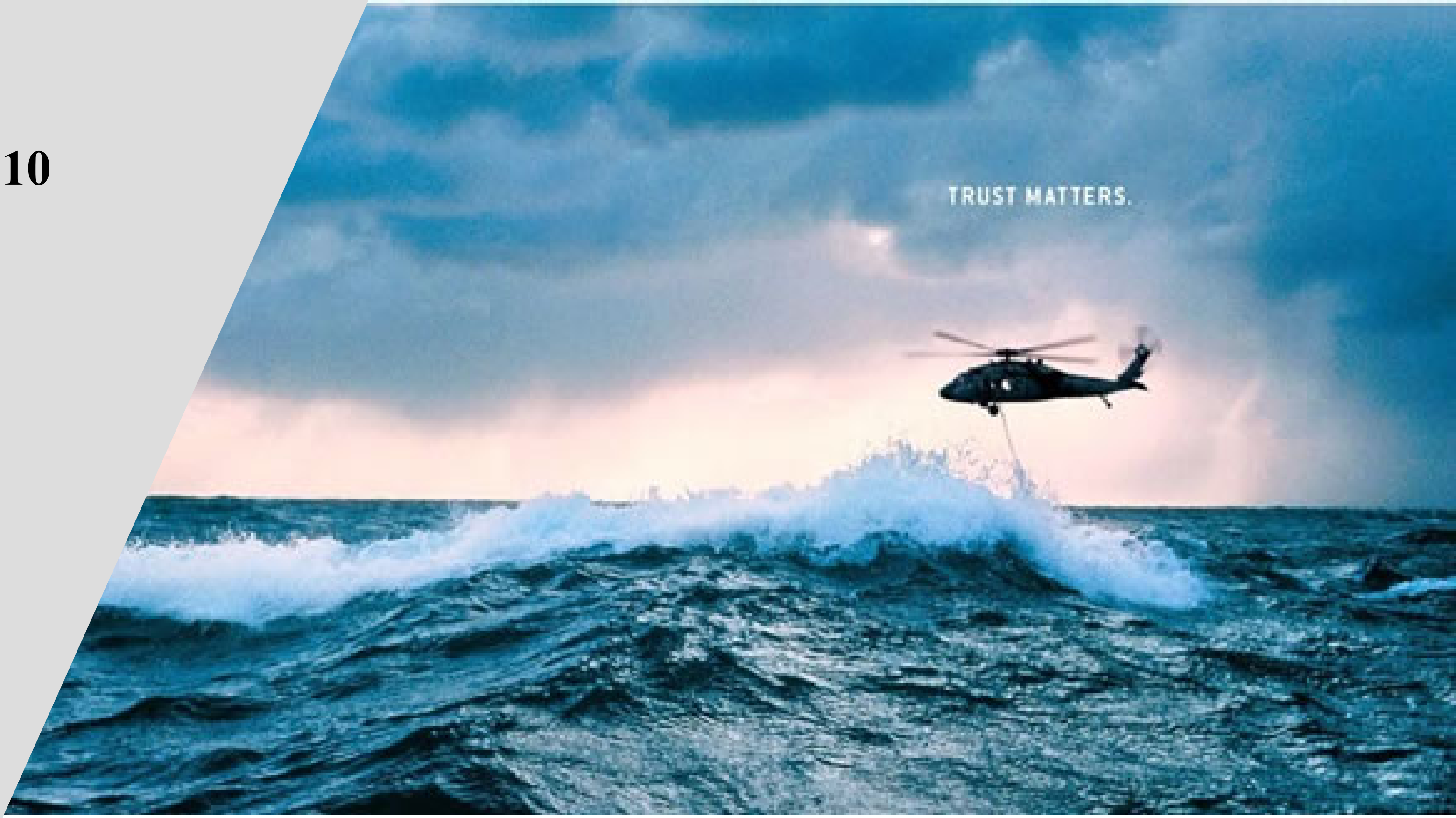
TEITP 2010



User and Evaluator Expectations for Trusted Extensions

David Hardin
Rockwell Collins Advanced Technology Center
Cedar Rapids, Iowa
USA

Rockwell Collins

Outline

- What Does a Security Evaluation Entail?
  - Example: AAMP7 MILS Evaluation
- User and Evaluator Expectations for Formal Verification Tools
- Can Trusted Extensions Help?
- Issues
- Discussion

