UF Herbert Wertheim College of Engineering
UNIVERSITY of FLORIDA

POWERING THE NEW ENGINEER TO TRANSFORM THE FUTURE

Department of Electrical & Computer Engineering

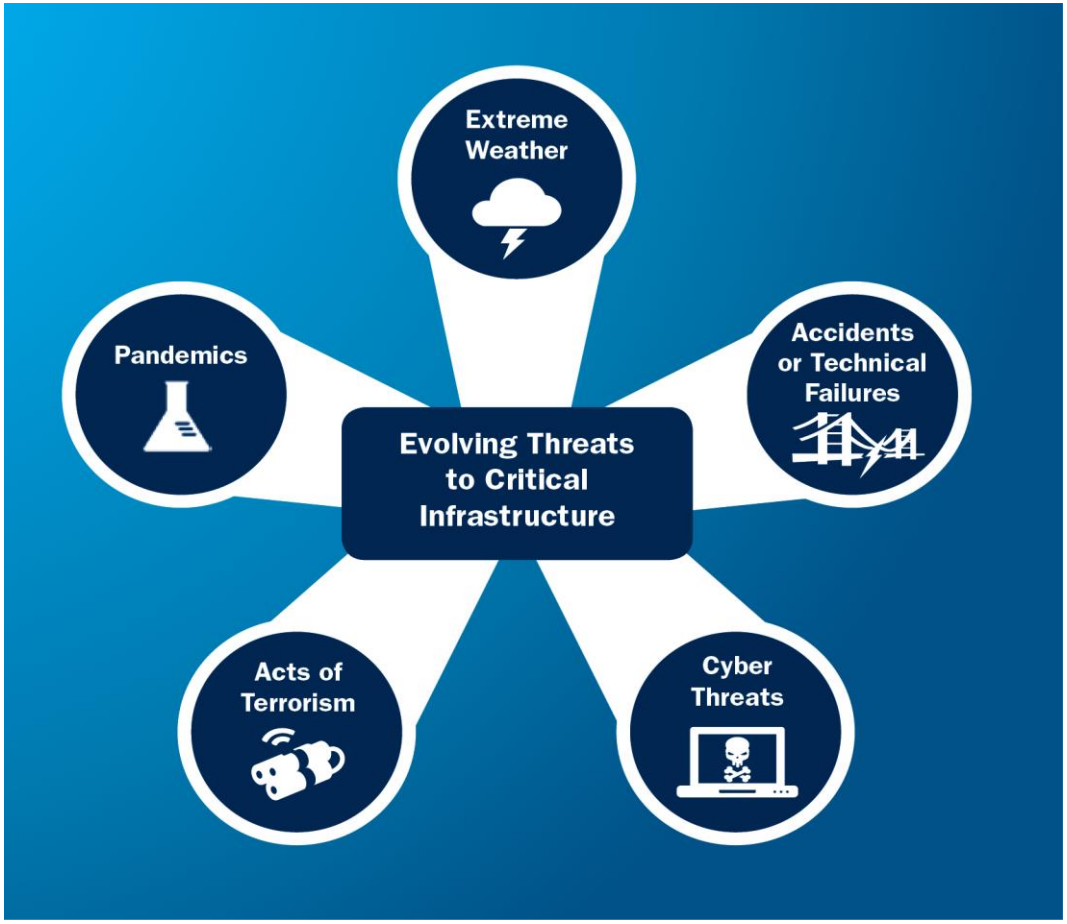
Advances In System Isolation For Cyber Attack Mitigation

Dr. Christophe Bobda

Agenda

- Motivation
- Security through Isolation
- Cloud in FPGA-Accelerated Clouds
- Security in System-On-Chips
- Design Framework

Evolving Threat to National Infrastructure



Challenges

- While well-known computer security techniques certainly work, they are not sufficient

  → size, scale, and scope inherent in complex national infrastructure

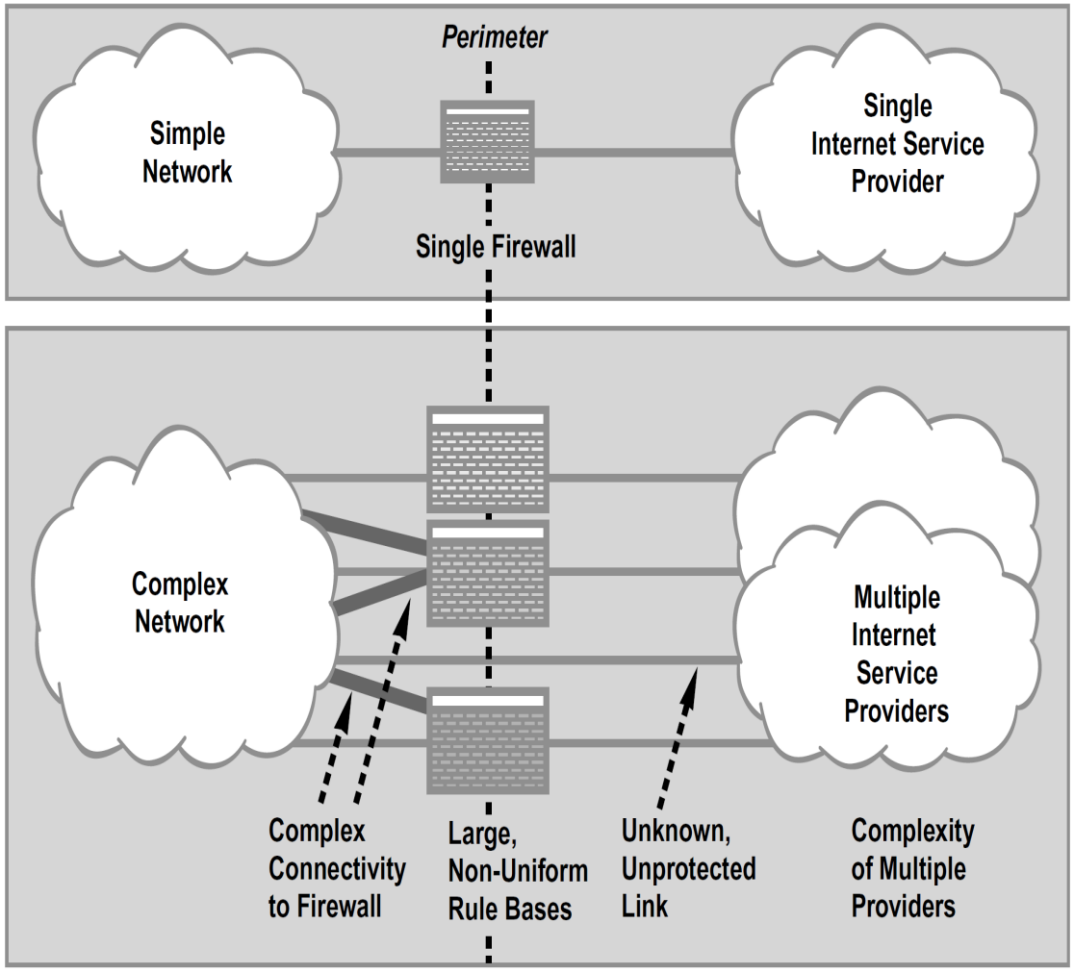
Defense Mechanism

- Deception
- Separation
- Diversity
- Consistency
- Depth
- Discretion
- Collection
- Correlation
- Awareness
- Response

