High-assurance Cryptography

High-Confidence Software and Systems Conference (HCSS '15)

Tom DuBuisson, Trevor Eliott, **Joe Kiniry**, Daniel Wagner, Dan Zimmerman
Galois, Inc.

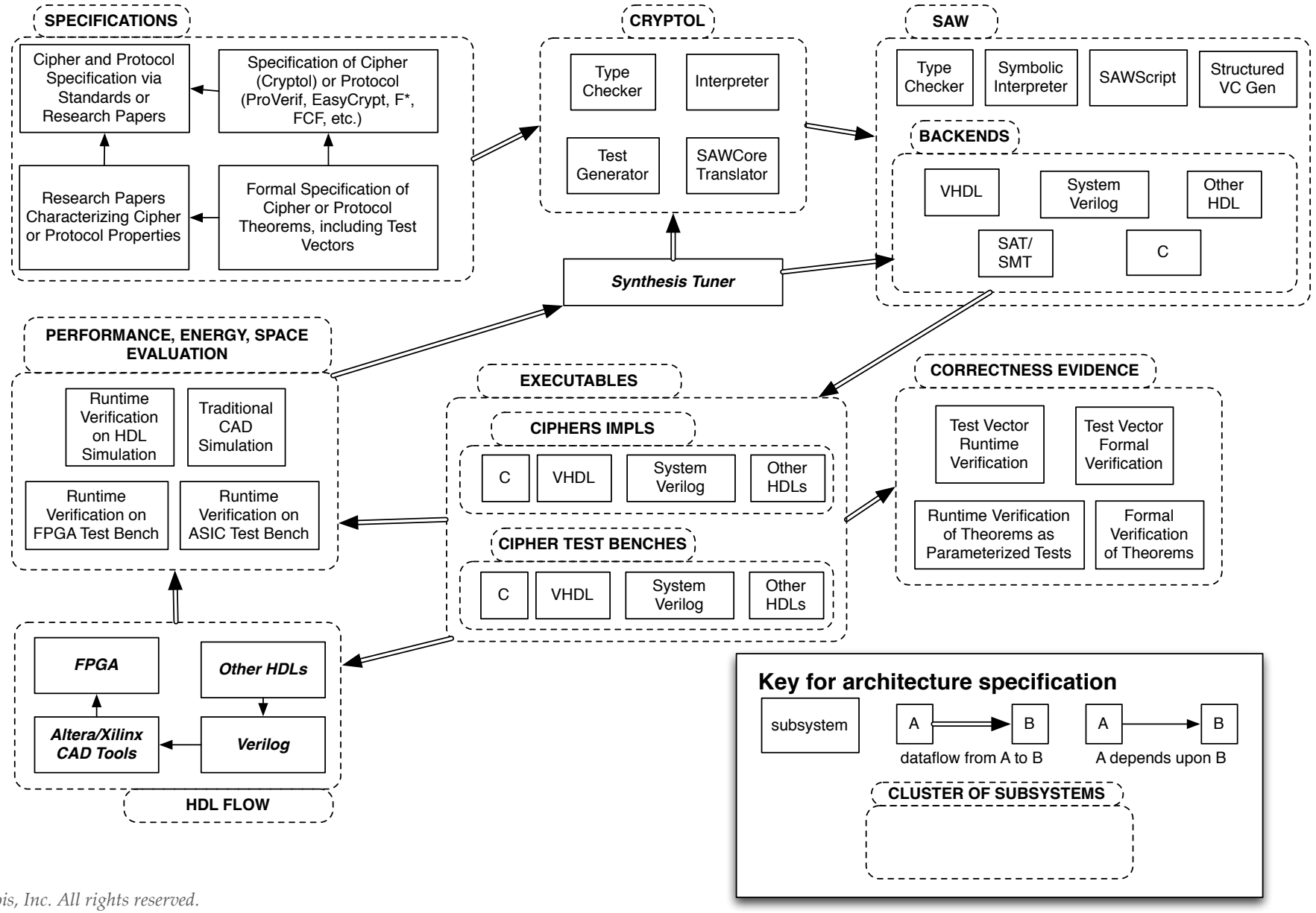
May 2015

Problem Framing

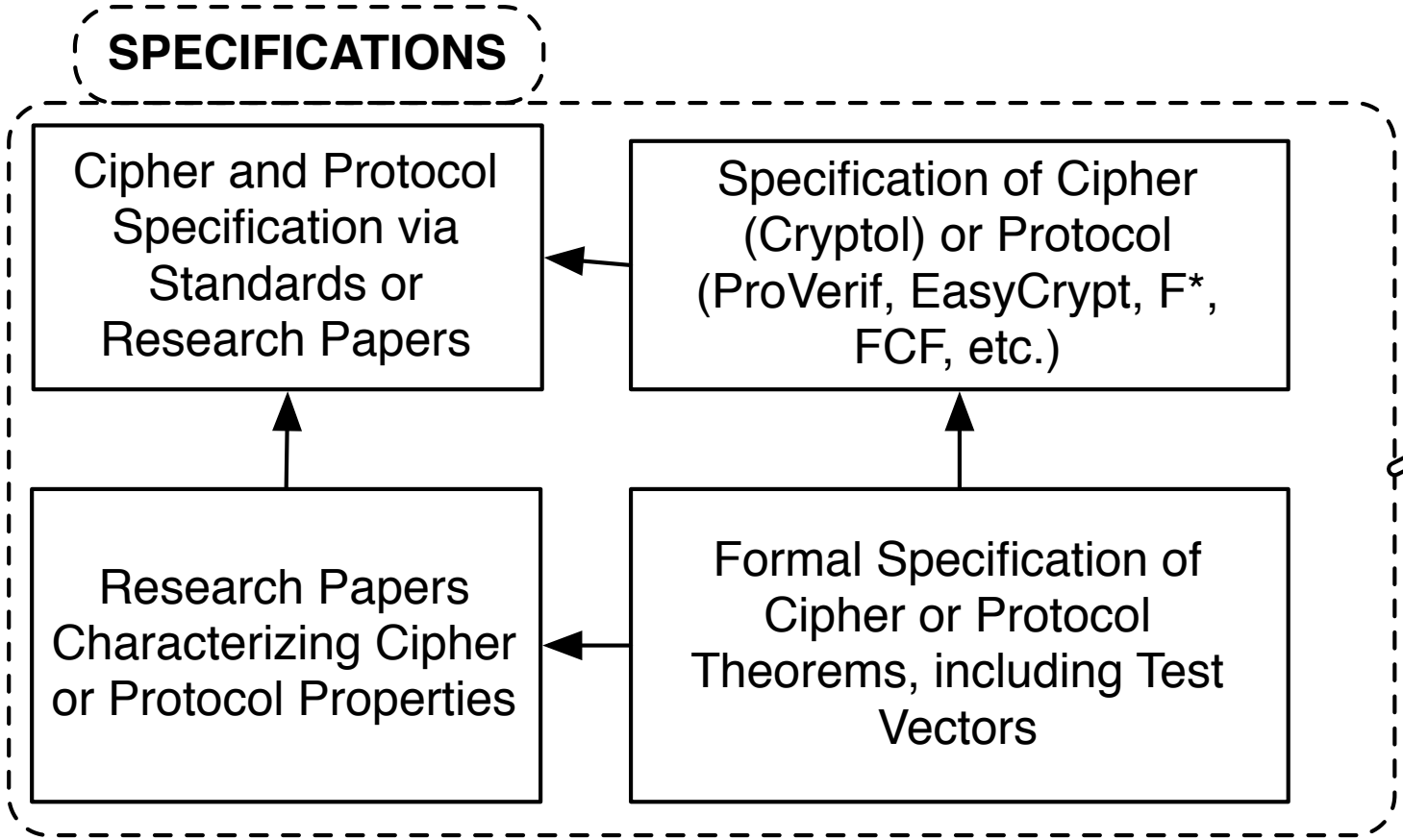
|galois|

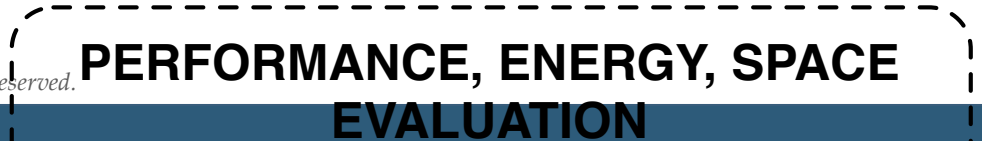
- Galois has performed a high-level audit of every open source crypto library ever released to reflect upon their engineering, correctness and security
  - few are maintainable and very few witness engineering principles, despite the fact that they are FIPS certified
  - none provide an adequate foundation for nationally critical high-assurance systems development
- Galois has been assisting in FIPS certifications of several crypto libraries, thus has intimate knowledge of the CAVP and FIPS certification schemes
- as an offer to the world, we intend to design and develop a rigorously engineering, fully verified (for all appropriate correctness and security properties) cryptographic library and crypto hardware implementation

Galois HACrypto Toolchain



Galois HACrypto: Specs





Literate Cryptol Specs

|galois|

- Cryptol is a DSL for specifying cryptographic algorithms and their properties (aka theorems)
- Literate Cryptol specifications are LaTeX or Markdown documents in the Knuth tradition
- for each algorithms, we formalize every test vector, example, and theorem from the original standards documents and related research papers as properties
- each property is validated using runtime verification (through the use of our symbolic evaluator and automatic test generation) or formally verified via SAT or SMT