Security Evaluations in the USA

- Common Criteria for Information Technology Security Evaluation
  - Internationally recognized standard
  - Provides a common language for vendors and consumers
    - Evaluation Assurance Levels (EALs)

- National Information Assurance Partnership (NIAP)
  - US Common Criteria Certification Authority

- National Security Agency (NSA)
  - Evaluation Authority for formal methods work for 'high assurance' certifications in the USA

Common Criteria Evaluation Assurance Levels

- EAL 1 – functionally tested
- EAL 2 – structurally tested
- EAL 3 – methodically tested and checked
- EAL 4 – methodically designed, tested, and reviewed
- EAL 5 – semiformally designed and tested
- EAL 6 – semiformally verified design and tested
- EAL 7 – formally verified design and tested

The "EAL scale" is basically logarithmic in evaluation difficulty – like the Category scale for hurricanes ;-)

Degrees of Formality

- Informal
  - Written as prose in natural language
- Semiformal
  - Specifications written in a restricted syntax language, internally consistent. Correspondence demonstration requires a structured approach to analysis
- Formal
  - Written in a notation based upon well-established mathematical concepts

Protection Profiles and Security Targets

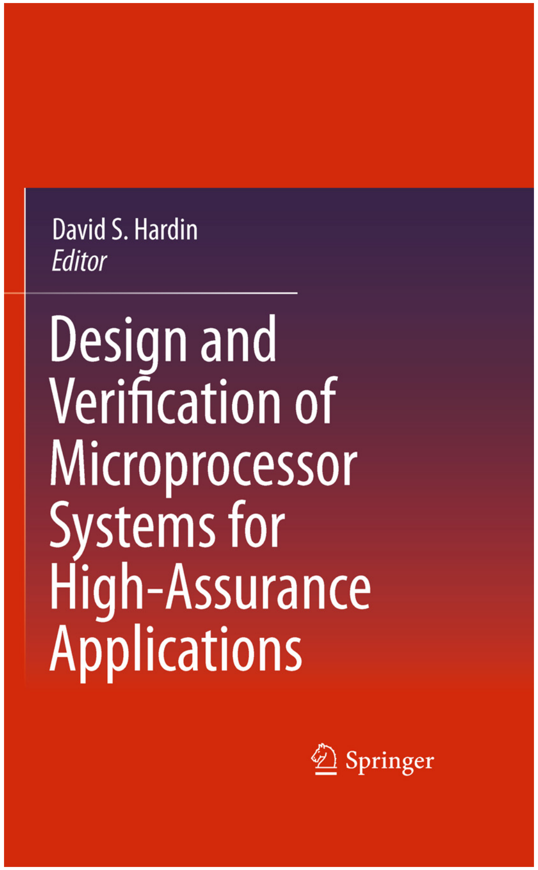
- These documents tailor the Common Criteria requirements
  - Requirements profiles
- Protection Profiles (PP) specifies requirement profiles for a class of applications
  - Separation Kernel Protection Profile
  - Optional artifact
- Security Target applies to a specific application
  - Each certification must have a security target

