System-on-Chip Security Architecture and CAD Framework for Hardware Patch

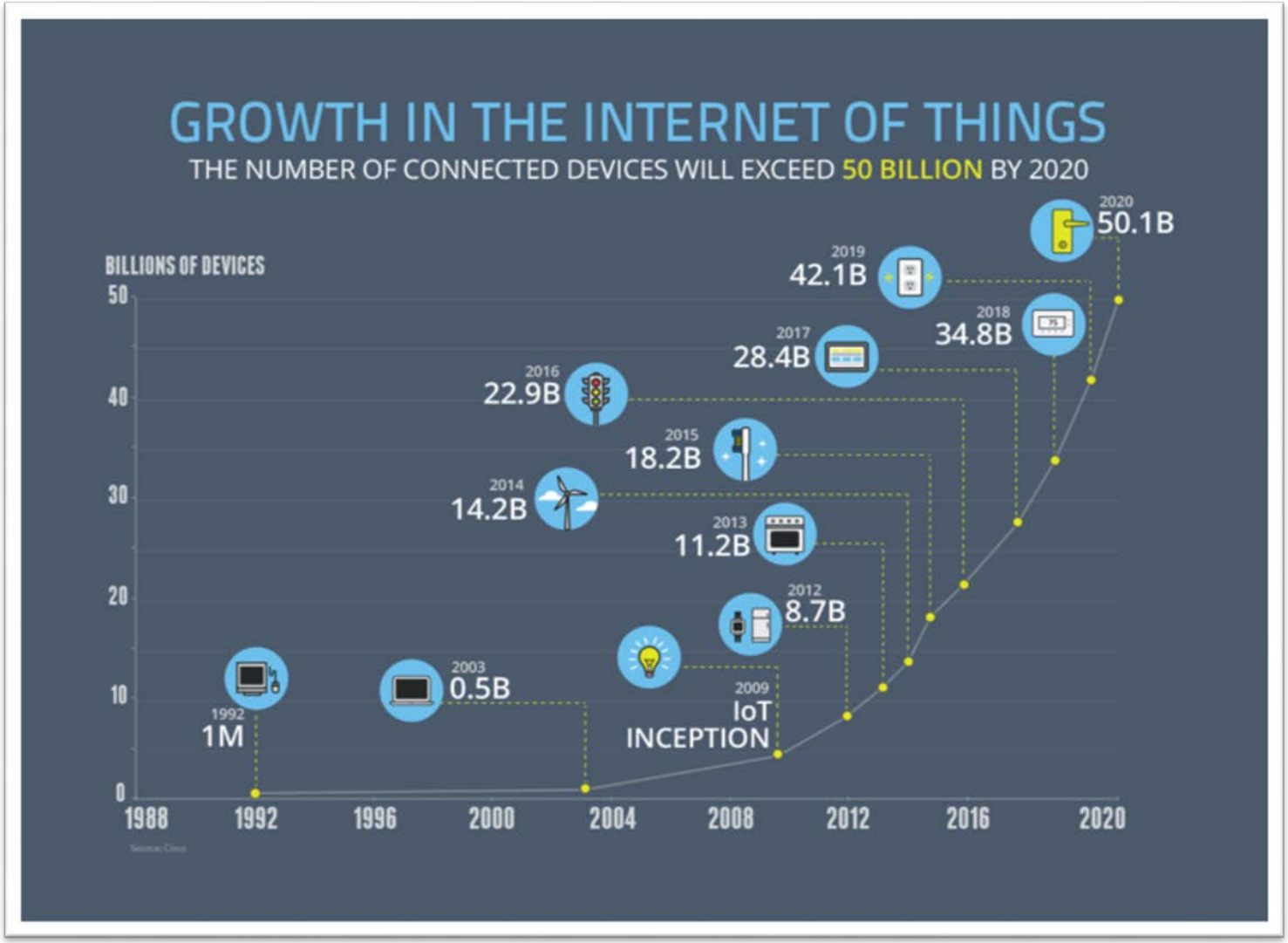
Atul Prasad Deb Nath, Sandip Ray, Abhishek Basak, and Swarup Bhunia

Outline

- SoC Security: Requirements and Challenges
- Background: Security Policies
- Problem Statement: Limitations of Status Quo
- Proposed Solution: Hardware Patching
- Patching Infrastructure and CAD Framework
- Experimental results
- Conclusion

The Rise of Internet of Things (IoT)

- CISCO: 50 billion devices by year 2020
- Intel's Projection: 200 billion
- 26 smart objects for every human being (Intel)



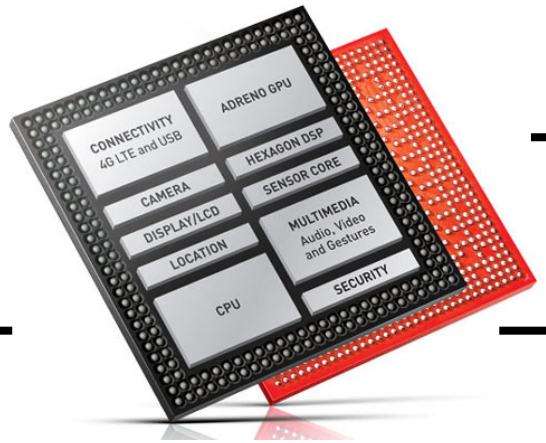
(Source: National Cable & Telecommunications Association)