Literate Cryptol Example

|galois|

```
% ChaCha20 and Poly1305 for IETF protocols
% Y. Nir (Check Point),  A. Langley (Google Inc),  D. McNamee (Galois, Inc)
% July 28, 2014

## Abstract

This document defines the ChaCha20 stream cipher, as well as the use
of the Poly1305 authenticator, both as stand-alone algorithms, and as
a "combined mode", or Authenticated Encryption with Additional Data
(AEAD) algorithm.

…

The elements in this vector or matrix are 32-bit unsigned integers.

```cryptol
module ChaCha20 where

type ChaChaState = [16][32]
```
```

Literate Cryptol Example

|galois|

````
## The ChaCha Quarter Round

The basic operation of the ChaCha algorithm is the quarter round.  It
operates on four 32-bit unsigned integers, denoted a, b, c, and d.
The operation is as follows:

```cryptol
ChaChaQuarterround : [4][32] -> [4][32]
ChaChaQuarterround [a, b, c, d] = [a'', b'', c'', d''] where
    a' = a + b
    d' = (d ^ a') <<< 16
    c' = c + d'
    b' = (b ^ c') <<< 12
    a'' = a' + b'
    d'' = (d' ^ a'') <<< 8
    c'' = c' + d''
    b'' = (b' ^ c'') <<< 7
```
````

Literate Cryptol Example

|galois|

The ChaCha Quarter Round

The basic operation of the ChaCha algorithm is the quarter round. It operates on four 32-bit unsigned integers, denoted a, b, c, and d. The operation is as follows:

```
ChaChaQuarterround : [4][32] -> [4][32]
ChaChaQuarterround [a, b, c, d] = [a'', b'', c'', d''] where
    a' = a + b
    d' = (d ^ a') <<< 16
    c' = c + d'
    b' = (b ^ c') <<< 12
    a'' = a' + b'
    d'' = (d' ^ a'') <<< 8
    c'' = c' + d''
    b'' = (b' ^ c'') <<< 7
```