Formal Methods and the CC

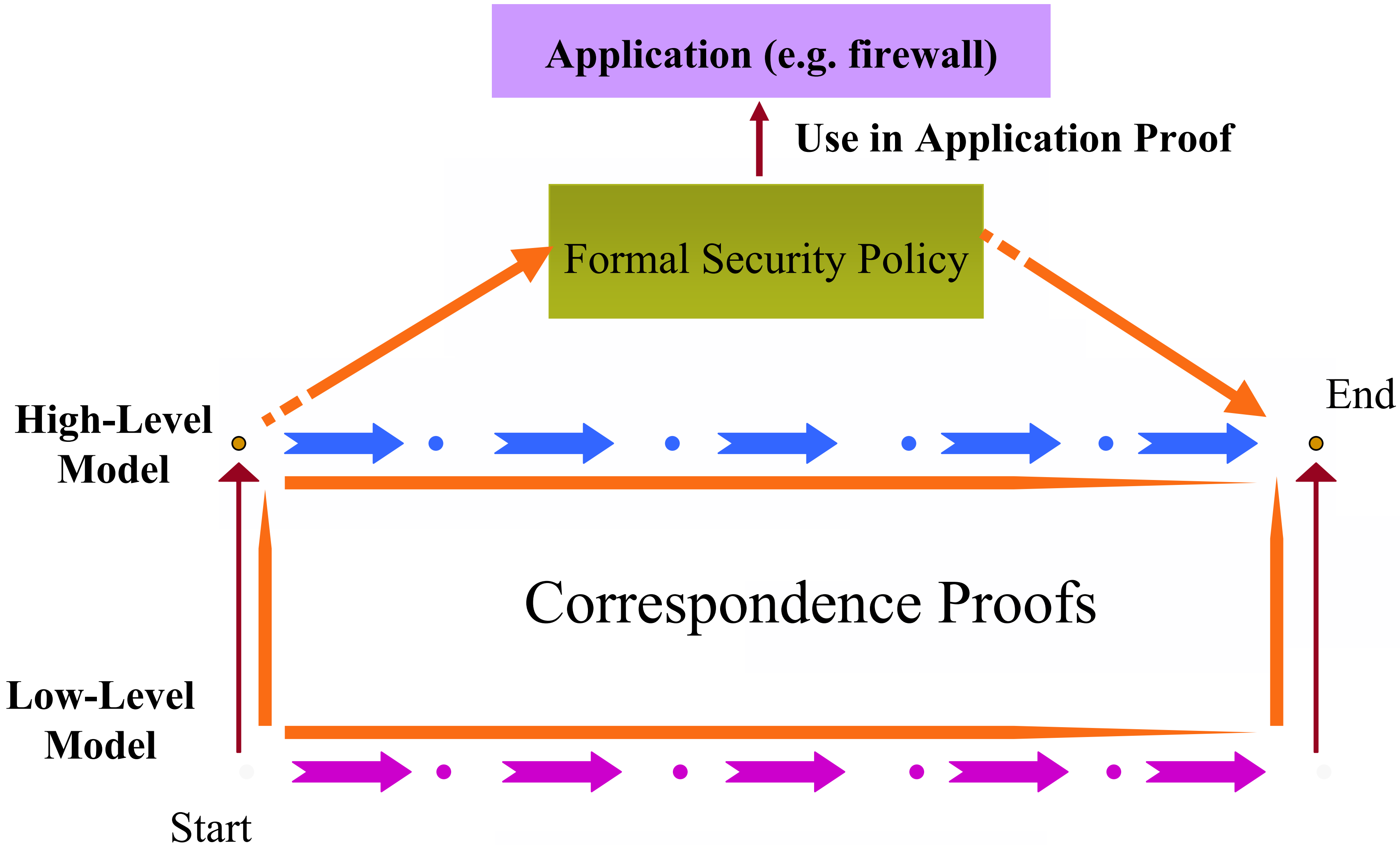
- Formal methods analysis satisfies the following CC sections
  - ADV_FSP (Functional Specification)
  - ADV_HLD (High-Level Design)
  - ADV_LLD (Low-Level Design)
  - ADV_RCR (Representation Correspondence)
  - ADV_SPM (Security Policy Modeling)
- Fundamental properties of the system are proven
- System may be modeled in a formal language
  - Multiple models with a decreasing degree of abstraction
  - Correspondence between levels rigorously proven.
- Properties proven on each model
- **Most detailed model shown to correspond to implementation by code-to-spec review**