Cybersecurity Mitigation

System Isolation

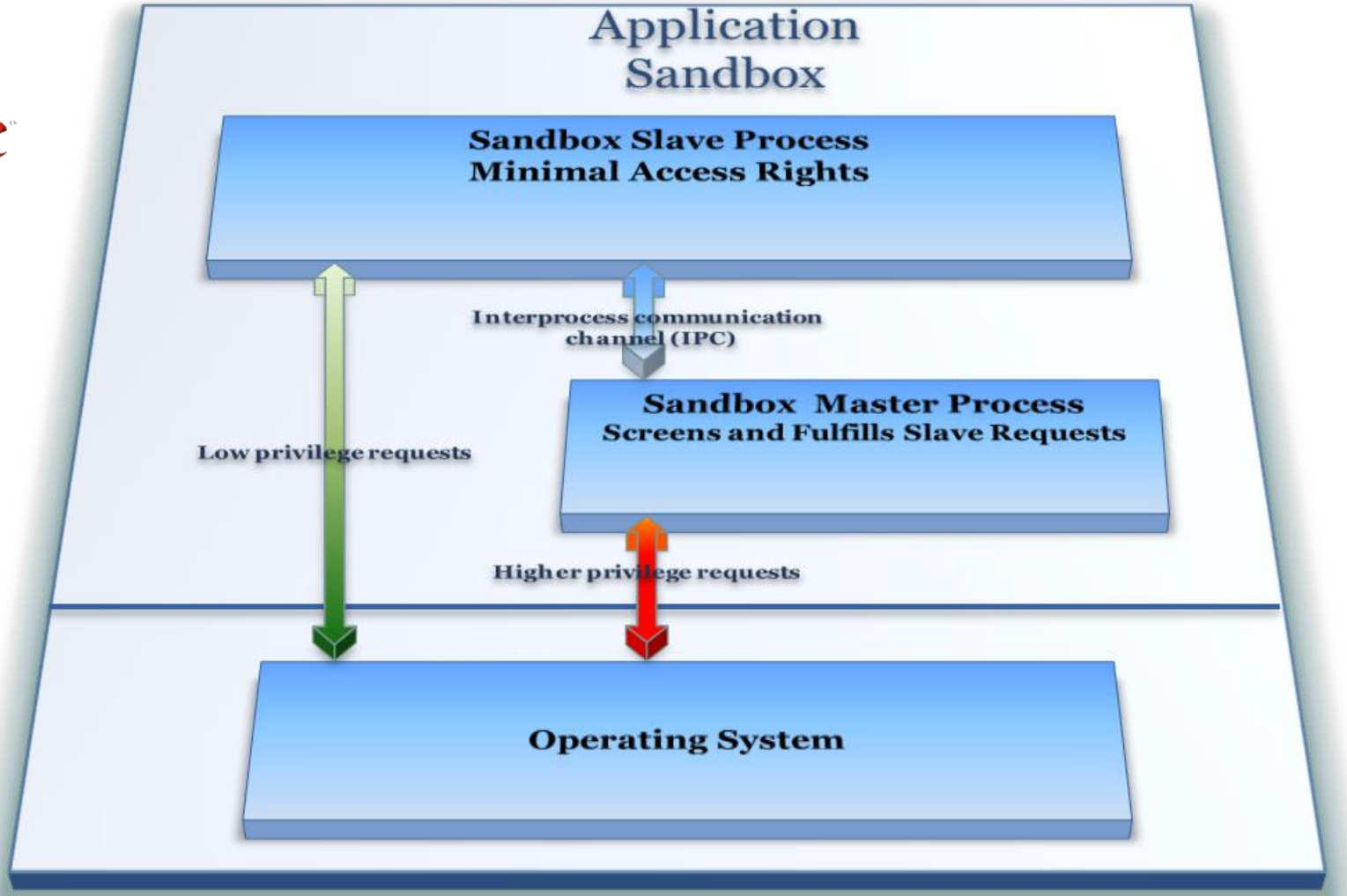
- Firewalls



Physical Separation

- Air Gapping

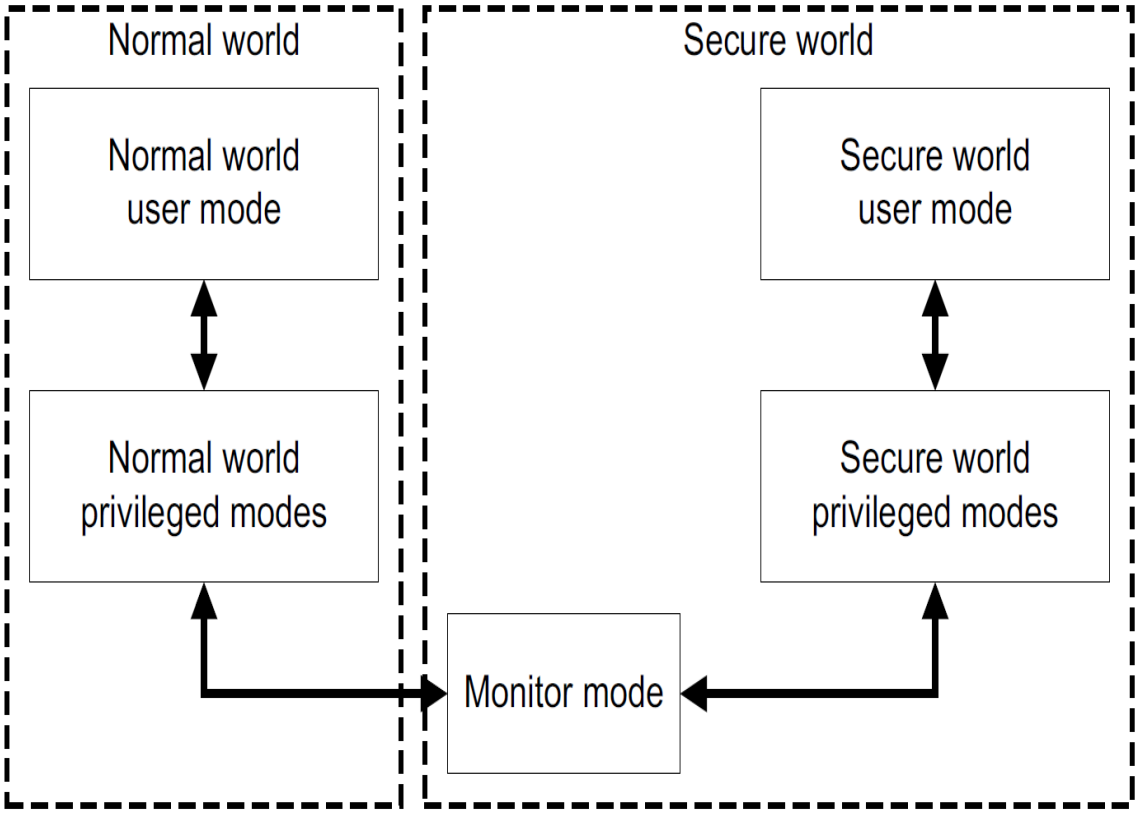
Application Sandbox Type B



TrustZone

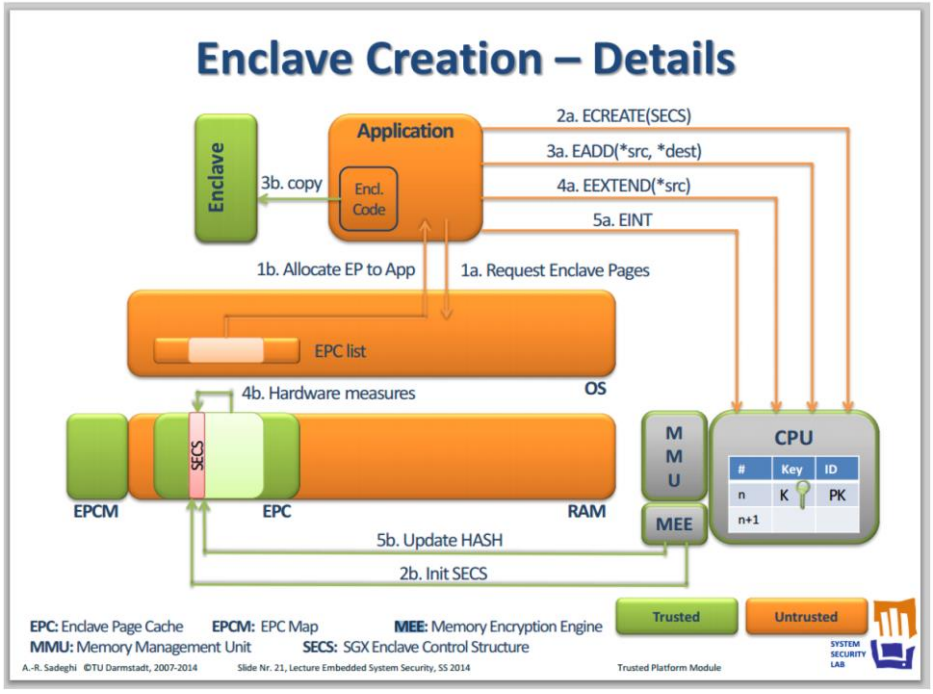
ARM®

Texas Instruments



Secure Enclave

- Intel Secure Guard Extension (SGX)
  - C3 runs MapReduce + Code and Data Secret
  - HEAVEN: Shields trusted applications in untrusted cloud



Re

- In
  -
  -
  -
  -
  -
  -
  -
  -
  -

Electronic Engineering JOURNAL

ARTICLES NEWS FISH FRY FRESH BYTES CH

cadence Now. Check for DFM Errors While You Design SEE DEMO ▶

feature article

May 29, 2018

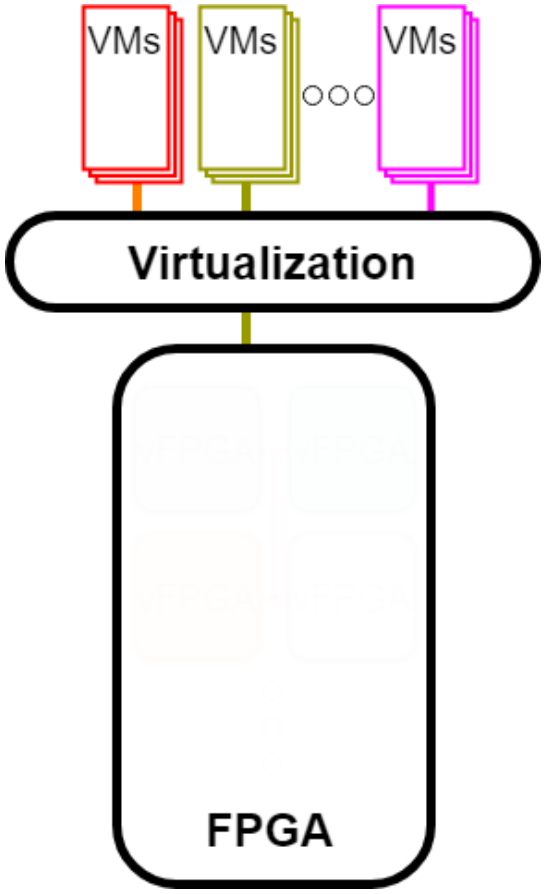
Intel Delivers Xeon Scalable Processor 6138P with Arria 10 GX 1150 FPGA

Ratchets Up FPGAs in Data Center

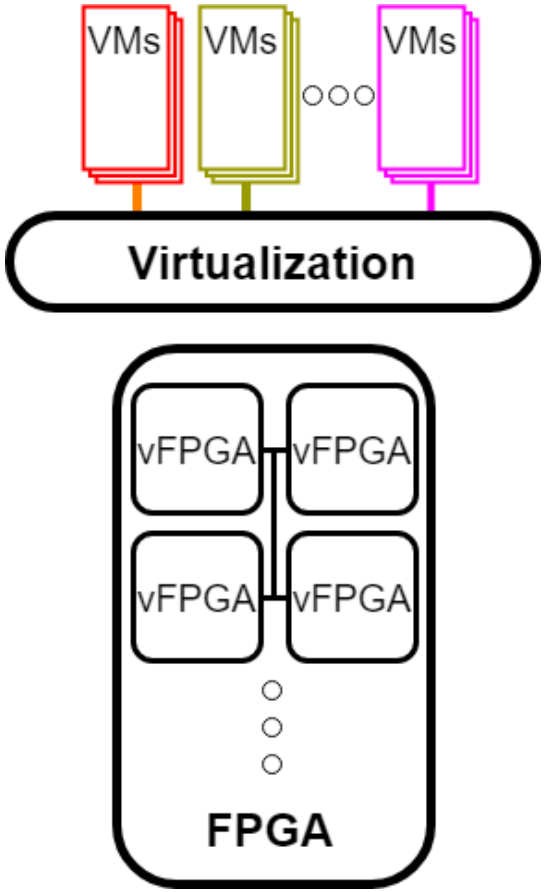
by Kevin Morris

Almost exactly four years ago, at Gigaom Structure 2014, Intel's Diane Bryant announced that the company would be "integrating [Intel's] industry-leading Xeon processor with a coherent FPGA in a single package, socket compatible to [the] standard Xeon E5 processor offerings." It was a bare-bones sort of announcement with zero details, except that she expected the combination

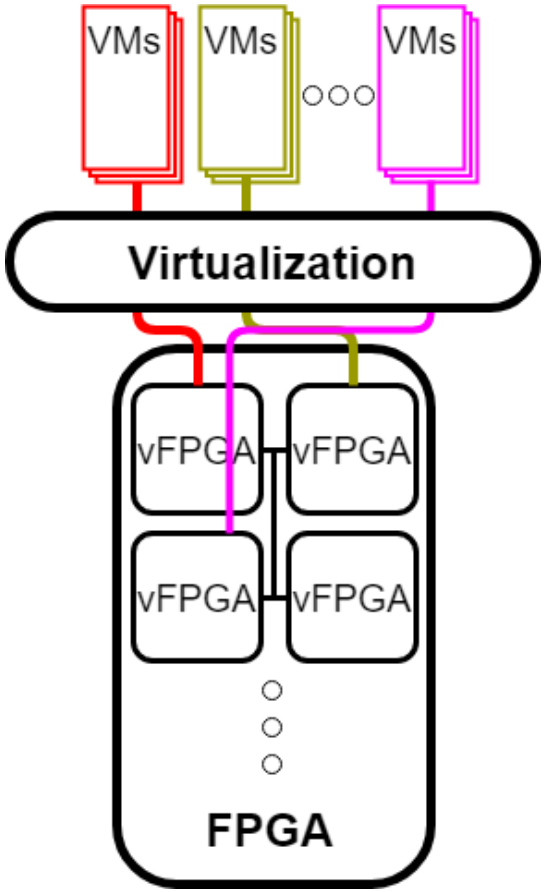
IaaS Resources Provisioning



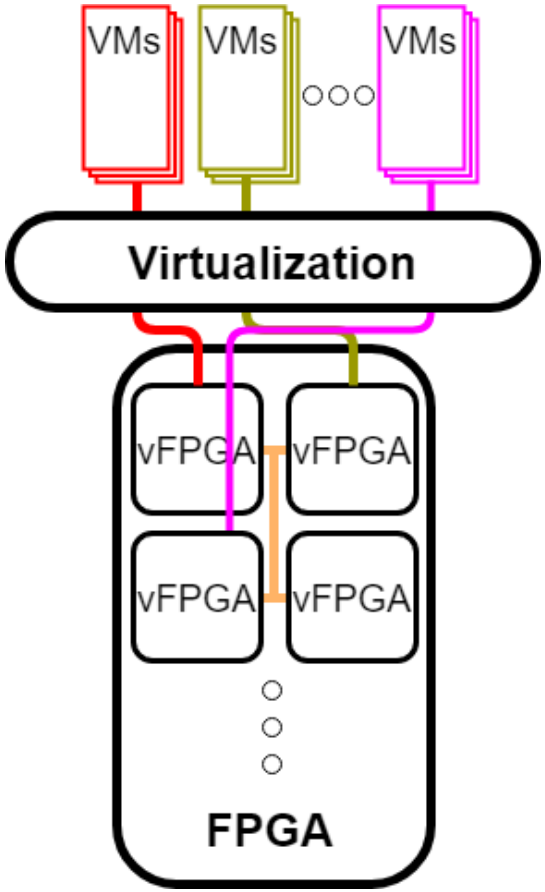
IaaS Resources Provisioning



IaaS Resources Provisioning



IaaS Resources Provisioning



New Challenges: Security

- Remote side channel attacks
  - Power monitoring (Zhao & Suh SP 18)
  - Leaking wires (Giechaskiel et al. ASIACCS'18)
  - Line probing (Ramesh et al. FCCM 18)

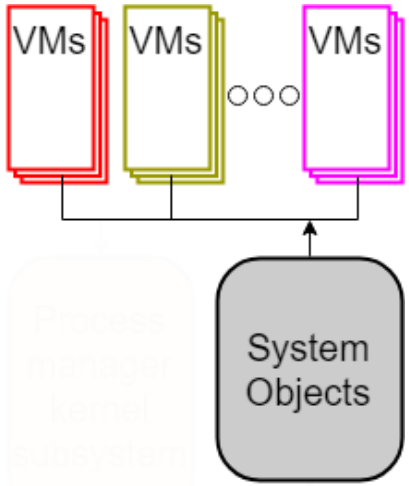
New Challenges: Security

- Remote side channel attacks
  - Power monitoring (Zhao & Suh SP 18)
  - Leaking wires (Giechaskiel et al. ASIACCS'18)
  - Line probing (Ramesh et al. FCCM 18)

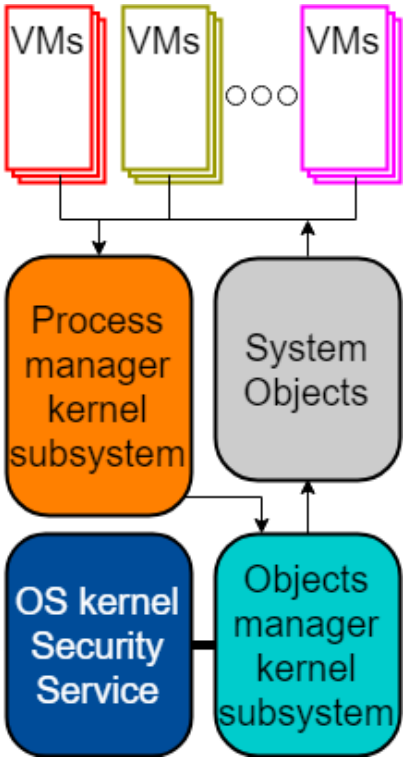
Hardware Isolation

- Shared FPGAs can be better protected by:
  - ***Extending*** *separation* ***kernel policies within hardware components,*** *and*
  - ***Enforcing*** *corresponding* ***access decision*** *rules* ***directly*** *in the hardware*