Literate Cryptol Example

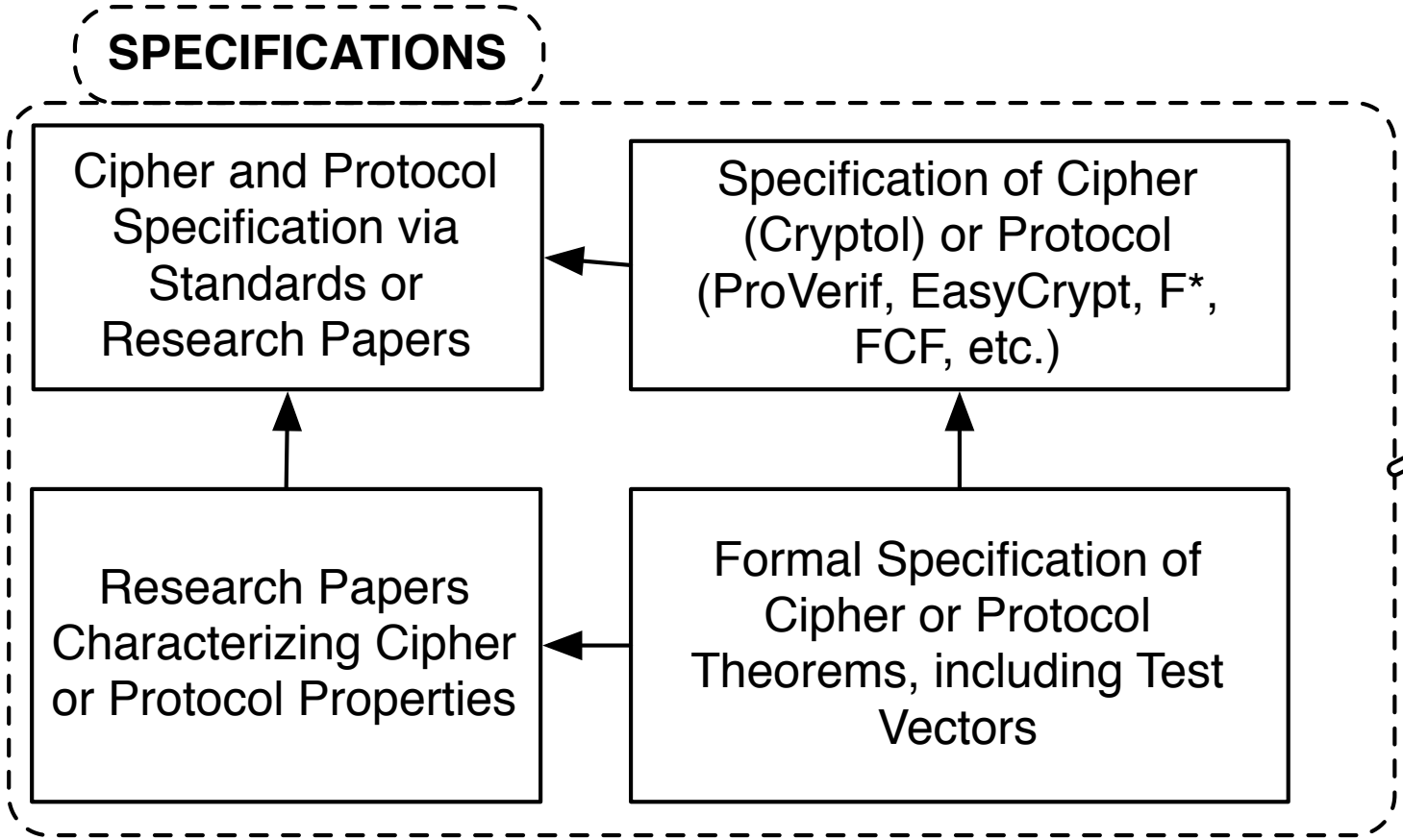
|galois|

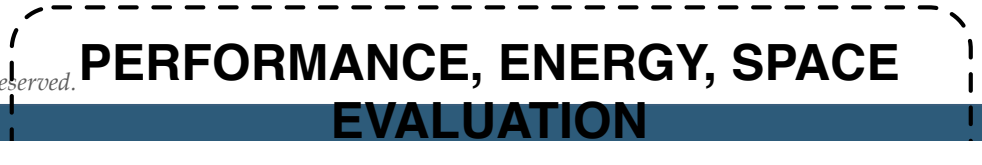
```
property TV1_plaintext_correct = isValid && pt == TV1_plaintext where
    (pt,isValid) = TV1_calculate_plaintext

property decryption_vector_correct =
    TV1_plaintext_correct &&
    TV1_tag_correct &&
    TV1_otk_correct

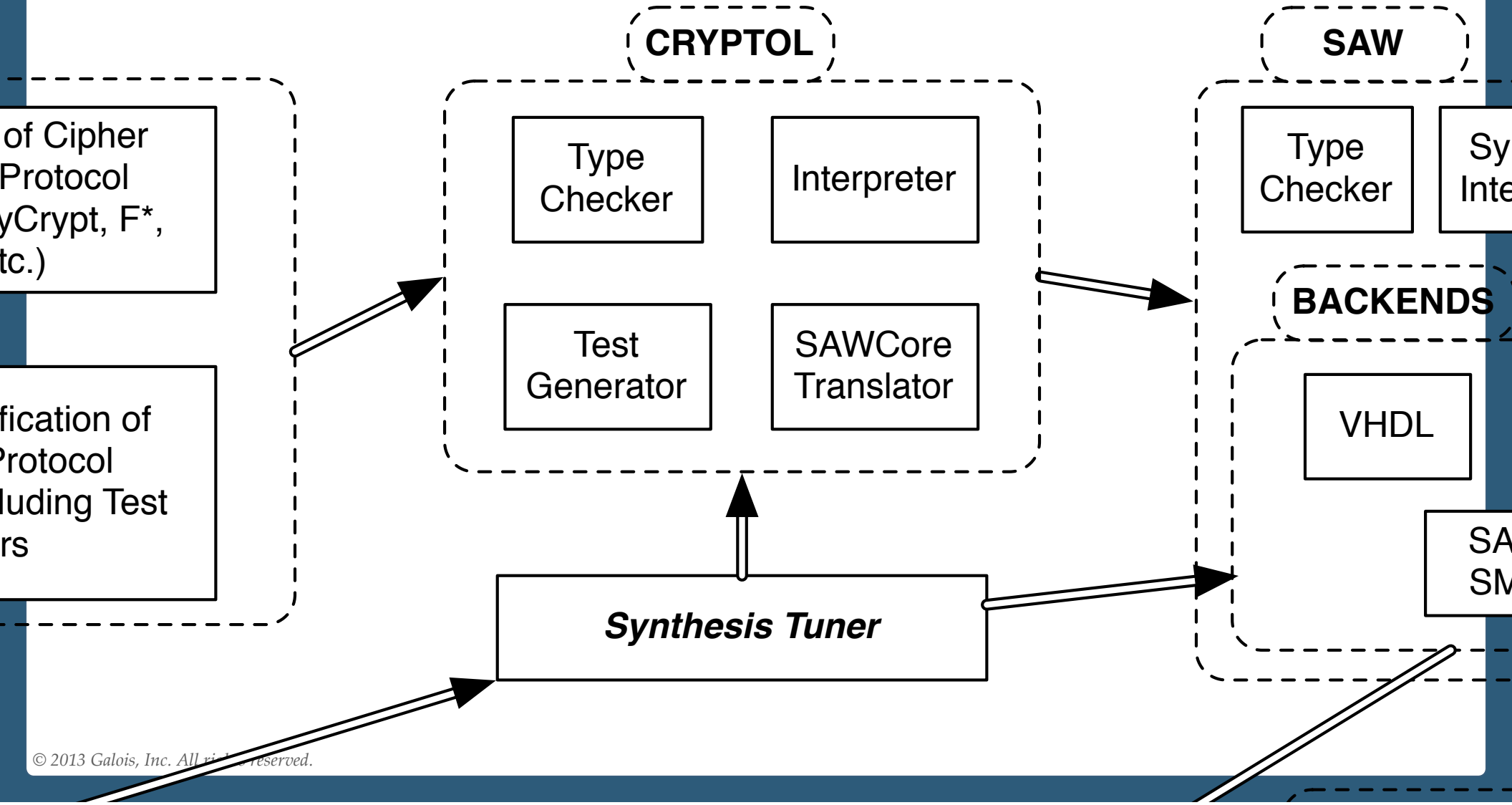

property all_test_vectors_correct =
  all_block_tests_correct &&
  all_enc_tests_correct &&
  all_MAC_tests_correct &&
  all_key_tests_correct &&
  decryption_vector_correct
```