A Formal Modeling Approach: Calculus of Indices

- Computing System is Modeled **Functionally**
  - No Side-Effects!
  - Step Function (Next)
  - Multiple levels of abstraction
  - Lowest level (for this work) typically a microcode interpreter
- Information is Modeled Indirectly, in terms of Location (indices)
  - Not "What the Information is", But "Where the Information is"
- Dynamic Process involving the movement of information (information flow) from one location to another
  - Associated with some action in the system
  - Carried out by functions
- This philosophy has been codified in a formal theory called "The Calculus of Indices"
  - Documented in a chapter by D.A. Greve, Information Security Modeling and Analysis, in the book *Design and Verification of Microprocessor Systems for High-Assurance Applications* (Springer 2010)



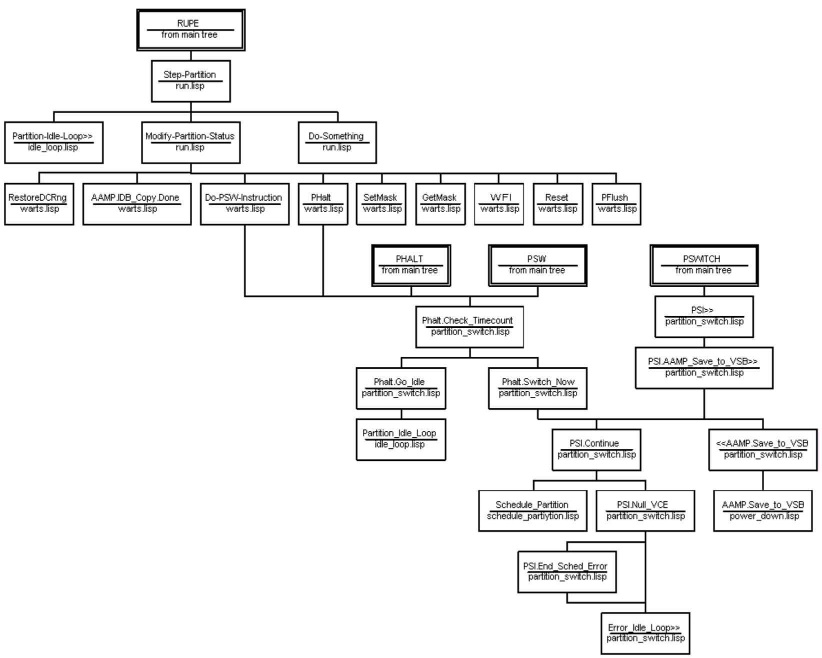
Assurance Architecture



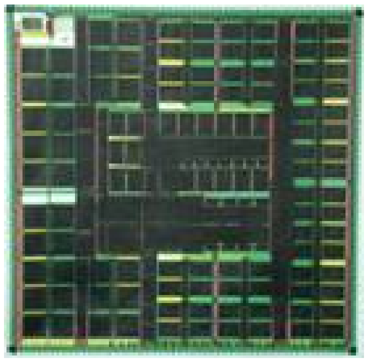
Validating the Low-Level Model

Q: Is the model the right model?