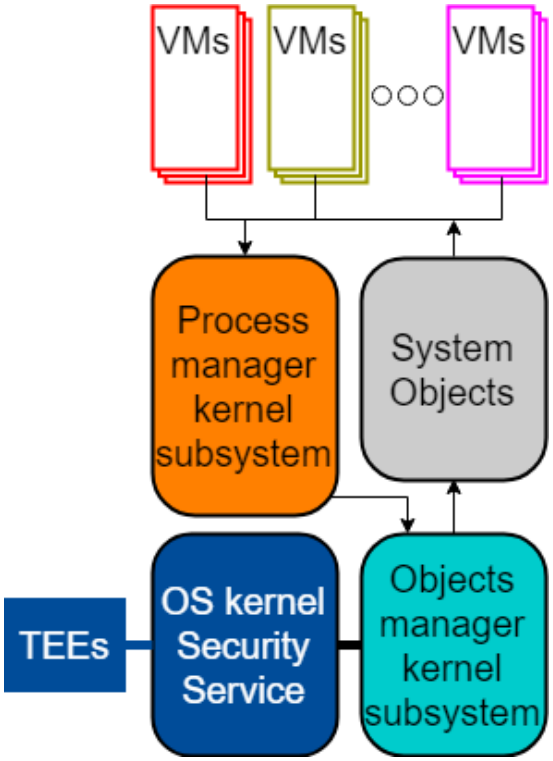
Domain separation in software environments



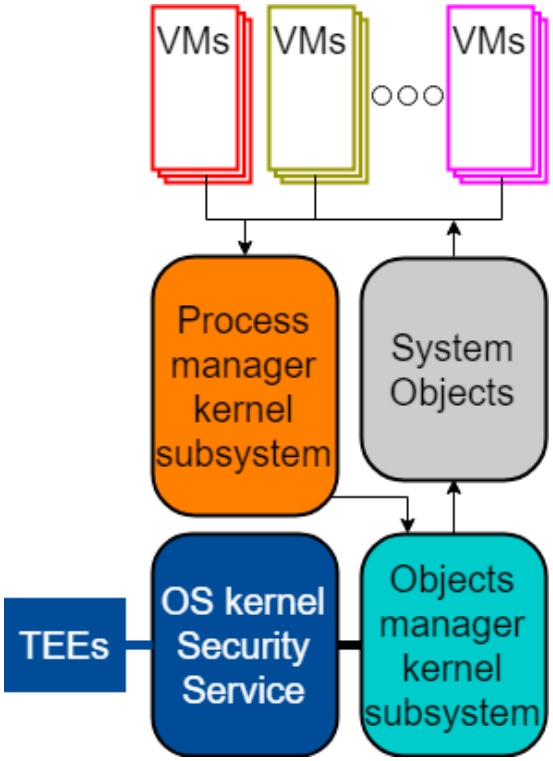
Domain separation in software environments



Domain separation in software environments

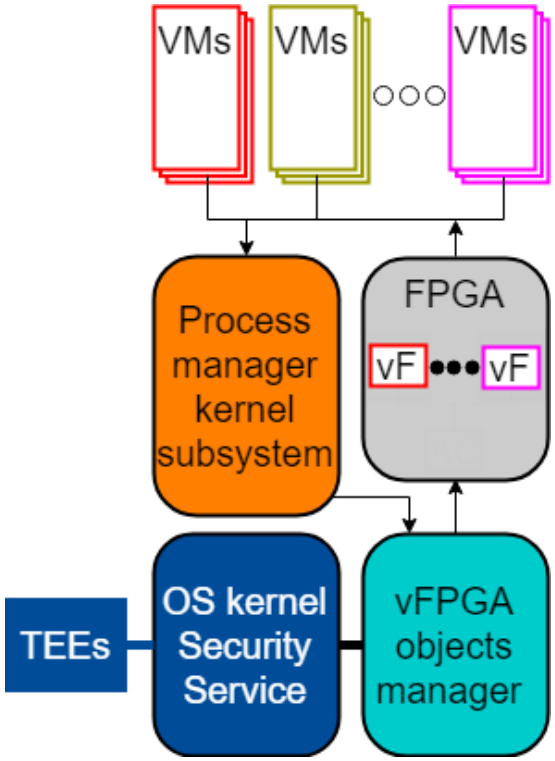


Using Workflows to Isolate vFPGAs (vFs) execution



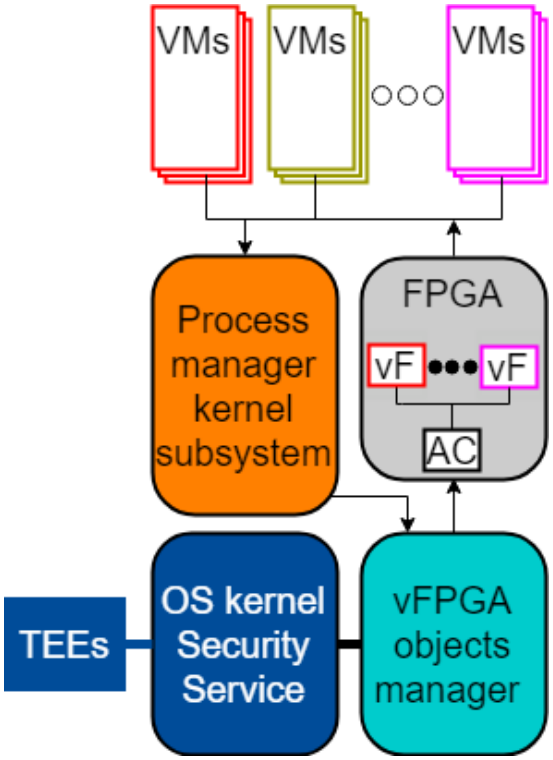
Using Workflows to Isolate vFPGAs (vFs) execution

1. Inherit runtime separation kernel's policies within the vFPGAs.



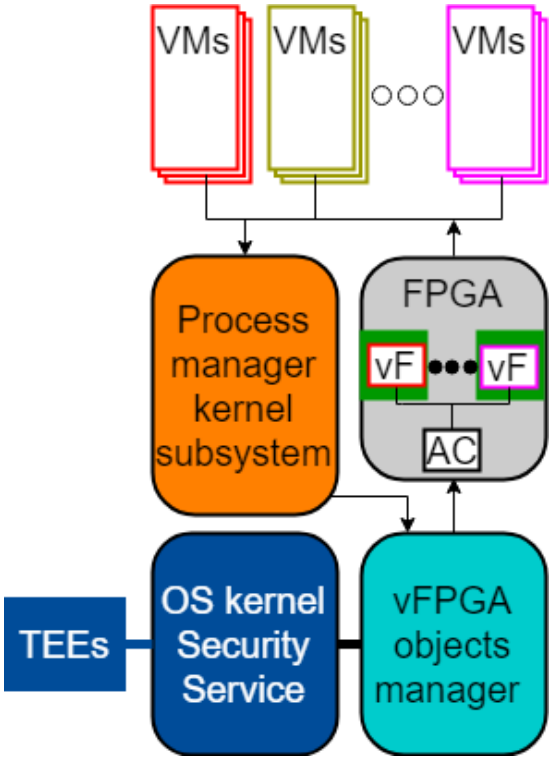
Using Workflows to Isolate vFPGAs (vFs) execution

2. Enforce access decisions directly in the FPGA through an access controller (AC) circuit.



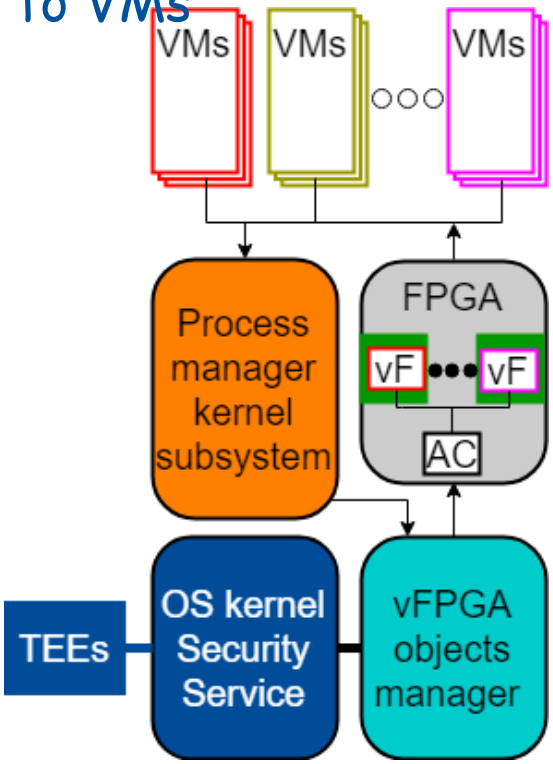
Using Workflows to Isolate vFPGAs (vFs) execution

3. Isolate vFs from shared resources through a hardware sandbox



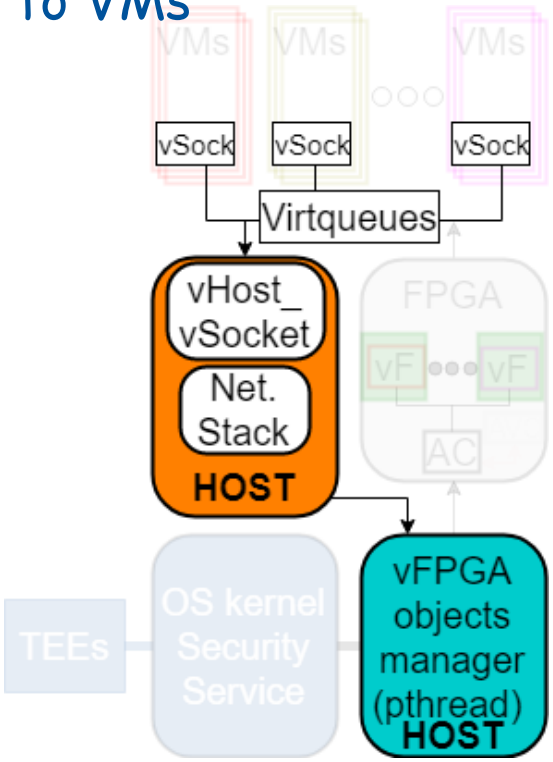
Implementation Details

- Provisioning vFPGAs to VMs



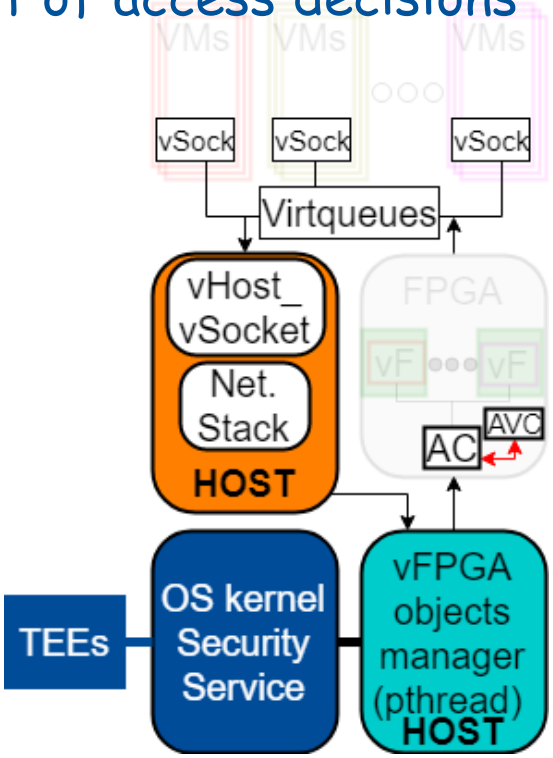
Implementation Details

- Provisioning vFPGAs to VMs



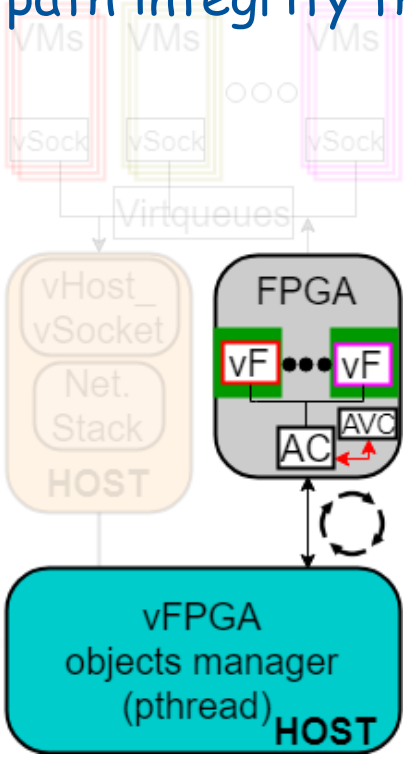
Implementation Details

- Runtime management of access decisions

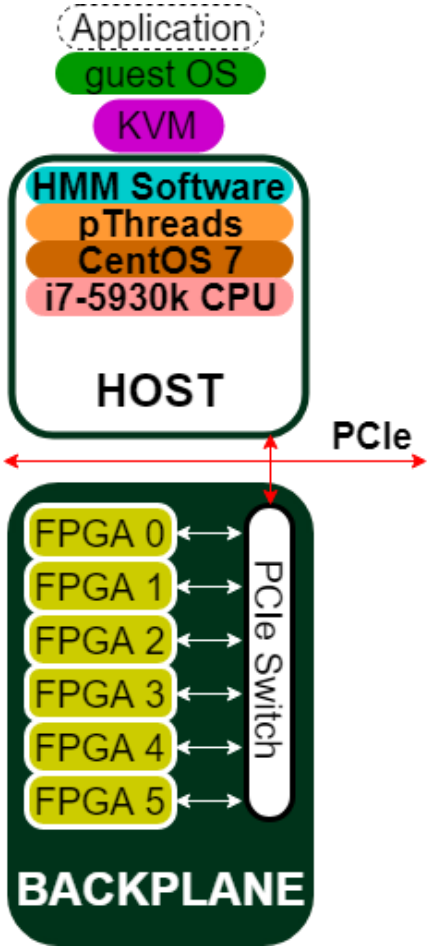


Implementation Details

- Preserving VM-vFPGA IO path integrity through chained authentication



Evaluation - Setup



Evaluation I

- Security performance
  - **Secure by design**
    - Solution implements "deny by default" rule **by design**.
    - Access decision is made by the OS kernel according to its security policy.
    - Correctness of decisions depends on policy definition.
    - Access is granted *only if the security policy explicitly allows it.*
      - E.g., On CentOS 7, 200K+ test labels could not gain access to accelerators since policy didn't define the requested access