Galois HACrypto: Specs





Galois HACrypto: Cryptol

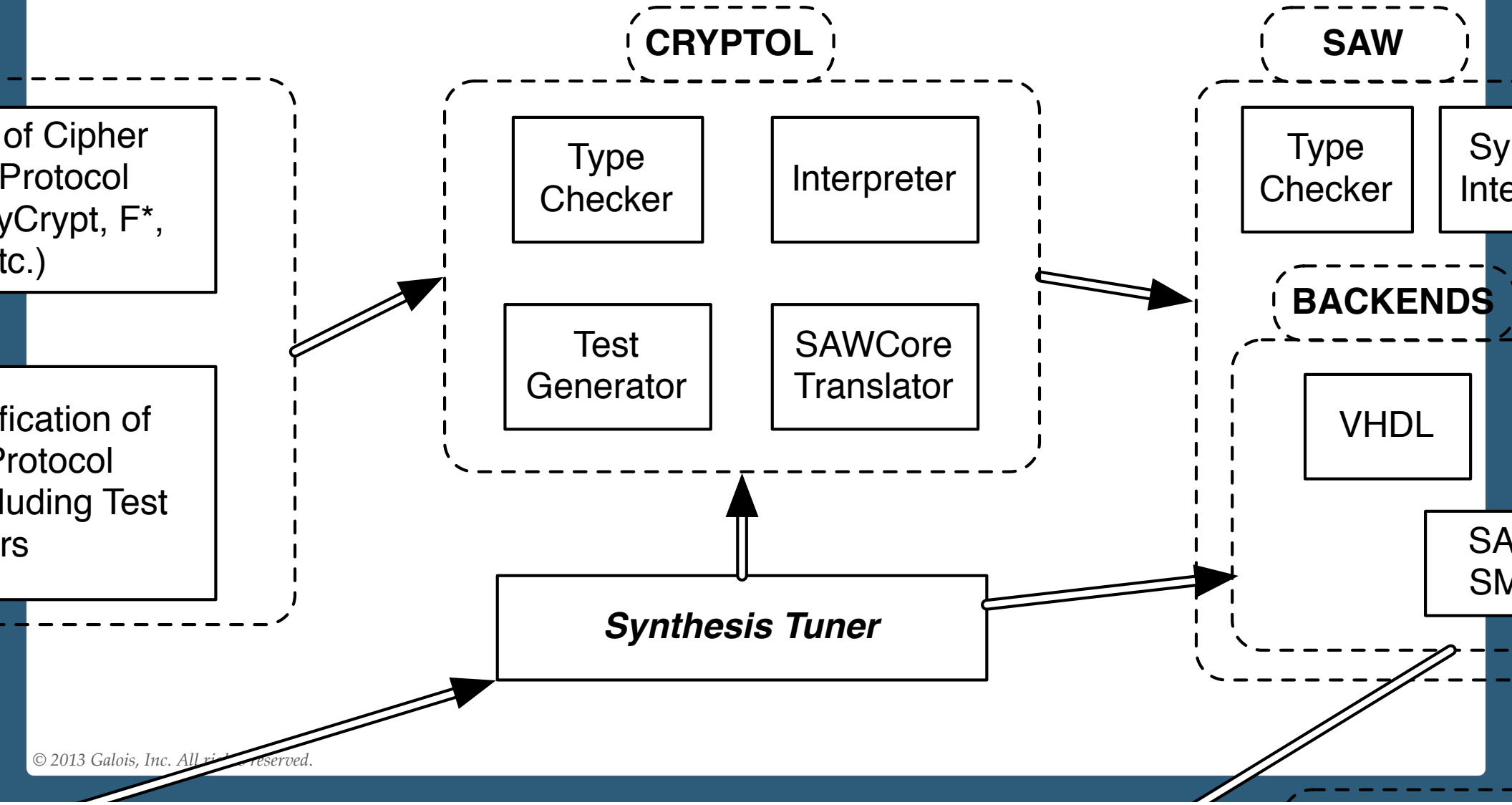


Galois Crypto IP

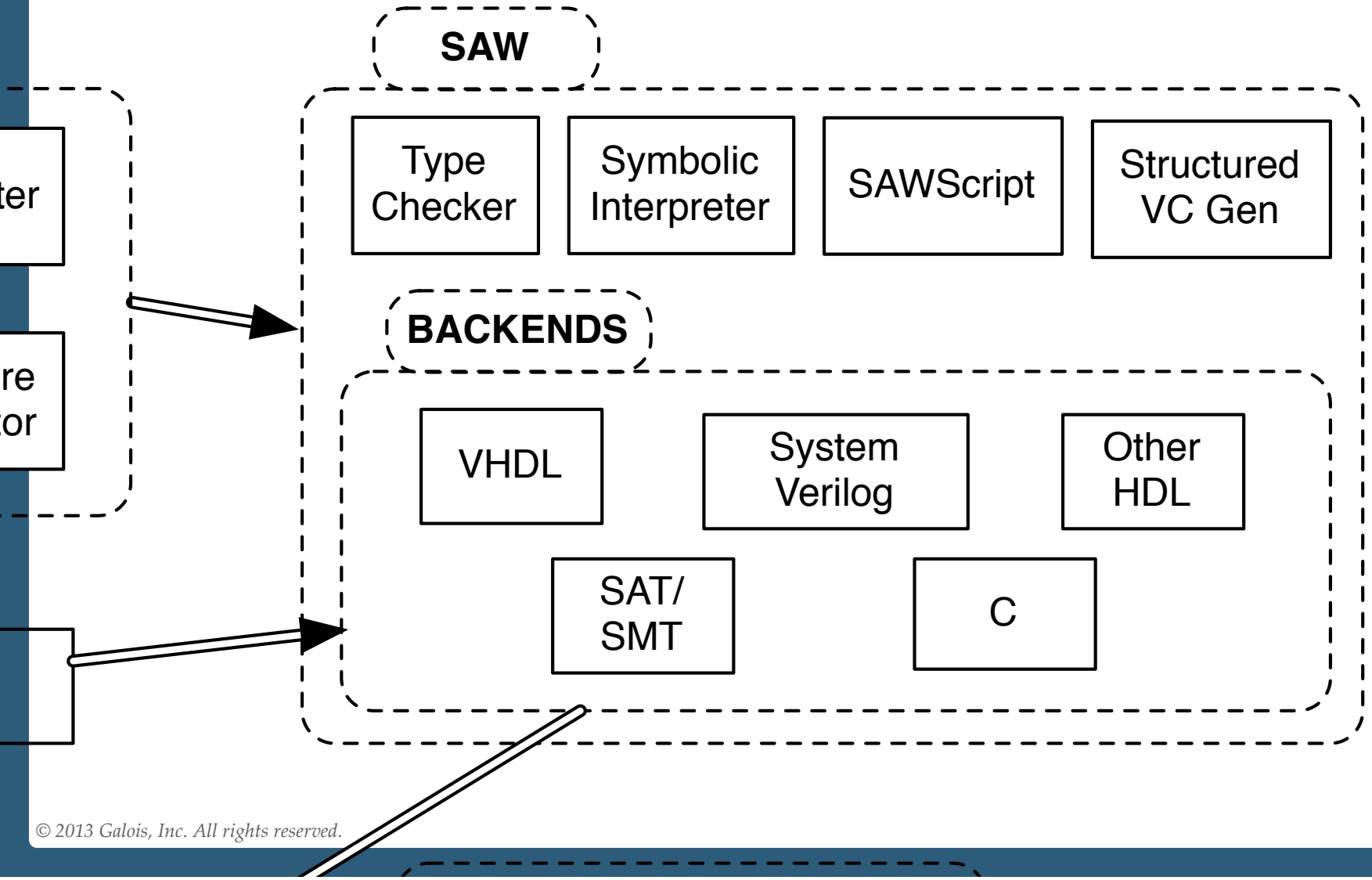
|galois|

- our flagship product in the crypto space is **Cryptol**
  - Crypto is a DSL and verification system for formally specifying and verifying cryptographic algorithms
  - used extensively by intelligence customers
  - we hope to convince NIST to use it as a foundation for future cryptographic standards' development
- we have formally specified and verified nearly all modern and historical cryptographic algorithms in Cryptol, and we have formally verified implementations of many of them