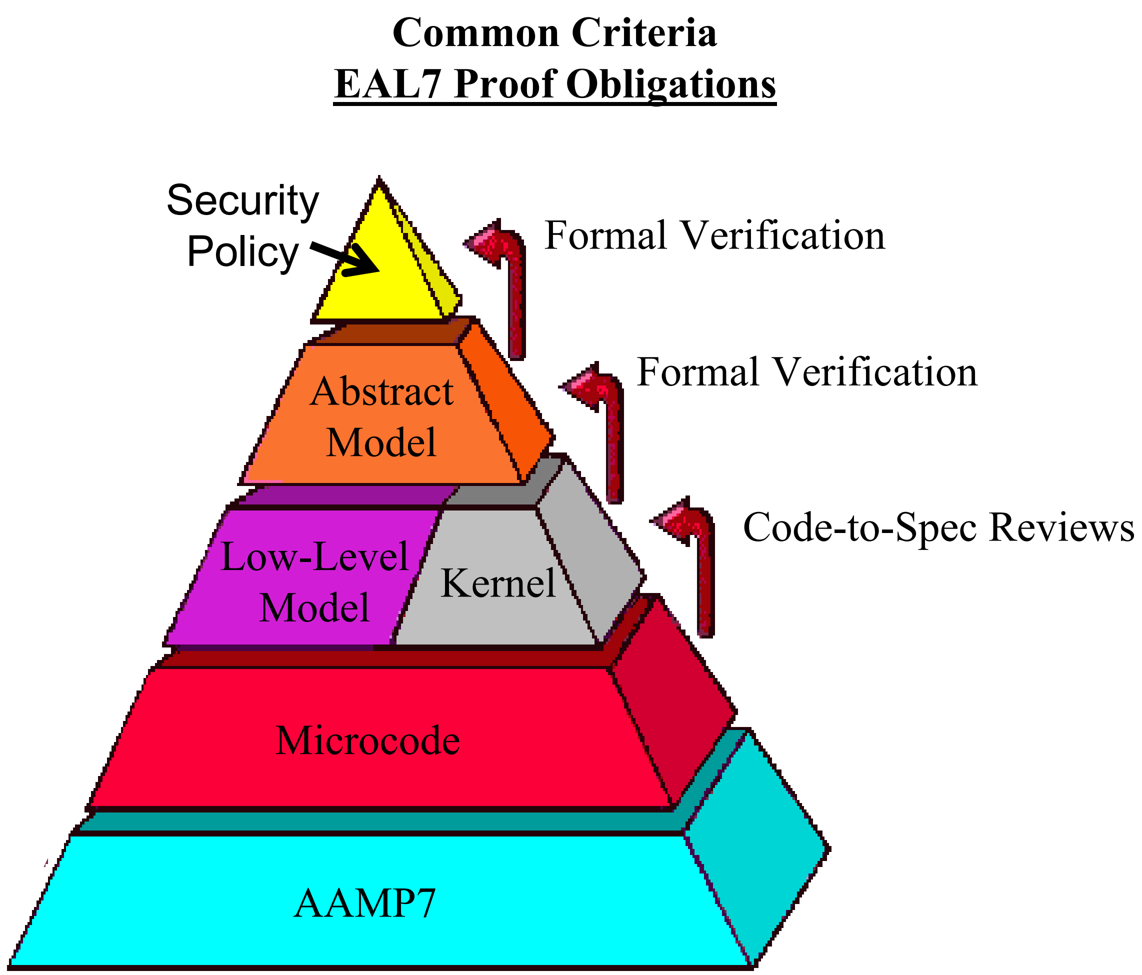
A: The 'Code-to-Spec' review with NSA evaluators determines that the lowest-level model accurately depicts the system's true behavior



?
=

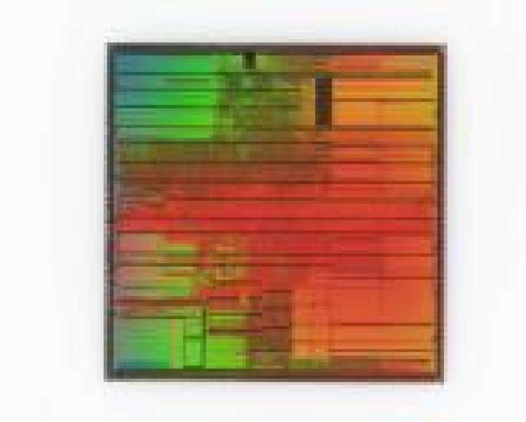


AAMP7 MILS Verification



AAMP7 Microprocessor

- Utilized in a number of Rockwell Collins navigation and communications products
- High Code Density (2:1 Over CISC, 4:1 Over RISC)
- Low Power Consumption
- Screened for full military temp range (-55 C to +125 C)
- Design artifacts owned by Rockwell Collins
- Architecturally-defined threads, executive/user modes, exception handling
- Intrinsic Partitioning
  - Allows multiple independent applications to execute concurrently on the same CPU
  - “Separation Kernel in Hardware”
  - Very low latency
  - Ripe target for formal verification