Security Challenges

- **Modern SoC**

Third-party IPs

Custom security requirements



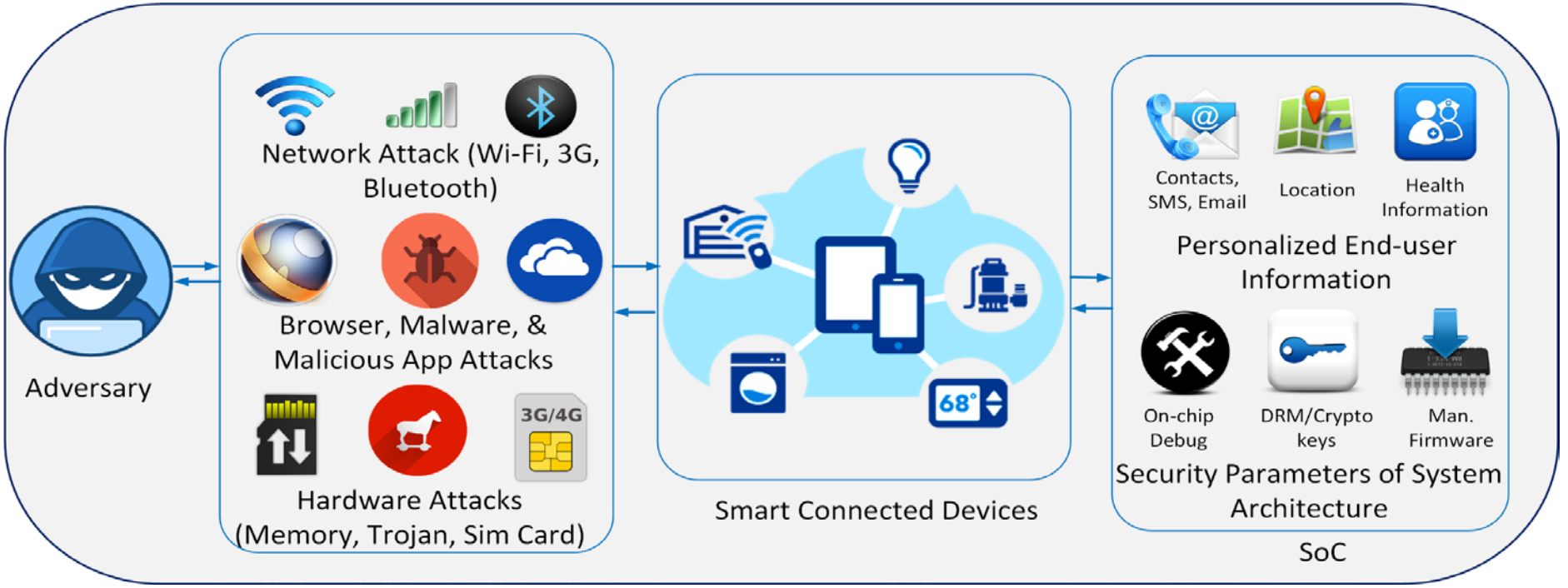
Mitigation of emerging threats

Design constraints

Ensuring security → extremely difficult

Fig. Source: Jargon Buster: The Guide to Understanding Mobile Processors

Generic Threat Model



System-on-Chip Assets:

- **System parameters:**
  - Cryptographic/DRM keys
  - Manufacturer firmware
  - On-chip debug info.
- **End-user information:**
  - Contacts, emails, location, etc.
  - Health information
  - Energy consumption

Research

Background: Security Policies

- Policies governing *confidentiality*, *integrity*, & *availability* of assets
- Policy Categories:
    1) *Access Control*
    2) *Information Flow*
    3) *Liveness*
    4) *Time-of-Check Time-of-Use (ToC-ToU)*
- Security policies map to design features/constraints
    - Used by IP designers, SoC integrators

Ex. 1 – *During boot, data transmitted by crypto-engine cannot be observed by any IP in the SoC other than its intended target (Confidentiality)*

Ex. 2 – *A secure key container can be updated during silicon validation, but not after production (Integrity)*

State of the Practice

Limitations:

- Natural language representation in architecture documents
  - Sprinkled over the IPs of SoC
  - Often continuously refined during SoC integration
  - No systematic method

Existing Constraints:

- No in-field configurability or "**Patchability"**
- Tight boundary of energy and performance profiles
- Software or firmware implementation:
  - Overhead issues : unsuitable for IoT and automotive applications
  - Difficulty in aggressive in-field threat mitigation

Proposed Solution

Hardware Patchable Policies:

- Security policies implemented as a sequence of "commands"
- Centralized Security Policy Engine
- Communicates with IP blocks via standardized interface