Galois HACrypto: Cryptol



Galois HACrypto: SAW

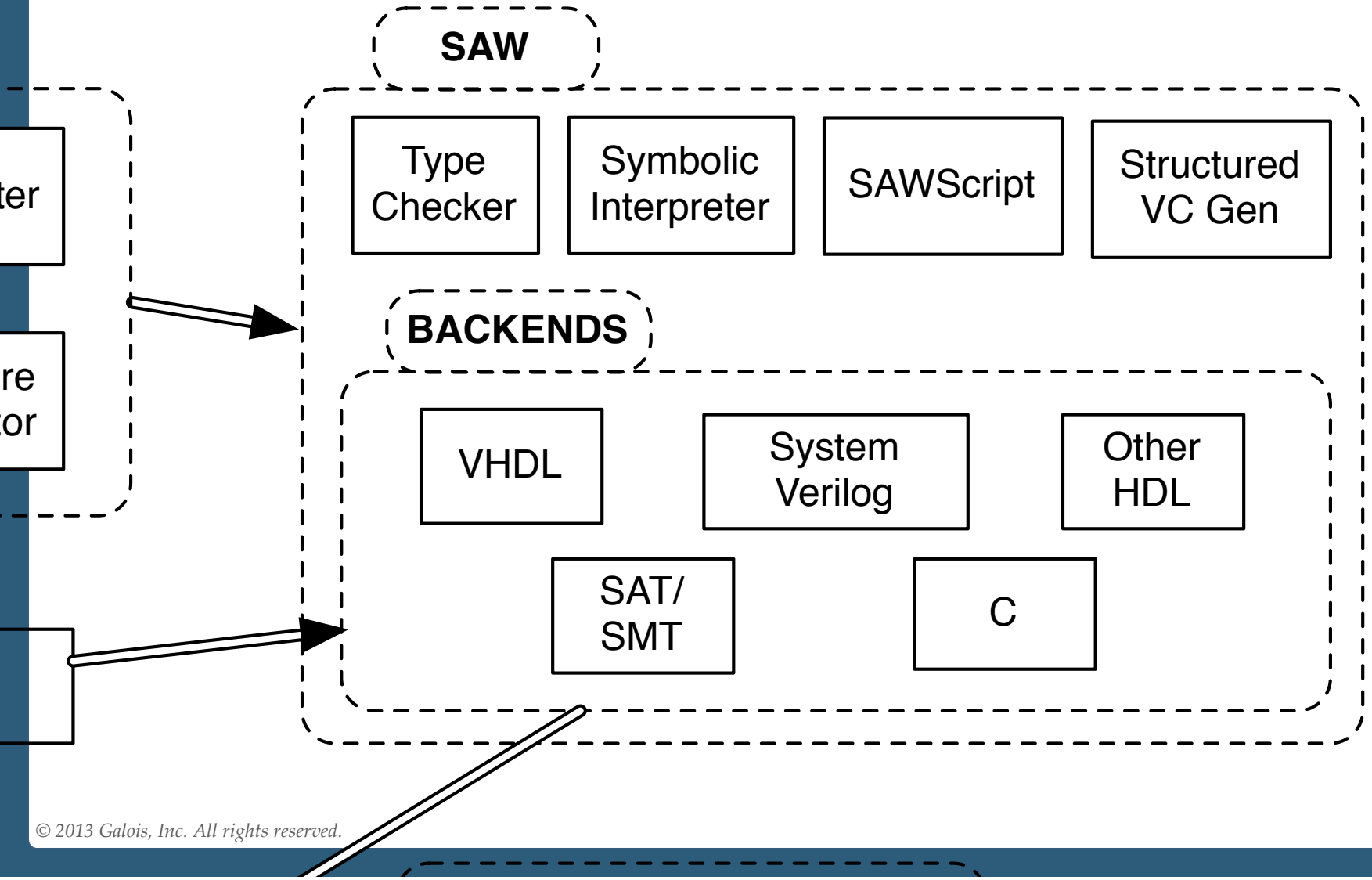


Galois Verification IP

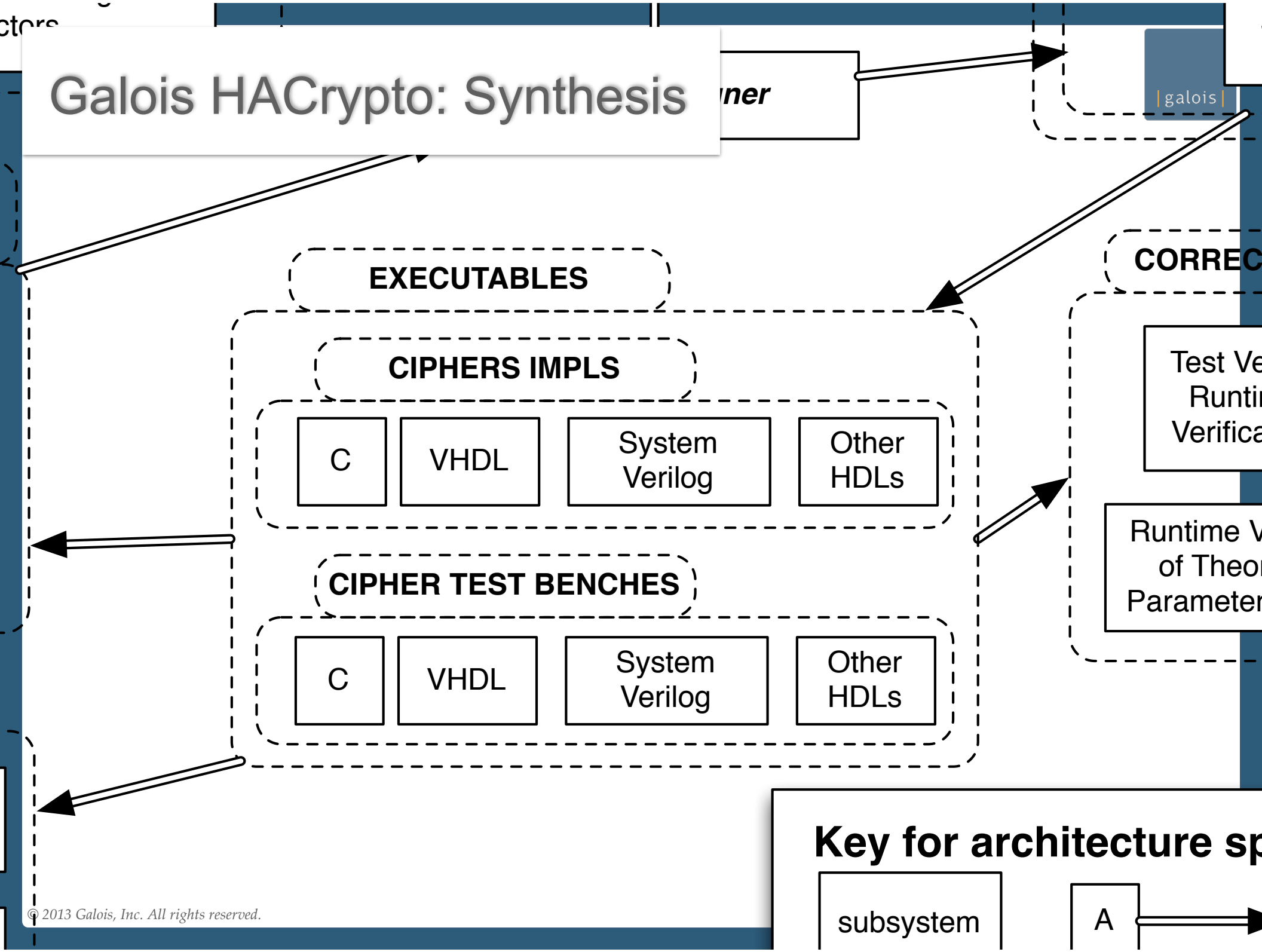
|galois|

- our flagship product in the verification space is **SAW**, the Software Analysis Workbench (presented by Aaron Tomb this week at HCSS on Tuesday afternoon)
  - capable of reasoning about the total correctness of LLVM, JVM, and MATLAB implementations
  - highly tuned toward bit-centric computing (e.g., crypto)
  - works in tandem with Cryptol
- working with IMDEA, we use *EasyCrypt* for complementary verification about side-channels and hardware fault analysis
- we are also using FRAMA-C, Leroy's CompCert, VST from Appel et al., and FCF from Morrisett et al.

Galois HACrypto: SAW



Galois HACrypto: Synthesis



Galois Synthesis IP

|galois|

- our flagship product in the synthesis space is an earlier version of Cryptol (Cryptol version 1)
  - capable of generating verifiable C, JVM, VHDL, and Verilog implementations of Cryptol specs
  - implementations witness decent performance
  - implementations can be verified with other toolchains
  - synthesis goals previously focused exclusively on code as the target artifact, not validation or verification artifacts like test benches or proofs, resp.