AAMP7 Design for Verification Characteristics

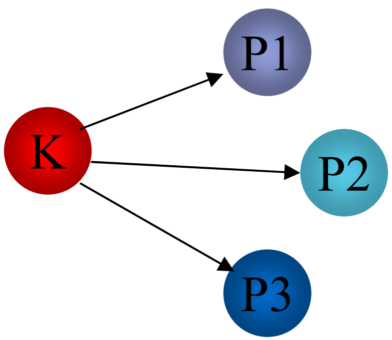
- AAMP7 partitioning logic is (relatively) localized in the design
- AAMP7 partitions are controlled by "Trusted mode" microcode
  - No software in separation kernel
  - Non-trusted mode microcode cannot affect partitioning data structures
- Simple range-based memory protection
  - Physical memory model
  - Partitions can define up to eight memory regions
    - code/data, read/write attributes
- Strict Time partitioning
  - Partitions have fixed time allocations
  - Partitions execute in round-robin fashion according to a partition schedule defined by the partitioning data structures
- Partition-aware interrupts
  - Interrupts for non-current partition are pended for delivery when that partition becomes active

The ACL2 Theorem Prover

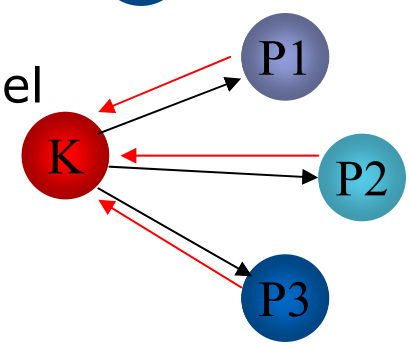
- A system for the development of machine-checked proofs for theorems expressed in a logic that is an applicative subset of Common Lisp
  – Applicative subset == no side effects
- Developed by Kaufmann and Moore at the University of Texas and Austin
- *Since ACL2 models are also applicative Common Lisp programs, they can be executed*
- First-order logic
- Proofs are guided by the introduction and proof of lemmas that guide the theorem prover’s simplification strategies
- Key evaluators were familiar with ACL2

The GWV Formal Security Policy

- GWV security policy developed for AAMP7 verification
  - Named after its authors: Greve (RCI), Wilding (RCI), and vanFleet (NSA)
- GWV validated by use in proof of firewall system exhibiting desired infiltration, exfiltration, mediation properties
- GWV only applicable to a narrow class of systems
  - Strict temporal partitioning
  - Kernel state cannot be influenced by execution of code within partitions