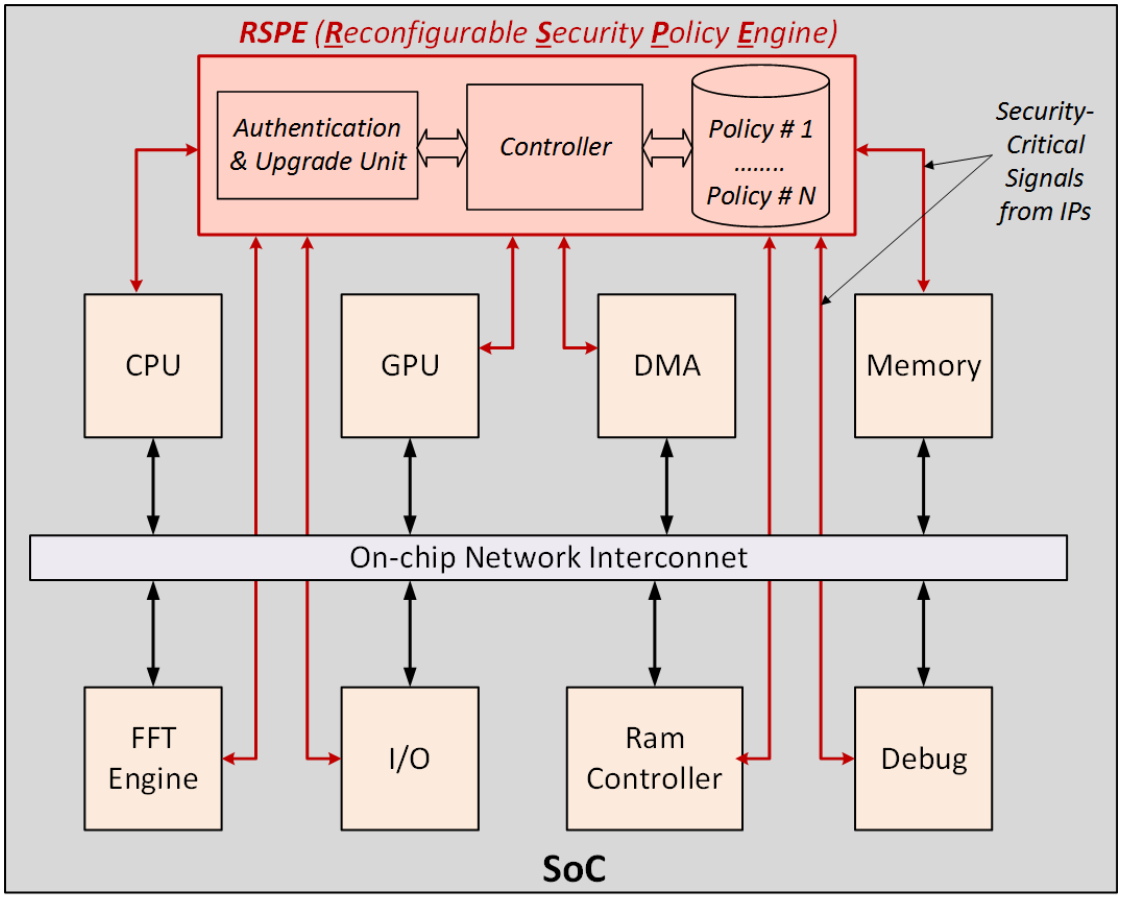


Fig. Generic SoC Architecture with Proposed RSPE (Reconfigurable Security Policy Engine).

Software Flow

Hardware Patchable Policies:

- Policy generation based on emerging threats
- Implementation through reconfigurable policy engine