Recent Galois HACrypto IR&D in Synthesis

|galois|

- we are forward porting ideas and code from our synthesis tools into Cryptol version 2 and SAW
  - fully automatic synthesis of rigorously engineered C, LLVM, JVM, and SystemVerilog implementations
  - synthesis includes domain model, requirements, correctness and security policies, architecture specification, source documentation, validation artifacts (unit and system runtime verification harness), and verification conditions
  - we evaluated adding synthesis to BlueSpec's BSV and to USC's CSP interfaces for SystemVerilog to this tool chain in November

BlueSpec Evaluation

|galois|

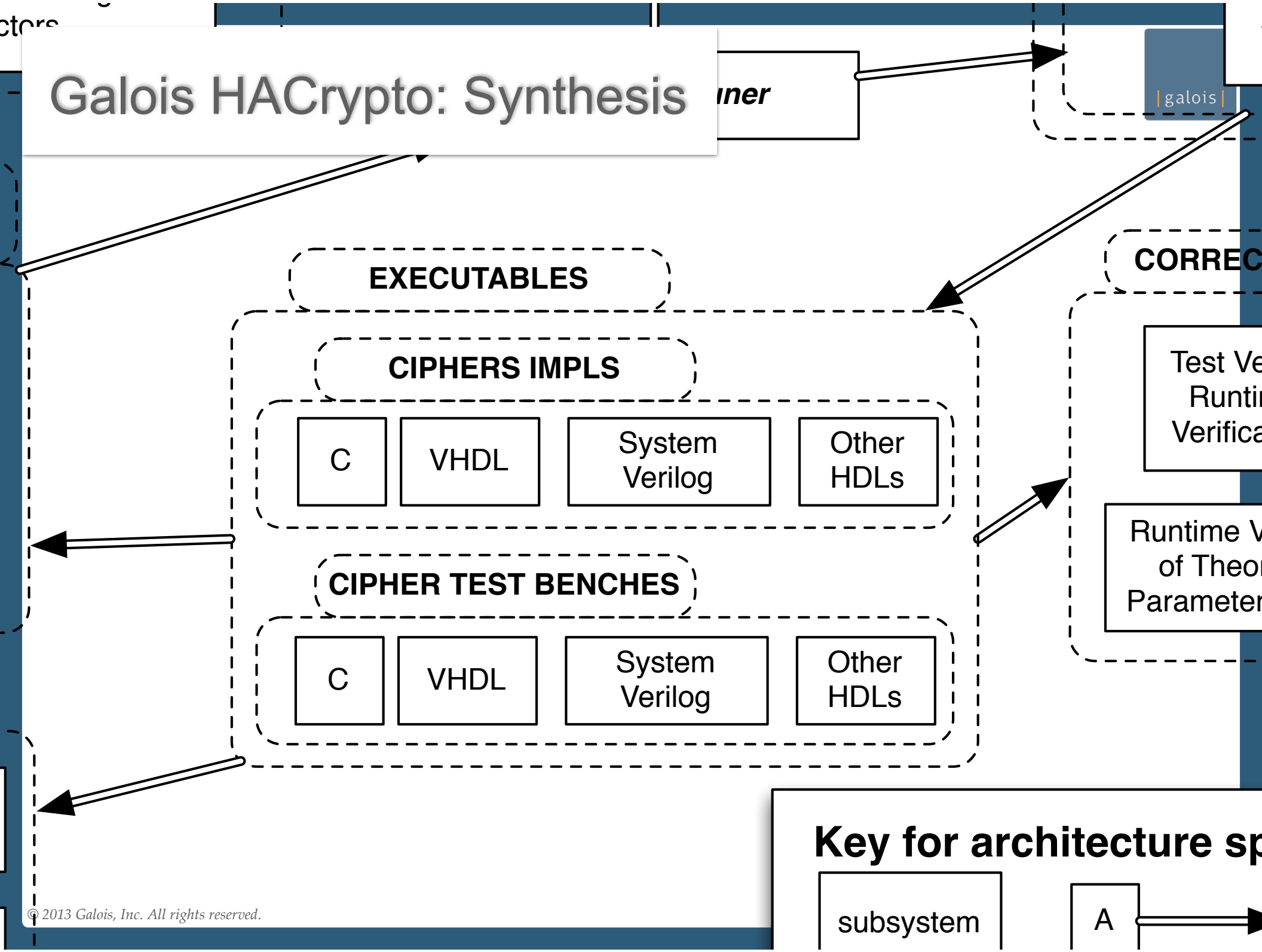
- BlueSpec has a powerful tool chain for modeling, simulating, and synthesizing clocked designs for FPGAs and ASICs
  - great for modeling architectures and variants, exploring products in design space, and measuring performance via simulation against a logical clock
- the semantic gap between Cryptol and BlueSpec's BSV, while not enormous, is large enough that it would take several man months to build a high-assurance toolchain with BSV
  - work would entail doing a mechanized denotational semantics of Cryptol, BSV, and the refinement relation between them (both relational and injective)