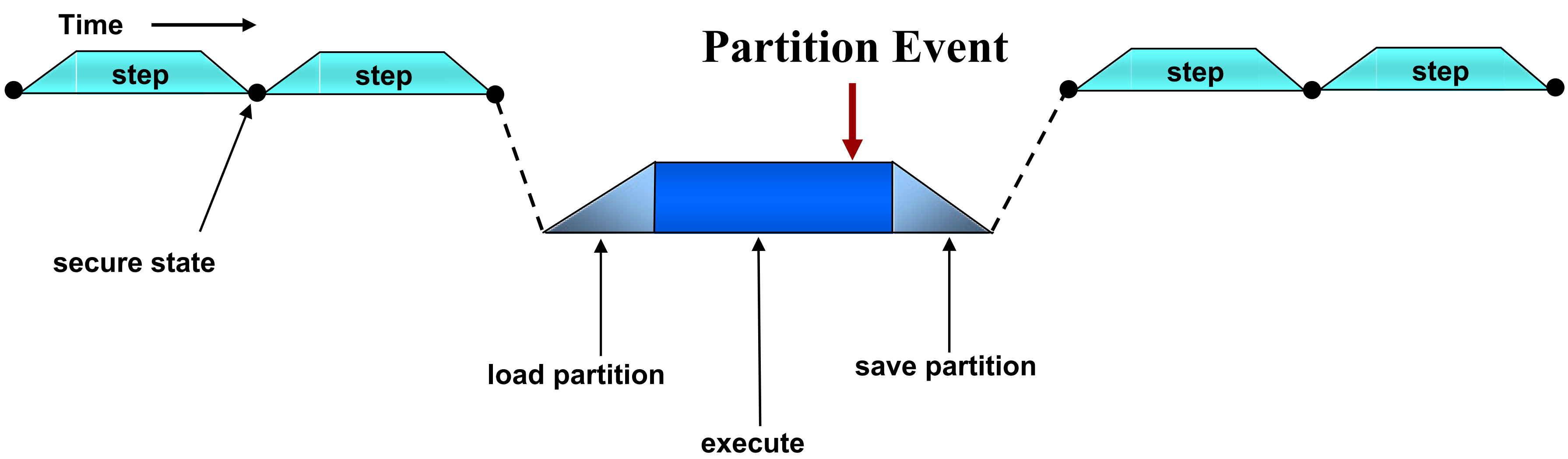


- Later generalized for a wider range of systems
  - GWVr2, used to verify a commercial RTOS kernel



Partition Execution Model

- Begins with the Loading of the Current Partition
- Ends with the Saving of the Current Partition State
  - And the updating of the value of "current partition"



GWV Separation Theorem

"Direct Interaction Allowed"

```
(defthm gwv
  (let ((dia-segs (intersection (dia seg) (get-segs (current st1)))))
    (implies
      (and
        (equal (select-list dia-segs st1)
               (select-list dia-segs st2))
        (equal (current st1)
               (current st2))
        (equal (select seg st1)
               (select seg st2)))
      (equal (select seg (next st1))
             (select seg (next st2))))))
```

Index

Partition Step

Code-to-Spec Review Details

- Goal: Validation of Low-Level Model
  - No “Proof of Correctness”
  - Must be done informally

- The Code-to-Spec Review
  - Inspection to determine whether the “code” implements the “specification”
  - Requires some understanding of both
  - Implementers have a “meeting of the minds” with evaluators

Rockwell Collins

Code-to-Spec Review Sample

Microcode

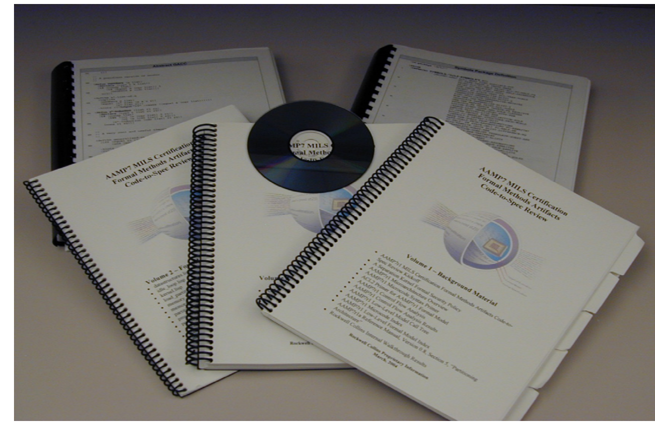
Formal Model

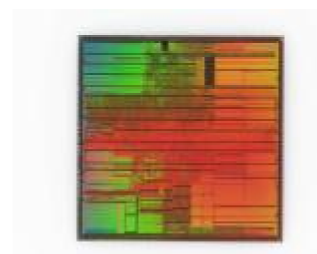
```
;=== ADDR: 052F

   (st. ie = nil)
   (Tx = (read32 (vce_reg st) (VCE.VM_Number)))

;=== ADDR: 0530

   (st. Partition = Tx)

;=== ADDR: 0531

   (TimeCount = (read32 (vce_reg st) (VCE.TimeCount)))

;=== ADDR: 0532

   (PSL[0]= TimeCount st)
```