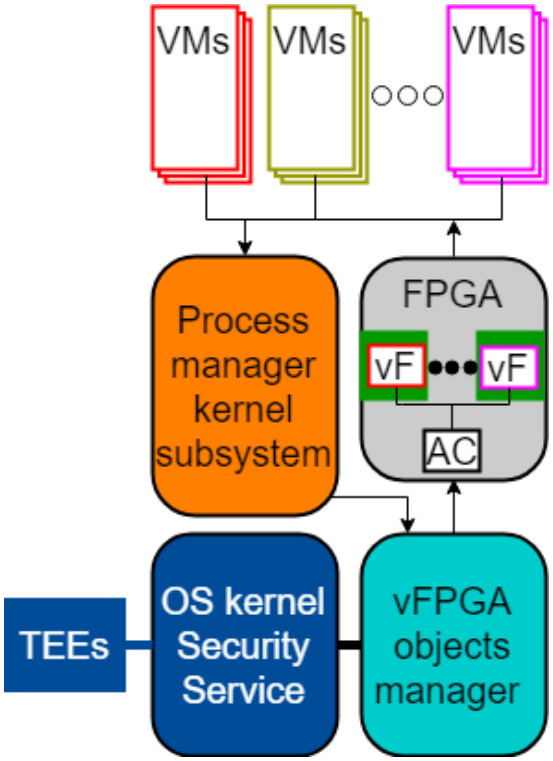
Evaluation I

- Security operations overhead on VM users' applications

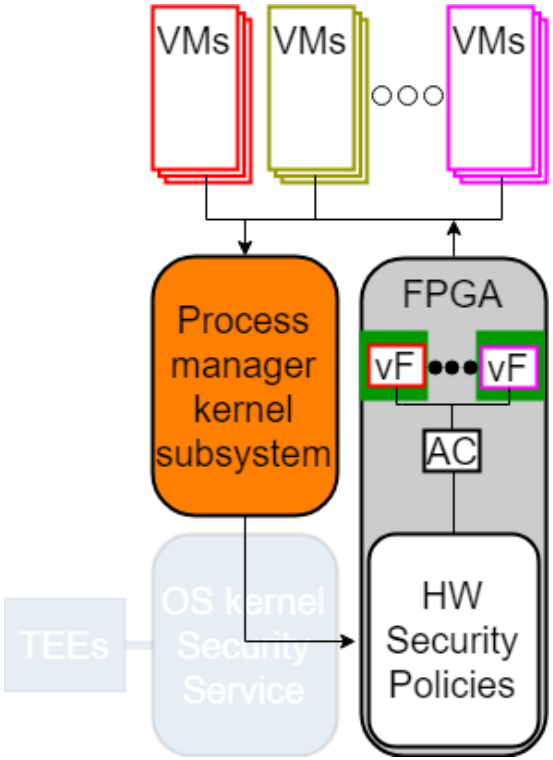
Calls	Execution Time (in us)
Label Creation	35
Decision Lookup (AVC Hit)	0.02
Decision Lookup (AVC Miss)	35.02
Decision Administration	0.012

Benchmarks	Input Size (Bytes)	Total Time W/Out Iso.	Time Spent on FPGA
Matrix Multiplication	64x64	46.55 us	30.5 us
Inner Product	256x1	21.55 us	5.5 us
Convolution 2D	64x64	164.05 us	148 us
Vector Addition	256x1	21.05 us	5 us

Performance Improvement



Performance Improvement



Performance Improvement

Calls	Execution Time (in us)
Label Creation	35
Decision Lookup (AVC Hit)	0.02
Decision Lookup (AVC Miss)	35.02
Decision Administration	0.012

Security Policy	Decision Administration Time	Speedup Gained
TE	0.015 us	2335x
TE-MLS	0.037 us	946x
TE-BIBA	0.036 us	973x
Controlled Sharing	0.032 us	1094x

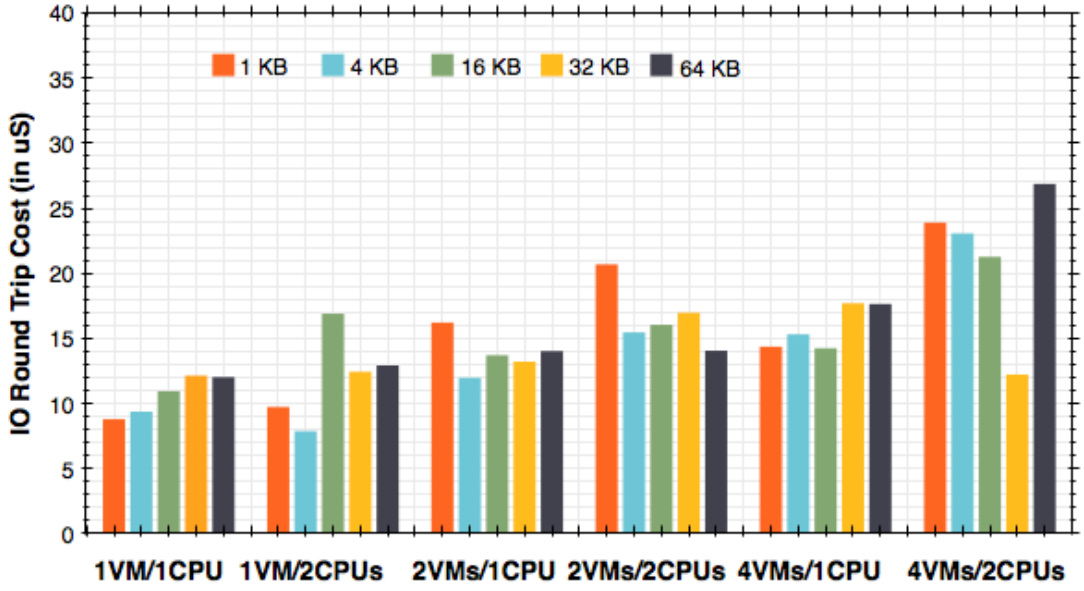
Evaluation II

- Resource Overhead FPGAs resources

Xilinx Kintex Ultrascale XCKU060	LUTs	FFs	BRAMs
Available	331680	663360	1080
Used	24250	26364	66
Utilization	7.31%	3.97%	6.11%
——Used by HMC Pico Framework	5778	11449	42
——Used by Access Enforcement Function	281	193	0
——Used by TE Policy Module	73	65	0
——Used by TE-MLS Policy Module (4 levels)	91	70	0
——Used by TE-BIBA Policy Module (4 levels)	91	70	0
——Used by HW Sandboxes (W/out Checker)	225	22	14
——Used by HW BasicRSA Core (2048-bit key)	4032	4696	4
——Used by HW AES Core (128-bit key)	444	402	5
——Used by SHA256 Core	1353	1076	1
——Used by RS232 Core	295	298	0
——Used by Accelerators Modules	11587	8023	0

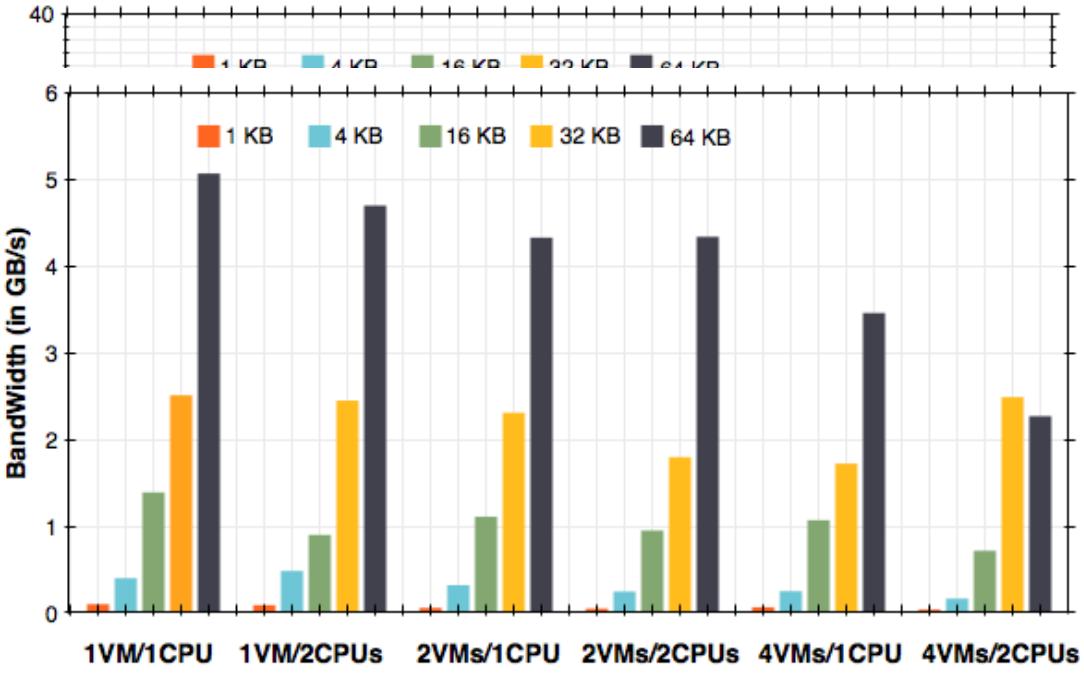
Evaluation III

- VM-vFPGA IO path performance and scalability analysis



Evaluation IV

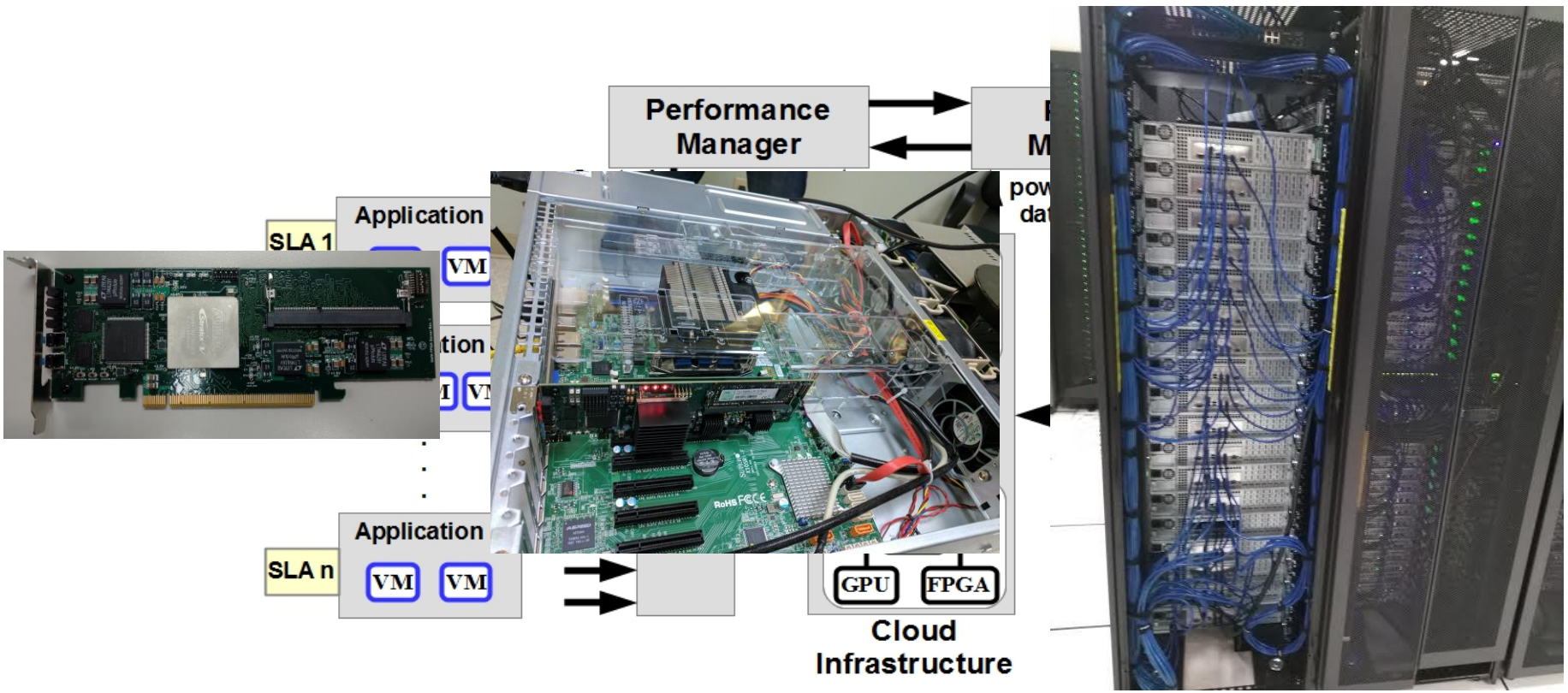
- VM-vFPGA IO path performance and scalability analysis