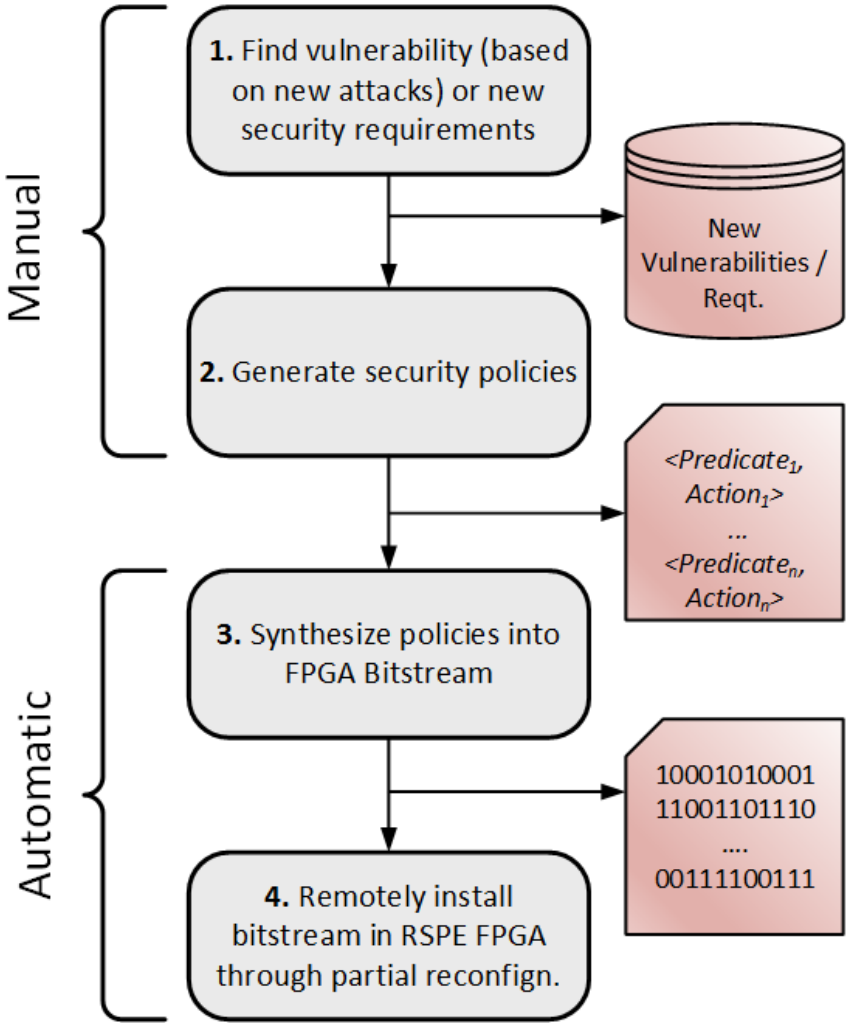


Fig. Software Flow for Security Policy Implementation in Proposed Architecture.

Hardware Patching Infrastructure

- *Reconfigurable Security Policy Engine (RSPE)*
- Implements and upgrades security policies via *Hardware Patching*.

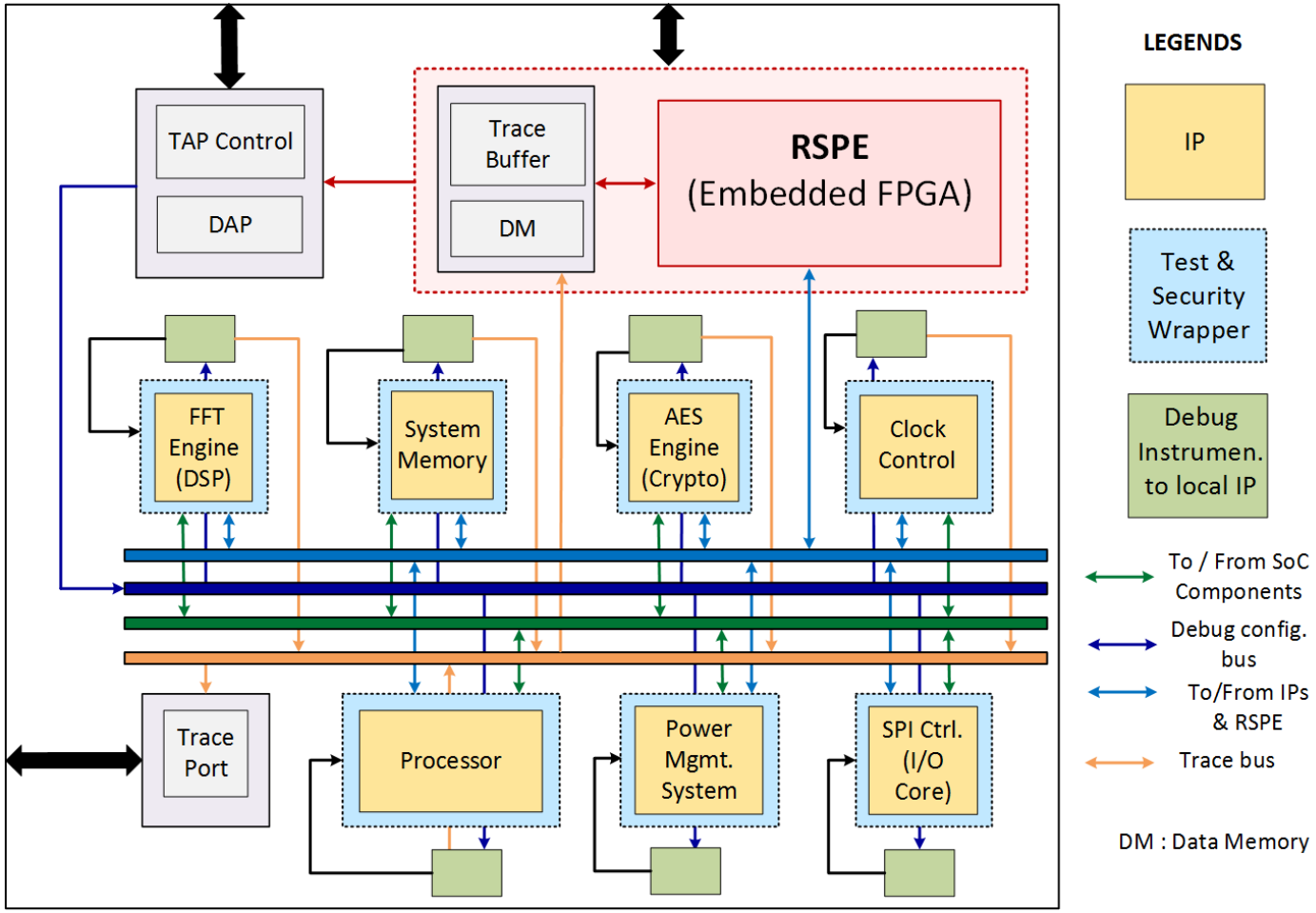


Fig. Proposed RSPE Architecture.