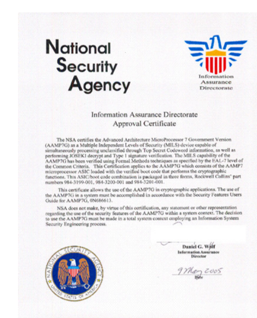
```
;-------------------------------------------------------------------------
;=== ADDR: 052F
A]
  CONT ;
H] clear InterruptEnable, read VM number
  IE=0                          \
                               T=BADDR.READ32(T) ;
L] hold VM number (a.k.a. partition number) in T
                               \
                               T=T ;
;-------------------------------------------------------------------------
;=== ADDR: 0530
A]
  CONT ;
H] load VM number into MSQ partition register
  P=T                           \
                               T=T ;
L] unused
                               \
                               T=T ;
;-------------------------------------------------------------------------
;=== ADDR: 0531
A]
  CONT ;
H] locate TimeCount in VCE
  R=VCE.TimeCount  W=RFB(VCE_REG)    \
                               T=R+W ;
L] read TimeCount
                               \
                               T=BADDR.READ32(T) ;
```

AAMP7 Verification Summary

- Developed formal description of separation for uniprocessor, multipartition system
- Modeled trusted AAMP7 microcode
- Constructed machine-checked proof of separation on the AAMP7 model
  - ACL2 theorem prover checked
  - Operations on pointer-laden, aliased data
- Model subject of intensive code-to-spec review with AAMP7 microcode
- Satisfied formal methods requirements for AAMP7 - certification awarded in May 2005
  - *AAMP7 was "verified using Formal Methods techniques as specified by the EAL-7 level of the Common Criteria"* and is *"capable of simultaneously processing unclassified through Top Secret Codeword"*







User and Evaluator Expectations, as Embodied by AAMP7 Formal Verification Tools

- Familiar to Key Evaluators
- ACL2 authors are highly regarded for the great care and strict control that they use to maintain and improve the ACL2 codebase
- Significant "service history" over the past 20 years
  - Rockwell Collins maintains key proof results initially developed over 10 years ago
- Freely available from a single, well-known web site
  - Ample documentation
  - Significant suite of regression tests
- ACL2 authors have stepped up the release frequency in recent years so that unofficial patches are not needed to perform leading-edge proofs
  - This means that we can hand the proof scripts to the evaluators, and they can "replay" the proofs using the most current version of ACL2, which they can download themselves

Can Trusted Extensions Help?

- The combination of a general-purpose theorem prover with customized decision procedures has shown to be an effective technique
  - Can “blow away” low-level subgoals that often arise when dealing with very concrete models
  - New decision procedures are arising constantly, with promises of dealing with increasingly complex problems
- Combinations of theorem provers (e.g., the HOL/ACL2 Connection) can be used to solve problems that would be difficult using a single prover
- Verification Time is a key consideration; if a tool exists that can help an industrial developer get the job done faster, there will be significant pressure to use it

Trusted Extensions: Issues to Discuss

- Provenance of an extension
  - Who is developing it?
  - If developed by a student, will it be maintained after the student has graduated? Is it under rigorous version control?
  - Is the extension well-documented?
  - Are evaluators familiar with it? Have they used it?
- Translation into the language of the extension
  - How can this translation be trusted?
- Production of uniform evaluation evidence
  - Proof-producing extensions would help
- Tool "version drift"
  - Tools are developed at different times, and at different rates
  - Extension version 1.0, which works great with Theorem Prover version 2.3, may utterly fail with version 2.4
  - Have observed this phenomenon with the HOL/ACL2 Connection