Summary

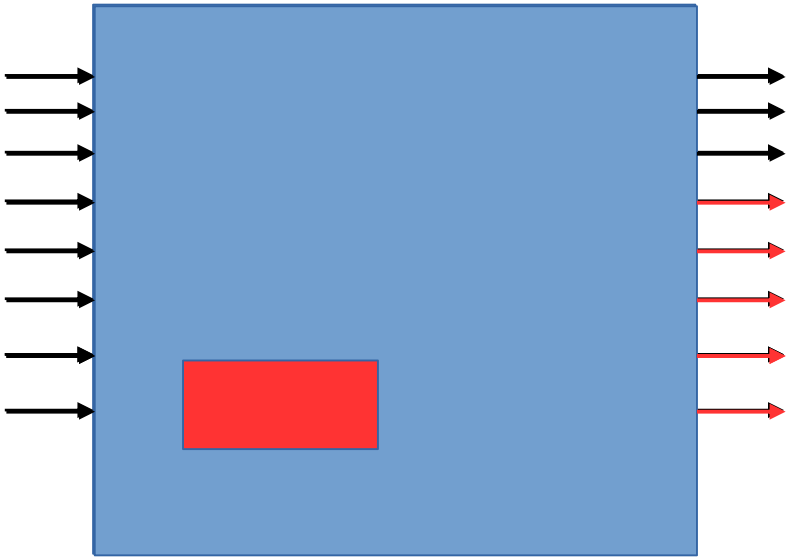
- Proposed solution provides to VMs, secure and isolated execution of vFPGA objects for FPGA-accelerated IaaS cloud.
- Isolation benefits path costs at worst ~11 clock cycles on FPGA.
- Proposed solution is fully compatible with existing cloud software

Architecture Diversification in Cl Computing Systems



ONR Award Nr: N00014-16-1-2014

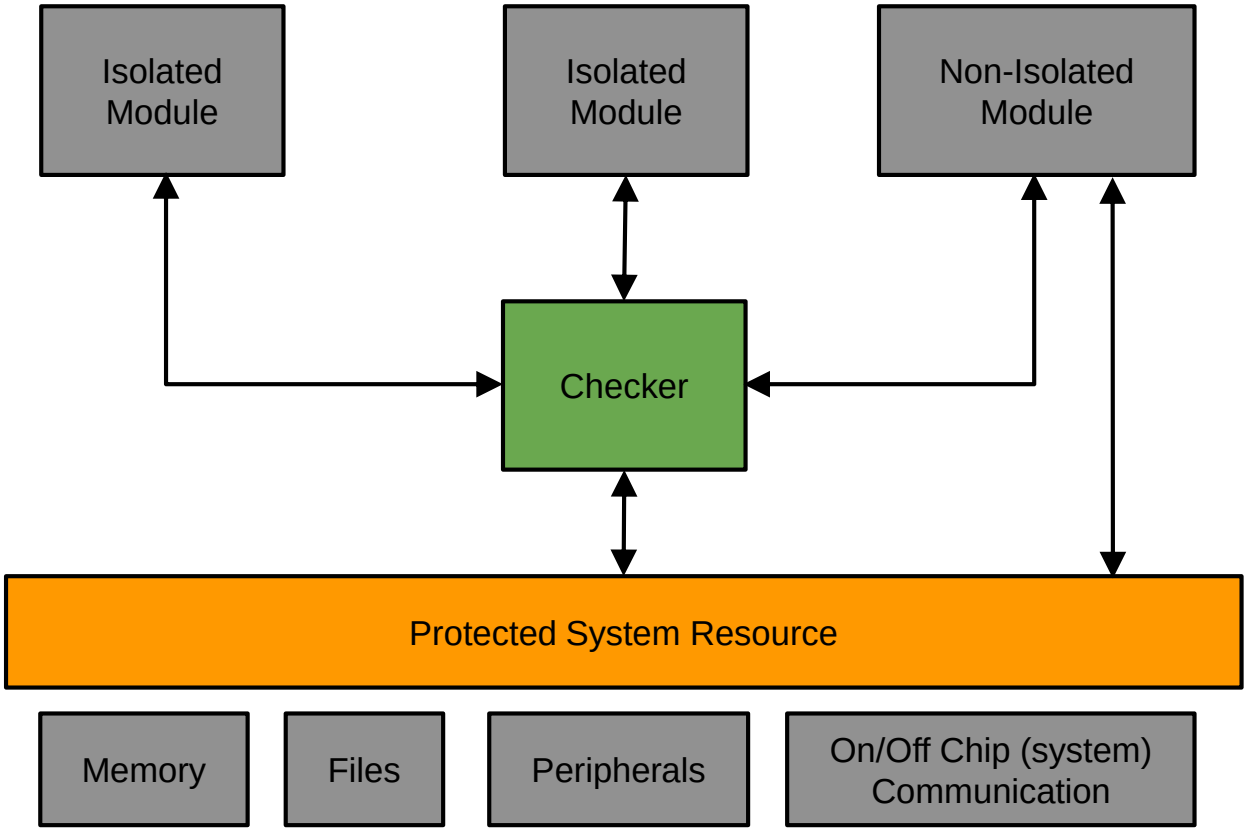
Hardware Trojan



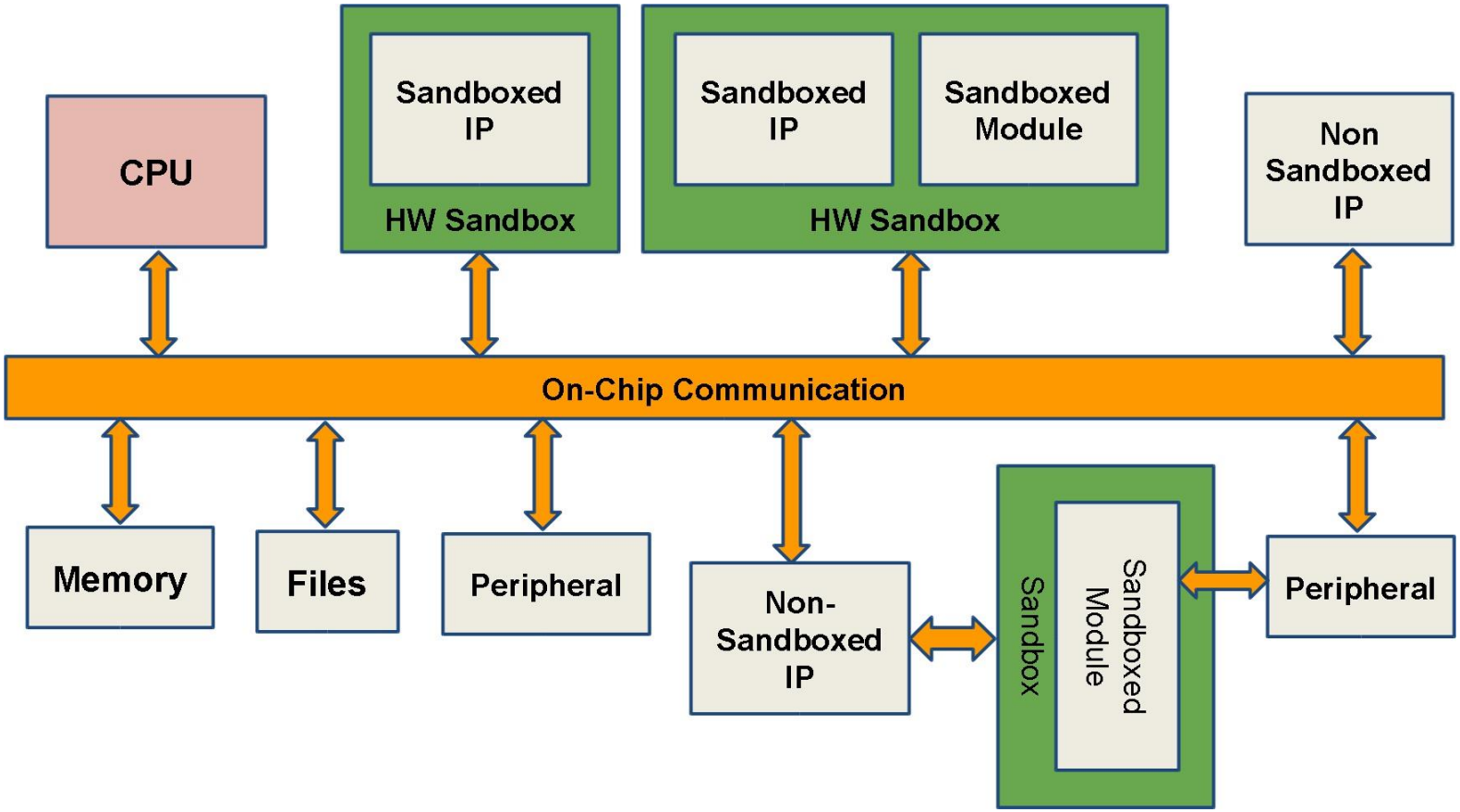
Research Rationale

- Trojans can be hidden in IPs, but as longer as they don't manifest, the system is safe.
- Trojans and their manifestations have the same relationship as errors as faults.
- The rationale of our research is therefore the same as fault-tolerant systems, namely to design and built systems along with dynamic methods, capable of detecting manifestation of Trojans at run-time and to prevent potential damage to the system.

On-Chip Module Isolation

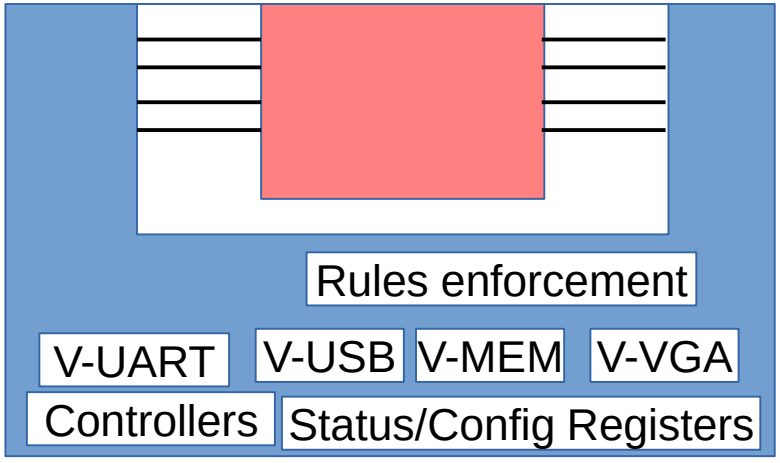


HW Sandboxing: SoC-Integration



HW Sandboxing Concepts

1. Managed Code
2. In-Line Reference Monitor
3. Hardware Memory Isolation
4. Resource Virtualization



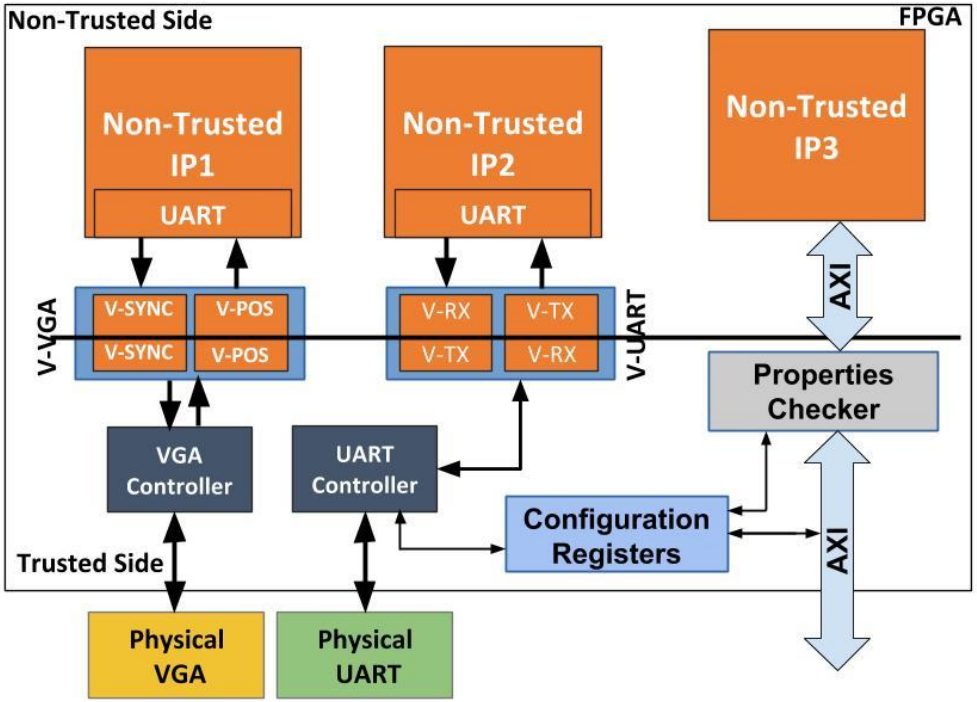
Evaluation

Design	Trojan Class	Checker	Type Area Overhead	Delay Overhead
T-300	Info leak	Cycle Sequence	243 LUT (1.38%)	-0.729 ns
T-400	Info leak	Cycle Sequence	278 LUT(1.58%)	-0.046 ns
T-400	DoS	Cycle Sequence	269 LUT (1.52%)	-0.174 ns
T-900	DoS	Cycle Sequence	265 LUT (1.51%) -	0.149 ns