Hardware Patching Infrastructure

- *Smart Security Wrappers*
- Transfer security relevant events between various IPs

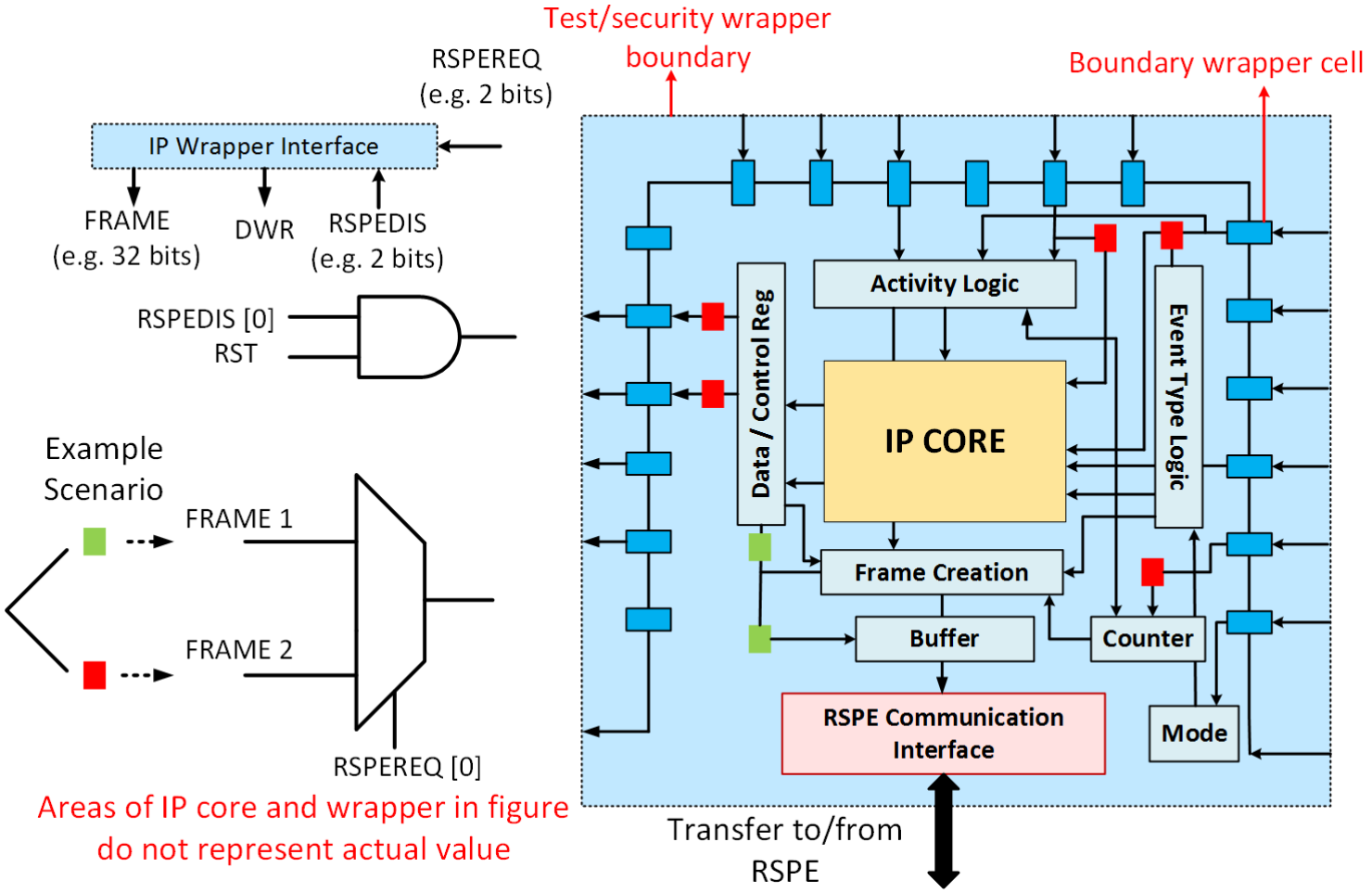


Fig. Standardized Security Wrappers.

