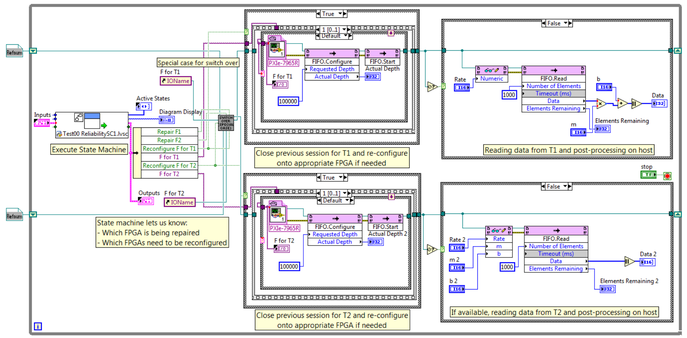
Design Flow

PXIe-based Platform

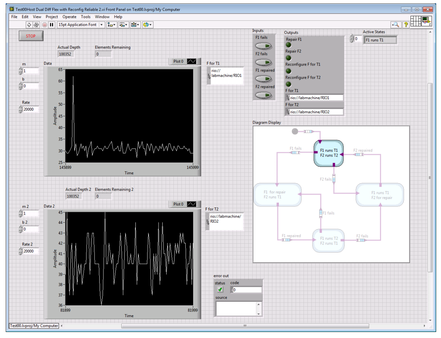
- *Prototyped parts of run-time infrastructure for design flow above*
  - For the experiments, we use elements currently available with the National Instruments PXI Express (PXIe) platform
  - With the proposed methodology, we are exploring a finer granularity for the FRUs compared to the current chassis level, and focusing on the reliability closer to the I/O ports of the system, so that we would need to only replace individual boards. We take advantage of some of the independent slot-to slot communication existent in PXIe, and the redundancy and reconfigurability of the I/O platform elements to implement parts of the reliable run-time support.



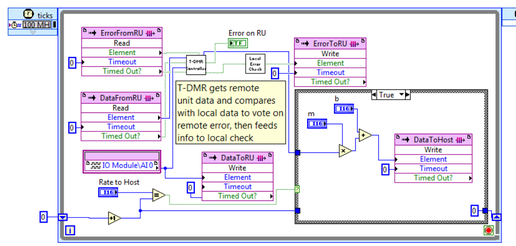
LabVIEW FPGA Task Block Diagrams (T1 top, T2 bottom) for Pattern 1



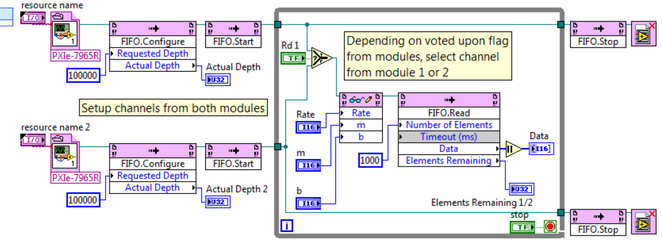
LabVIEW Host Block Diagram for Pattern 1



LabVIEW Host Front Panel for Pattern 1



LabVIEW FPGA DMR Block Diagram for Pattern 2



LabVIEW Host Block Diagram for Pattern 2

Related Work

- Researchers have extended conventional models of computation (e.g. dataflow or timed interaction of tasks) to specify reliability requirements, and include some synthesis efforts.
- LabVIEW framework supports multiple MoCs for functional aspects that can be synthesized to hardware. It has been extended with capabilities to capture and synthesize from non-functional requirements (e.g. throughput, latency). We extend this by including reliability requirements.
- Previous approaches have focused at the operating system (OS) level. We provide support at three levels: the ability to capture system level requirement with reliability constraints, the ability to integrate third party IP (e.g. OpenCPI), and the ability to support multiple commercial hardware platforms with corresponding run-time environments
- Prior work has disconnect between the model and the run-time environment. Inspired by Benoit's work, we propose a secondary run-time optimizing mapping engine that complements the initial system level mapping with updates to the platform model

Conclusions

- *We focused on two aspects of design and deployment of reliable systems on heterogeneous platforms*
  - Showed examples of how reliability/availability requirements can be mapped from application to platform using pattern information given MTTF/MTBF platform information
  - Used the NI PXIe platform and FlexRIO components to demonstrate the viability of a runtime environment that provides desired levels of reliability for two requirement patterns.
- *In the future, we intend to focus on*
  - Formalizing pattern specification language, modeling reliable platform elements, and defining techniques to map non-functional requirements on characterized platform
  - Combining individual tools and techniques into consistent automatic flow

ni.com/berkeley signal.ece.utexas.edu

International Conference on ReConFigurable Computing and FPGAs, Dec. 5-7, 2012, Cancun, Mexico