OVL(Open Verification Library)

- CC(0): xmitH == ’1’
- CC(1) to CC(16): uart xmit doneH == ’0’ AND uart xmit == ’0’
- CC(17+(16*i)) to CC(16+(16*(i+1))): uart xmit doneH == ’0’ AND uart xmit == xmit data(i), where i is the index of xmit data, range [0,7]
- CC(145) to CC(175): uart xmit doneH == ’0’ AND uart xmit == ’1’
- CC(176): uart xmit doneH == ’1’

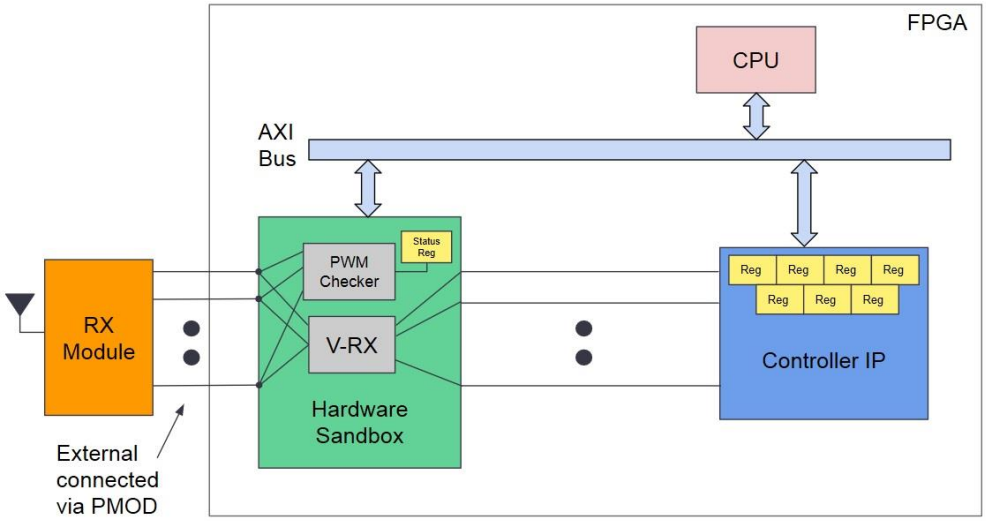
Evaluation



Virtual Resource	Area Overhead	Delay Overhead
VGA (V-VGA)	23 LUT (0.13%)	-0.037 ns
RS232-UART (V-UART)	120 LUT (0.68%)	-0.567 ns

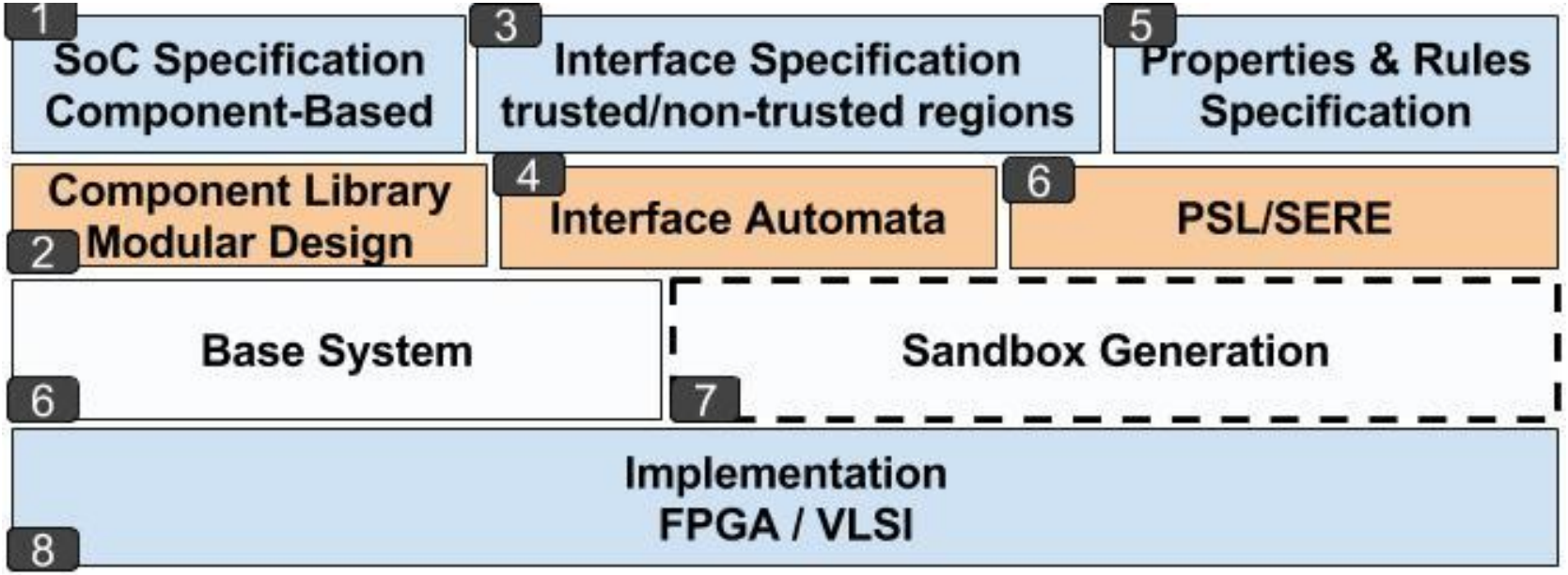
ARFL SFFP Award 2015 - 2016

Evaluation



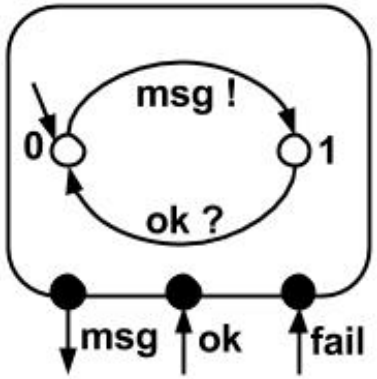
- Spectrum controller in the 2.4 GHz band using the DSM2 DSSS-based technology.
- Receiver decodes transmitted signals into seven channels to control the drone
  - throttle (throttle), ailerons (aile), elevators (elev), rudder (rudd), gear (gear), auxiliary1 (aux1), and auxiliary2 (aux2).
  - PWM between 5% and 9%
- Adapt cycle sequence to the dynamic nature of PWM

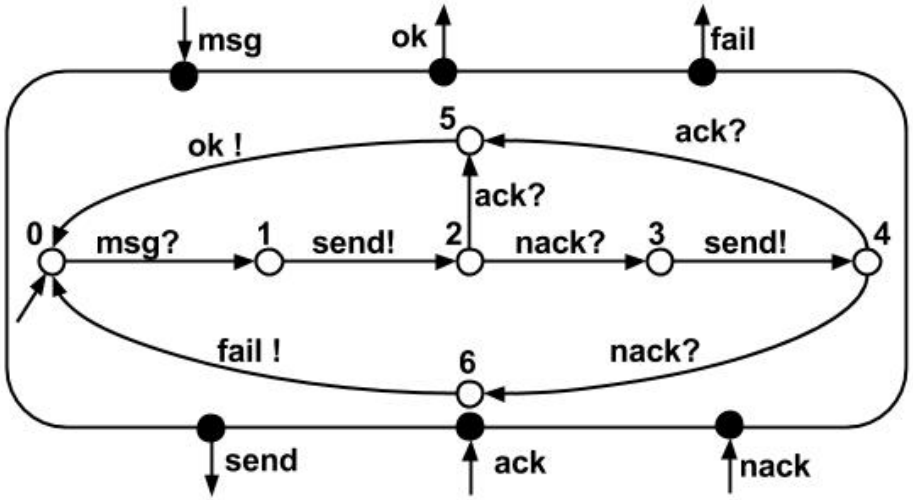
Design Flow: Proposed Approach



Interface Specification

- A interface automaton $F = \langle Q, q_0, Q_f, A^I, A^O, A^H, \delta \rangle$ consists of
  - a finite set Q of states,
  - an initial state $q_0 \in Q$; $Q_f \in Q$; set of final states,
  - three pairwise disjoint sets A^I , A^O, and A^H of input, output, and hidden actions,
  - *a set* $\delta \subseteq Q \times A \times Q$ *of* transitions*,*
    where $A = A^I \cup A^O \cup A^H$ *is the set of all* actions*.*





Interface Specification

- **Compositionality Given two IA**
  - $S = <Q_S, q_{0S}, Q_{fS}, A_S^I, A_S^O, A_S^H, \delta_S>$ and
  - $T = <Q_T, q_{0T}, Q_{fT}, A_T^I, A_T^O, A_T^H, \delta_T>$

- The $S \otimes T$ product of S and T is defined as
  $S \otimes T = <Q_{S\otimes T}, q_{0S\otimes T}, Q_{fS\otimes T}, A^I_{S\otimes T}, A^O_{S\otimes T}, A^H_{S\otimes T}, \delta_{S\otimes T}>$ with
  - $Q_{S\otimes T} = Q_S \times Q_T$
  - $q_{0S\otimes T} = (q_{0S}, q_{0T})$
  - $A^I_{S\otimes T} = A_S^I \cup A_T^I - Share(S,T)$,
  - $A^O_{S\otimes T} = A_S^O \cup A_T^O - Share(S,T)$
  - $A^H_{S\otimes T} = A_S^H \cup A_T^H \cup Share(S,T)$
  - $\delta_{S\otimes T} = \{(q,s), a, (q',s) \mid (q,a,q') \in \delta_S \wedge a \notin Share(S,T) \wedge s \in Q_T\} \cup \{(q,s), a, (q,s') \mid (s,a,s') \in \delta_T \wedge a \notin Share(S,T) \wedge q \in Q_S\}$
    $\cup$
    $\{(q,s), a, (q',s') \mid (q,a,q') \in \delta_S \wedge (s,a,s') \in \delta_T \wedge a \in Share(S,T)\}$

Interface Specification

- Illegal states

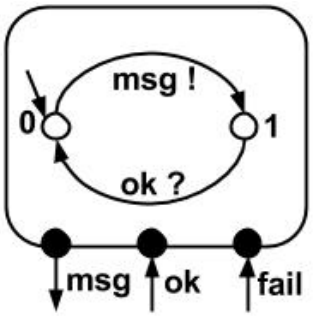
  $Illegal(S,T) = \{(q,s) \in Q_S \mathbf{X} Q_T | \exists a \notin Share(S,T),$
  $st\ (a \in A_S^O(q) \wedge \notin A_T^I(q))\ or\ (a \in A_T^O(q) \wedge \notin A_S^I(q))\}$

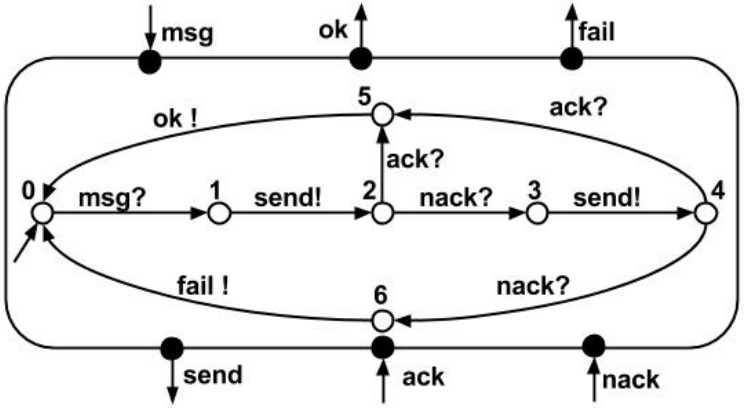
- A component, C, is a tuple (U, I) where
  - U is the core function of C
  - $I =< Q, q_0, Q_f, A^I, A^O, A^H, \delta >$ is an interface through which C interacts with other components

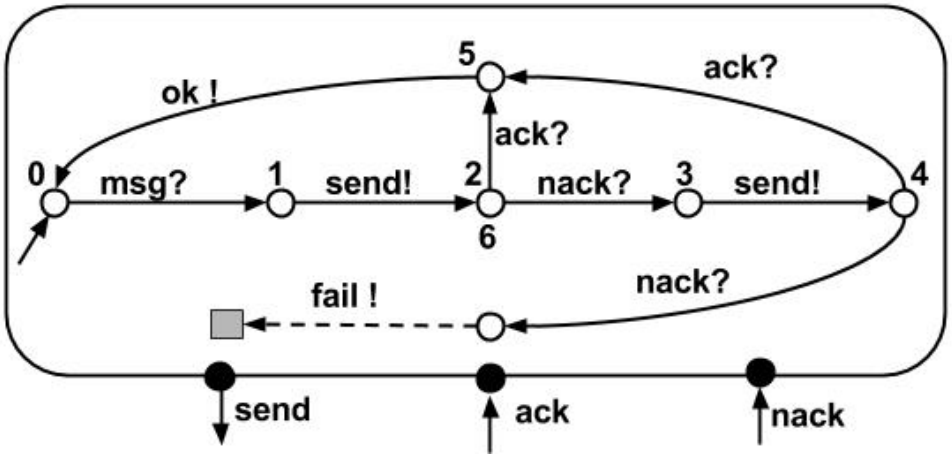
- Goal: Use component-based design and compose interfaces for
  - compatibility check
  - resource optimization