Hardware Patching Infrastructure

- *Design-for-Debug (DfD) Infrastructure*
- Enhances controllability and observability over required signals

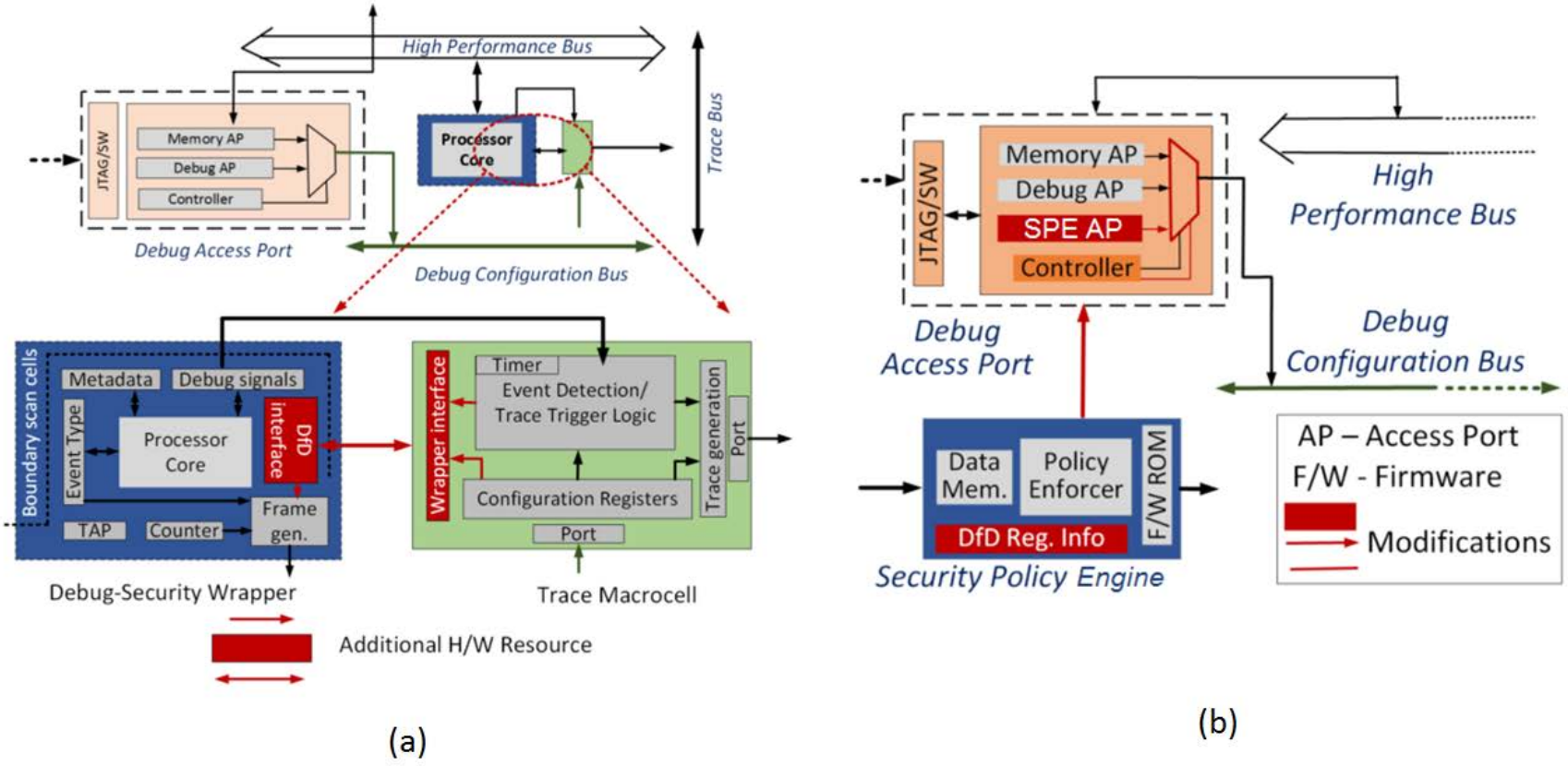


Fig. (a) Architecture for Interfacing DfD with IP Security Wrapper (b) Interfacing Security Policy Engine with On-chip Debug.

CAD Framework

- Systematic approach to synthesize policies into FPGA based Reconfigurable Security Policy Engine

Key Features:

- Amenable for automatic synthesis of arbitrary policies
- 3-tuple format: <timing, predicate, action>

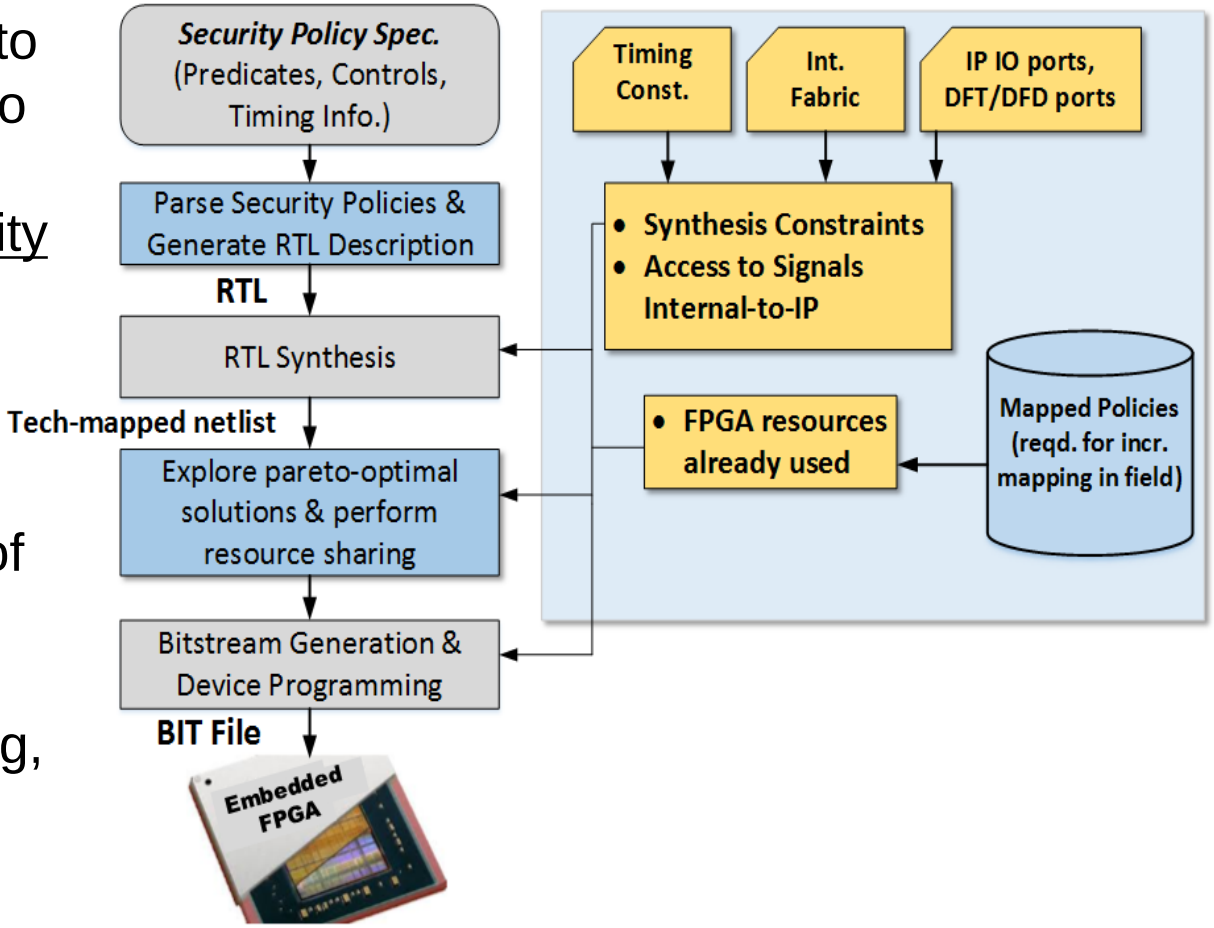


Fig. Mapping Diverse Security Policies on Embedded FPGA-based RSPE

Representative SoC Security Policies

Policy #	Predicate Part	Action Part	IPs Involved
1	User mode & (Mem RD/WR Req. by User — Mem RD/WR Req. by IP A — ...)	RD/WR Addr. within specified range	DLX µP & any other IP with access to system memory
2	Supervisor mode & (Mem RD Req. by User — Mem RD/WR Req. by IP A — ...)	RD Addr. within shared memory range & No WR	DLX µP & any other IP with access to system memory
3	Debug mode & (Trace cells busy — power mgmt. module busy)	No update in power control firmware & no changes in SPI controller Config. Reg	Power mgmt. module & SPI controller
4	!(Supervisor mode) & (Inst. Mem Update Req. through test access port or SPI controller)	No update of Inst. Mem. allowed	DLX µP
5	Active Crypto mode	No interrupt or Memory Access Req. from the DLX core or any IP is allowed	Crypto module, processor and other IPs access to processor

Illustrative Use Case Scenario

I/O Non-Interference Policy:

When CPU is executing in high security mode, I/O devices on SoC platform cannot access protected data memory