Interface Specification

- Assumption in IA: Environment will behave legally







Property Specification Language

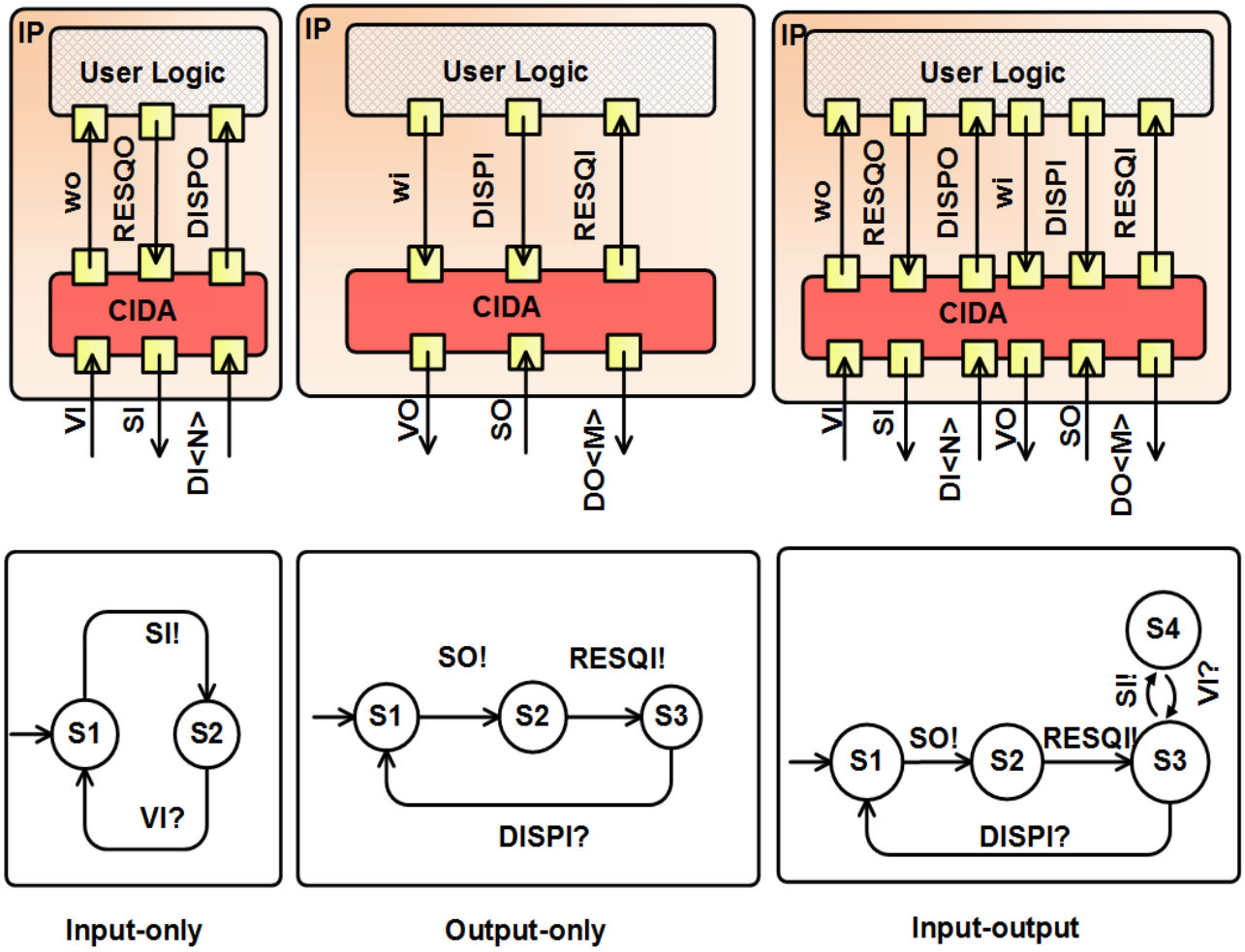
- PSL: Assertion Based Verification language
- Originated from the IBM Sugar language used for model checking, and evolved into an IEEE standard (1850-2005).
- PSL can be used to specify temporal properties of systems
  - combination of Linear Time Logic (LTL) and regular expression
- PSL consists of 4 layers
  - Boolean layer (*not* **reset** *and* **rd_en**) or (**reset** && **rd_en**).
  - Temporal layer
    - *always* **start** -> *next* **busy**
    - {[*]; **req**; **ack**} |=> {**start**; **busy**[*]; **done**}
  - Verification layer: *assert* → control verification
  - Modelling layer

Property Specification Language

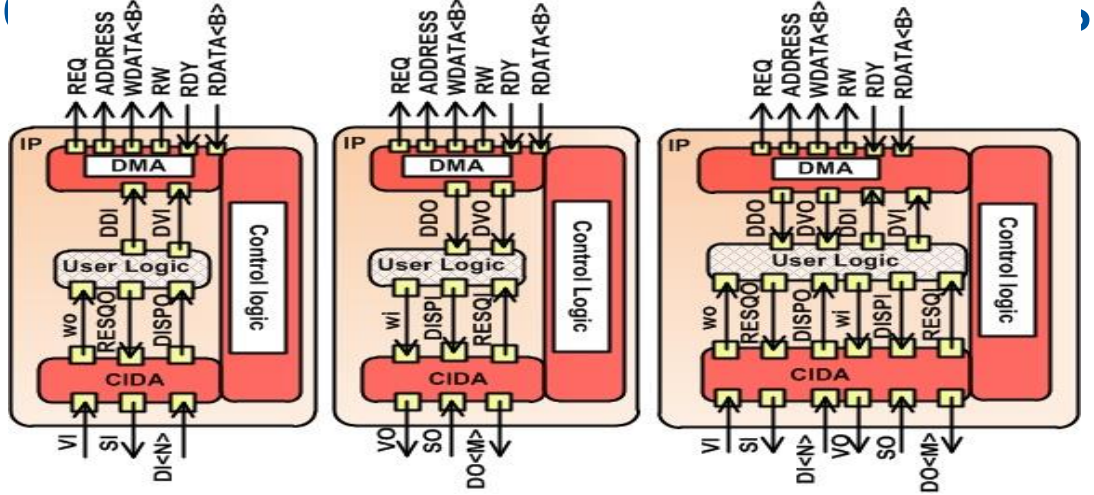
- PSL/SERE (Sequential-Extended Regular Expression)
  - Definition
    - If b is a Boolean expression and r, r_1 and r_2 are SERES then the following expression are SERE
      - b
      - {r}
      - r_1: r_2 (concatenation)
      - r_1; r_2 (fusion)
      - r_1| r_2 (or)
      - r_1&& r_2 (length matching and)
      - [*0]
      - r[*] (Kleene)
  - Sugaring
    - *always* **start** -> *next* **busy**
    - {[*]; **req**; **ack**} |=> {**start**; **busy**[*]; **done**}

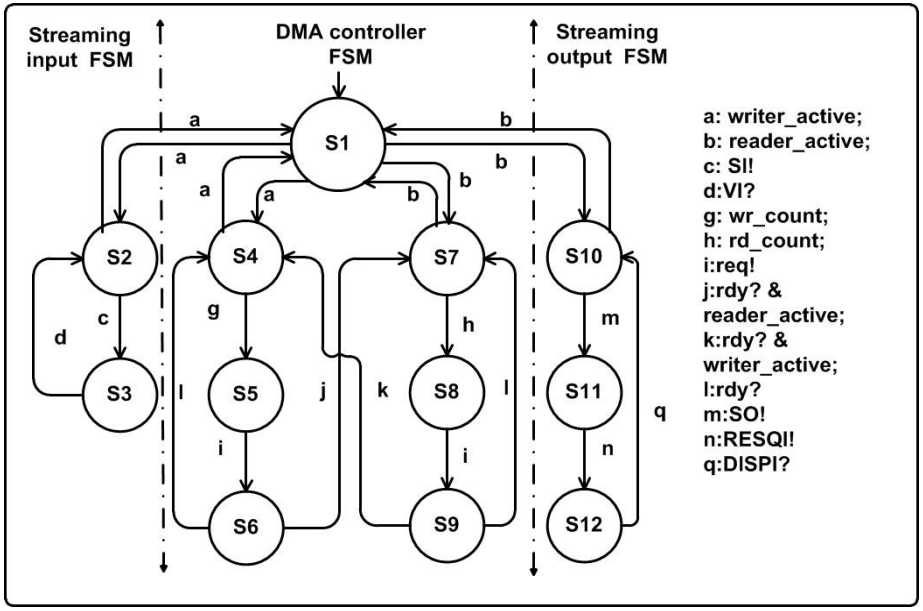
Regular expressions

Component Interconnect for Data Access



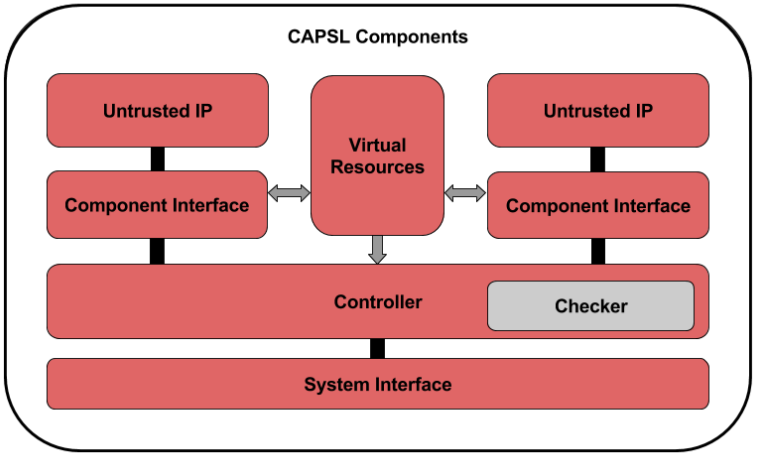
Component Interconnect for Data Access

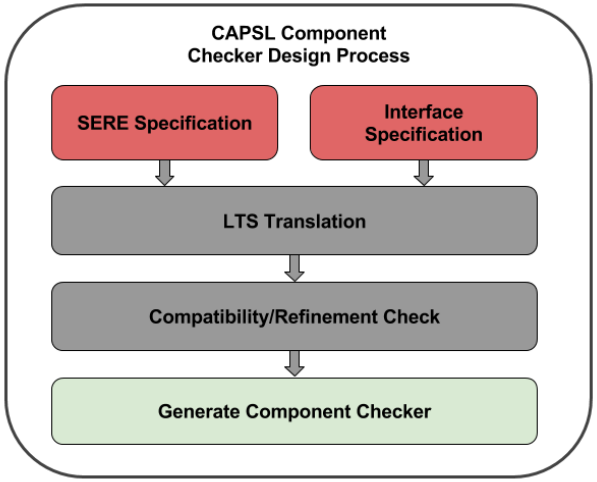





CAPSL
Component Authentication Process for Sandboxed Layouts

Components & Flow

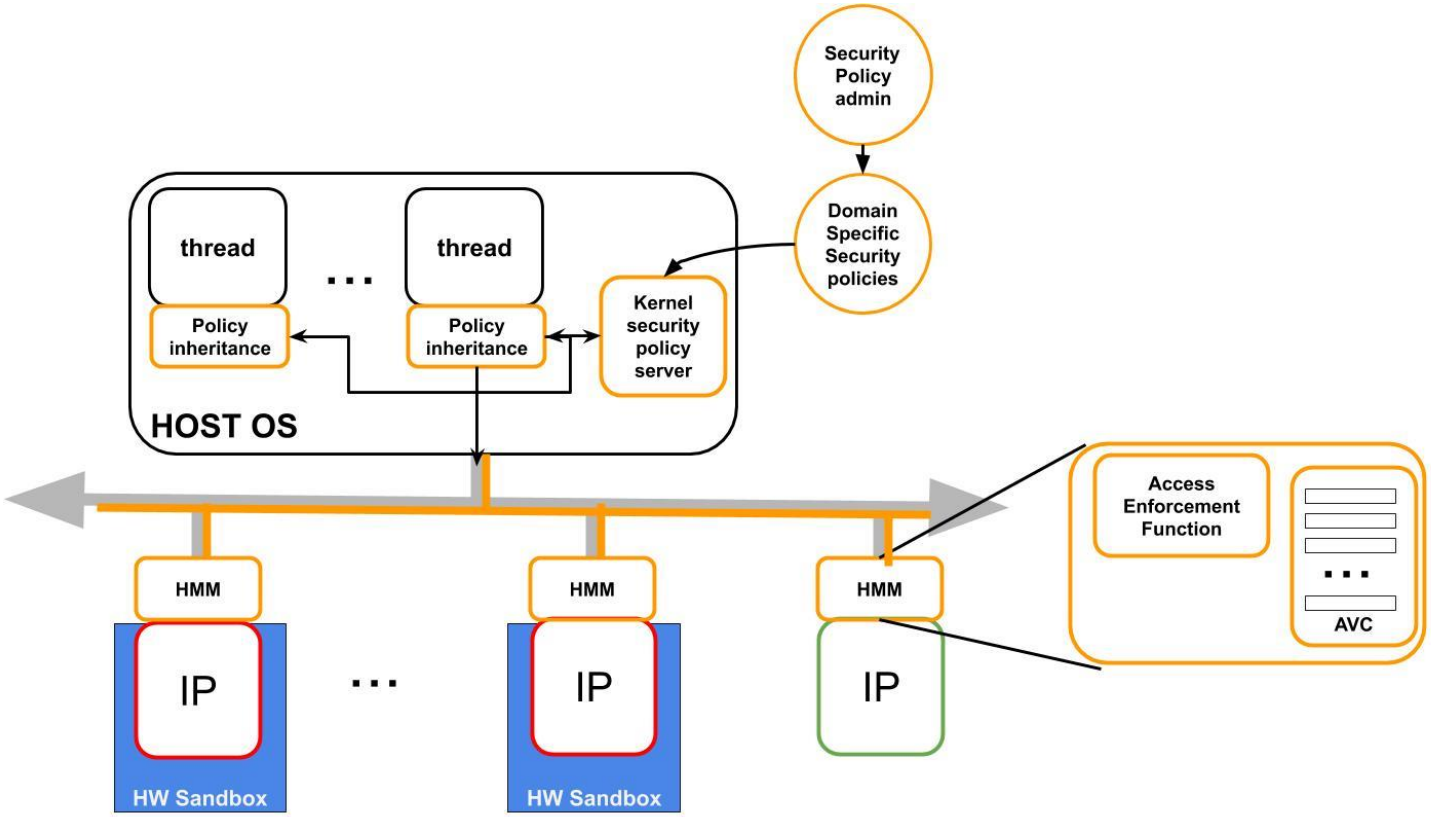




VIVADO™

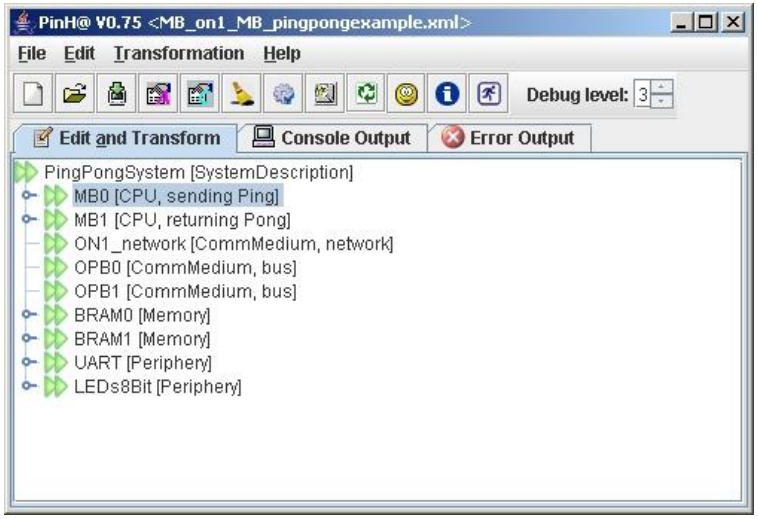
SYSTEMC™

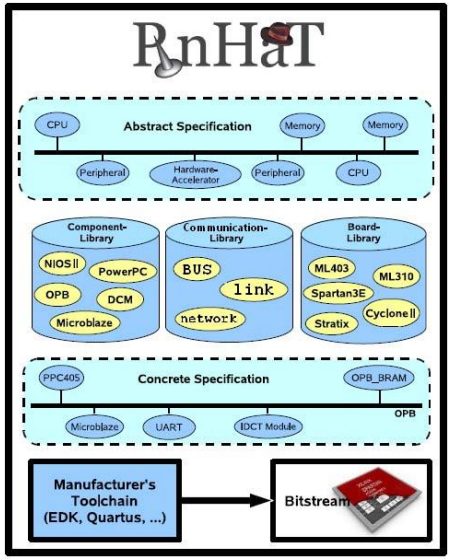
Security Integration
The Flask Architecture Model



MEXT (Multiprocessor On-Chip Exploration 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





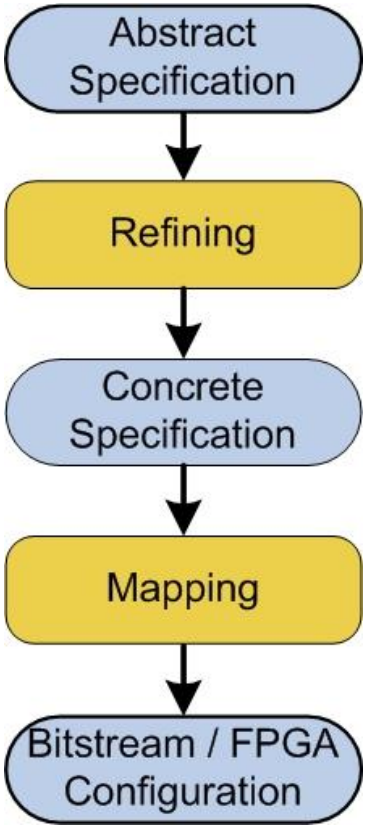
Automatic Generation of the MPSoC Infrastructure

Input: Platform-independent *Abstract Specification* (CPU, Memory, CommMedium, Periphery and HWaccelerator)

- Transforming of an Abstract Specification into a Platform-dependent *Concrete Specification* (e.g.: Platform → Xilinx ML310, CPU → PowerPC, Memory → BRAM, CommMedium → PLB,...)

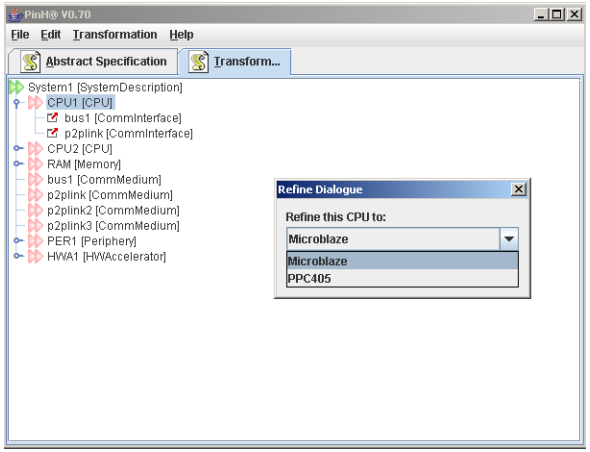
- Generation of the platform-dependent hardware description files (Concrete Component Description)

Output: FPGA configuration file

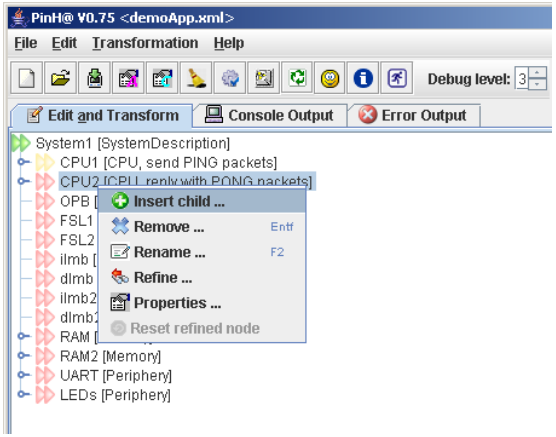


- The *Concrete Specification* can also be the result of the architectural synthesis

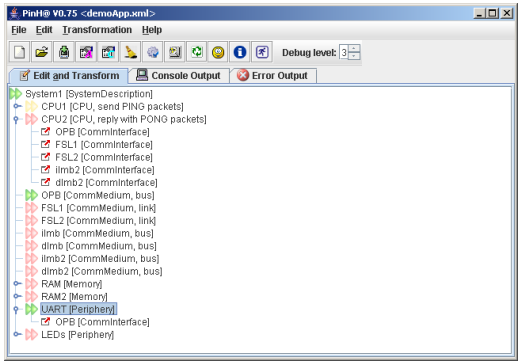
Security and Resiliency Integration



Configure Component



Interface Integration



Security/Reliability Extension

Ongoing Work

- Continuous development – Open Source
- Hardware/Software Systems
- Cloud Protection

Publications

- **F. Hategekimana**, J.M. Mbongue, J. H. Pantho and C. Bobda, "**Inheriting Software Security Policies Within Hardware IP Component**," *2018 IEEE 26th Annual International Symposium on Field-Programmable Custom Computing Machines (**FCCM**)*, Boulder, CO, 2018.

- **F. Hategekimana**, T. Whitaker, M. J. H. Pantho and C. Bobda, "**Shielding non-trusted IPs in SoCs**," *2017 27th International Conference on Field Programmable Logic and Applications (**FPL**)*, Ghent, 2017, pp. 1-4.

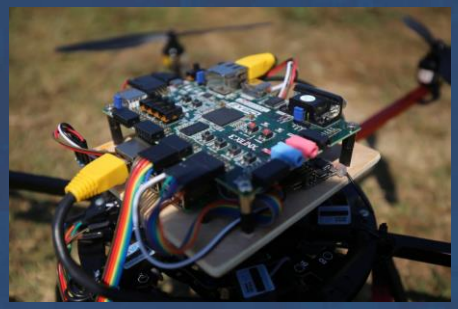
- Festus Hategekimana and Christophe Bobda. 2017. Towards the application of flask security architecture to SoC design: work-in-progress. In Proceedings of the Twelfth IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis Companion (CODES '17).

- F. Hategekimana, T. Whitaker, M. J. H. Pantho and C. Bobda, "Secure Integration of non-trusted IPs in SoCs," 2017 IEEE Asian Hardware-Oriented Security and Trust (AsianHOST), Beijing, 2017

- F. Hategekimana, P. Nardin and C. Bobda, "Hardware/Software Isolation and Protection Architecture for Transparent Security Enforcement in Networked Devices," 2016 IEEE Computer Society Annual Symposium on VLSI (ISVLSI), Pittsburgh, PA, 2016, pp. 140-145.

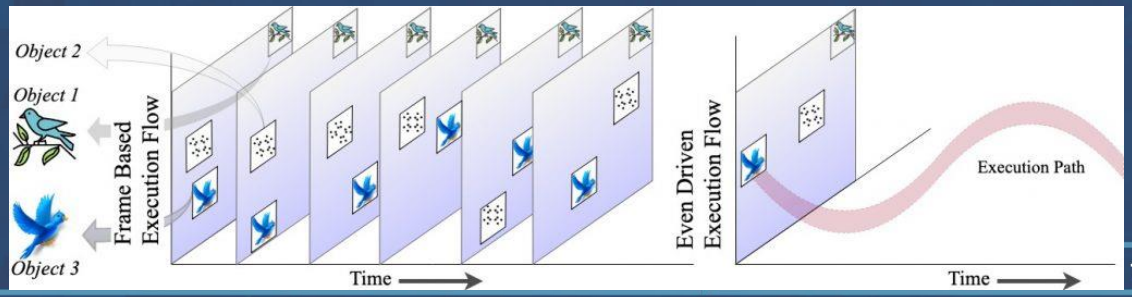
Publications

- Christophe Bobda, Taylor J. L. Whitaker, Charles A. Kamhoua, Kevin A. Kwiat, Laurent Njilla: **Automatic Generation of Hardware Sandboxes for Trojan Mitigation in Systems on Chip (Poster).** FPGA 2017: 289
- Taylor Whitaker and Christophe Bobda. **CAPSL: A Tool for Automatic Generation of Hardware Sandboxes for IP Security (Poster)** in *2015 IEEE International Symposium on Field-Programmable Custom Computing Machines (*FCCM 2017):
- Festus Hategekimana and Christophe Bobda: **Applying The Flask Security Architecture to Secure SoC Design (Poster)** in *2015 IEEE International Symposium on Field-Programmable Custom Computing Machines (*FCCM 2017):
- C. Bobda, T. Whitaker, C. Kamhoua, K. Kwiat, and L. Njilla. **Synthesis of Hardware Sandboxes for Trojan Mitigation in Systems on Chip (poster)** in IEEE International Symposium on Hardware Oriented Security and Trust **(**HOST 2017)
- Joshua Mead, Christophe Bobda, Taylor J. L. Whitaker: **Defeating drone jamming with hardware sandboxing**. IEEE Asian Hardware-Oriented Security and Trust, AsianHOST 2016, Yilan, Taiwan, December 19-20, 2016. : 1-6
- F. Hategekimana, A. Tbatou, C. Bobda, C. Kamhoua and K. Kwiat, "Hardware isolation technique for IRC-based botnets detection," 2015 International Conference on ReConFigurable Computing and FPGAs (ReConFig), Mexico City, 2015, pp. 1-6.

UF | Herbert Wertheim College of Engineering
UNIVERSITY of FLORIDA

smartsystems.ece.ufl.edu



TRANSFORM THE FUTURE