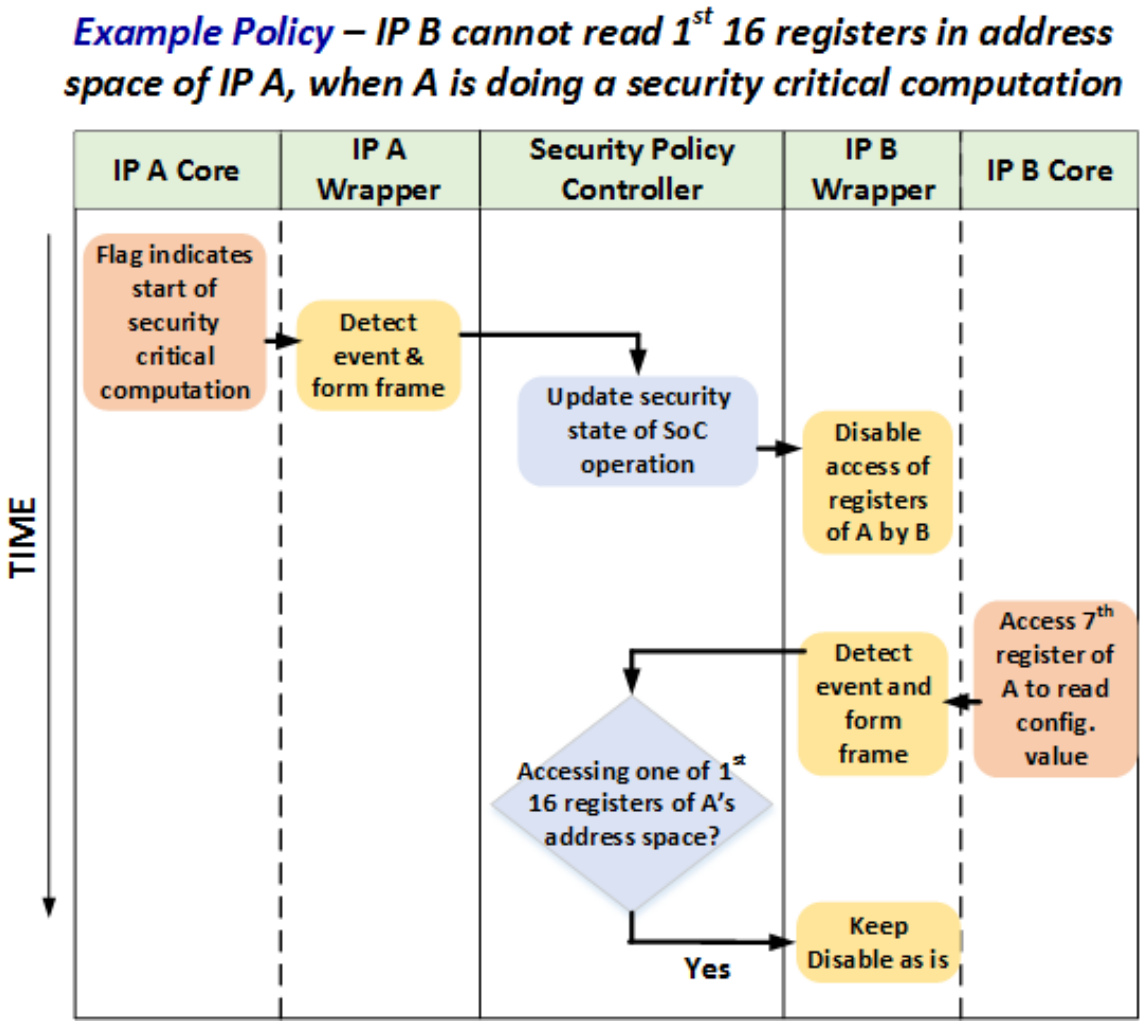


Fig. Implementation of Representative Security Policy

Results Analysis

- **Estimation of Arbitrary Security Policies**
  - Observable signals : Predicate tuples
  - Controllable signals : Action tuples
  - DfD Integration demonstrates superior performance

Tuple Type	Test Wrappers	Security Wrappers	Design-for-debug Infrastructure
2P, 1A	570	490046760	2987015850
4P, 1A	14535	7.91E+13	1.59E+15
8P, 1A	377910	1.75E+23	3.89E+25
8P, 2A	377910	4.42E+25	1.81E+28

Table. Estimation of Arbitrary Number Security Policies in Different Phases of Design

Results Analysis

- **Energy and Latency:**
  - FPGA-based design vs MCU-based Design
  - FPGA-based design:
    - 5.02 times more energy efficient
    - 5.5 times faster

	Die Area (μm^2)	Clock Freq. (MHz)	Cycle Count (10 Policies)	Total Latency (µs)	Dynamic Power (mW)	Static Power (mW)	Total Energy (nJ)
DLX µP	0.724	203	210	1.04	14.27	63.48	80.86
FPGA	1.06	138	26	0.189	64.9	20.43	16.13
Ratio	0.68	1.47	8.07	5.49	0.22	3.11	5.02

Table. Area, Performance, Power, And Energy Values For DLX uP Core And FPGA Based RSPE Module

Results Analysis

- **Area Overhead Comparison**
  - FPGA area is 0.68 times higher than MCU area
  - Total area overhead is less than 5%

SoC	Original Area (μm^2)	µC SPE Overhead (%)	FPGA RSPE Overhead (%)
SoC Model	13.1×10^6	21.7	30.74
Apple A6 (APL0598)	96.71×10^6	2.92	4.26
Qualcomm Snapdragon 800	118.3×10^6	2.39	3.49

Table. Comparison of Area Overhead for Entire SoC

Results Analysis

- **Execution Results**
  - Each policy executed via isolated testbench

Security Policy No.	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10
Energy (nJ)	1.86	1.84	1.85	1.87	1.86	1.83	1.85	1.85	1.87	1 .87
Latency (ns)	21.74	14.48	7.24	21.74	14.48	21.74	14.48	14.48	21.74	7.24
Resource (ALMs)	5465	4065	3260	5465	4065	4065	5465	5465	5465	3260

Table. Results For Execution of Each Policy in FPGA-based RSPE

Conclusion

- We presented ***Patchable Hardware*** for emerging applications:
  - Hardware patchable security architecture
  - Systematic CAD framework

- Distinct features of the design:
  - In-field configurability of diverse arbitrary policies
  - Low area and power overhead : suitable for IoT and automotive applications

- Future work:
  - Evaluation of architecture on industrial SoC models

Q & A