USC/REM Evaluation

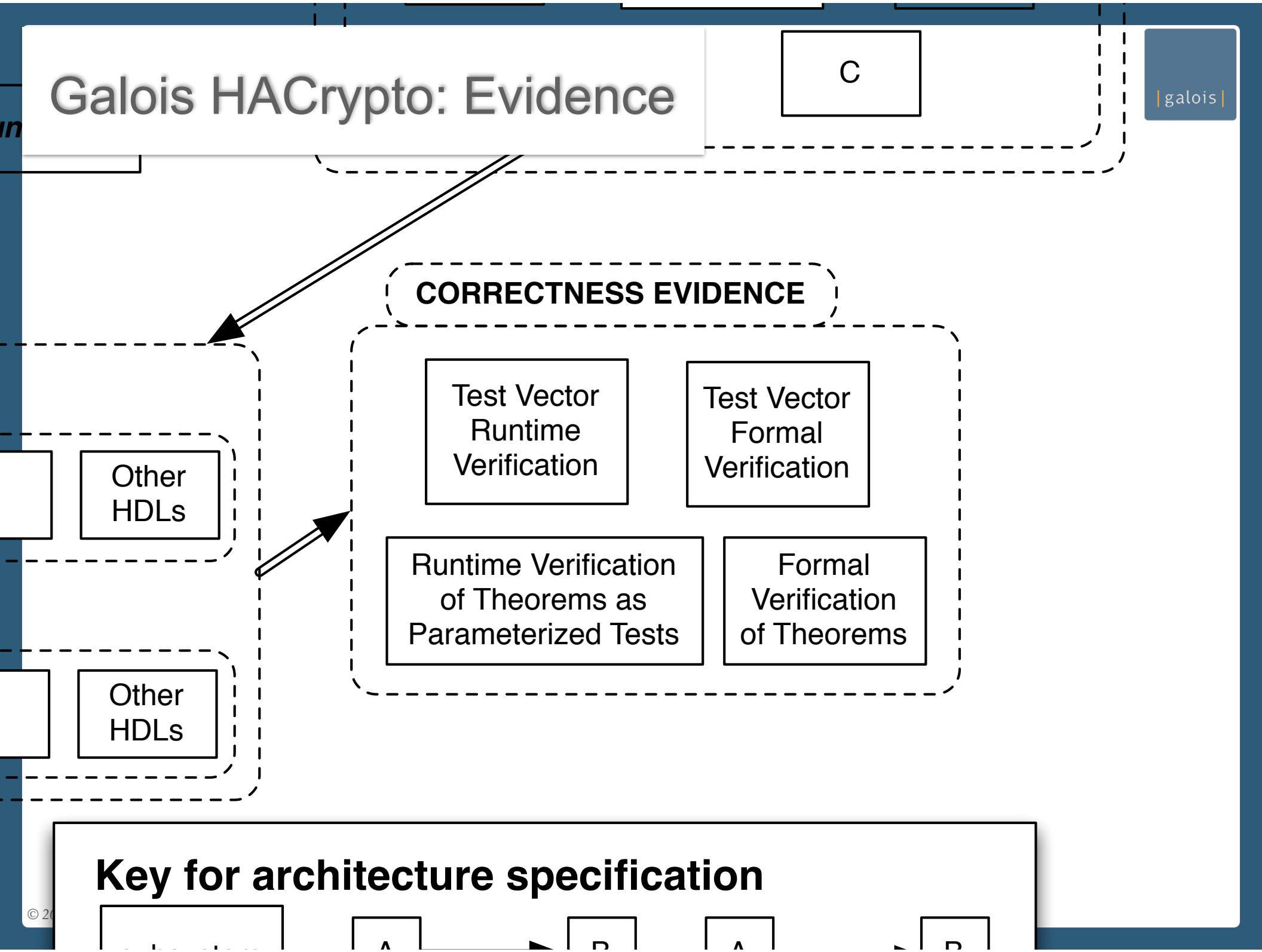
|galois|

- modeling via CSP and SystemVerilog interfaces is as straightforward as we expected
- lack of any support for the specification and reasoning of CSP programs was to be expected, but still somewhat disappointing after a decade of R&D on this topic at Fulcrum Microsystems and elsewhere
- our use case (both in terms of application domain and our tool chain) fits extremely well with USC methodology
- compilation of Cryptol and SawCore into SystemVerilog-CSP is straightforward and development is underway
- estimated performance looks great, but the proof is in the pudding; a full Synopsis license is necessary and has been obtained to do a proper ASIC evaluation

Galois HACrypto: Synthesis



Galois HACrypto: Evidence

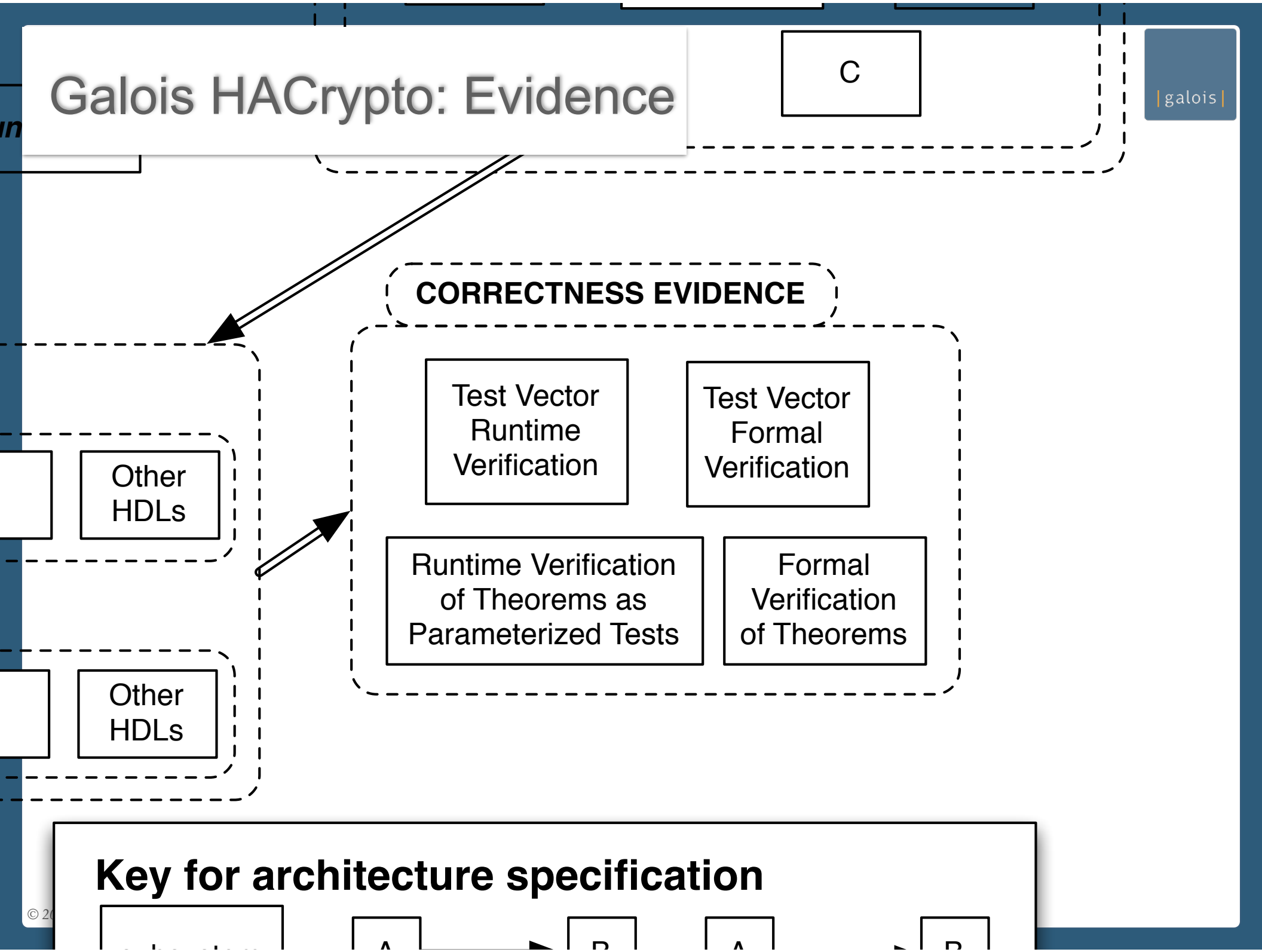


New Galois Toolchain

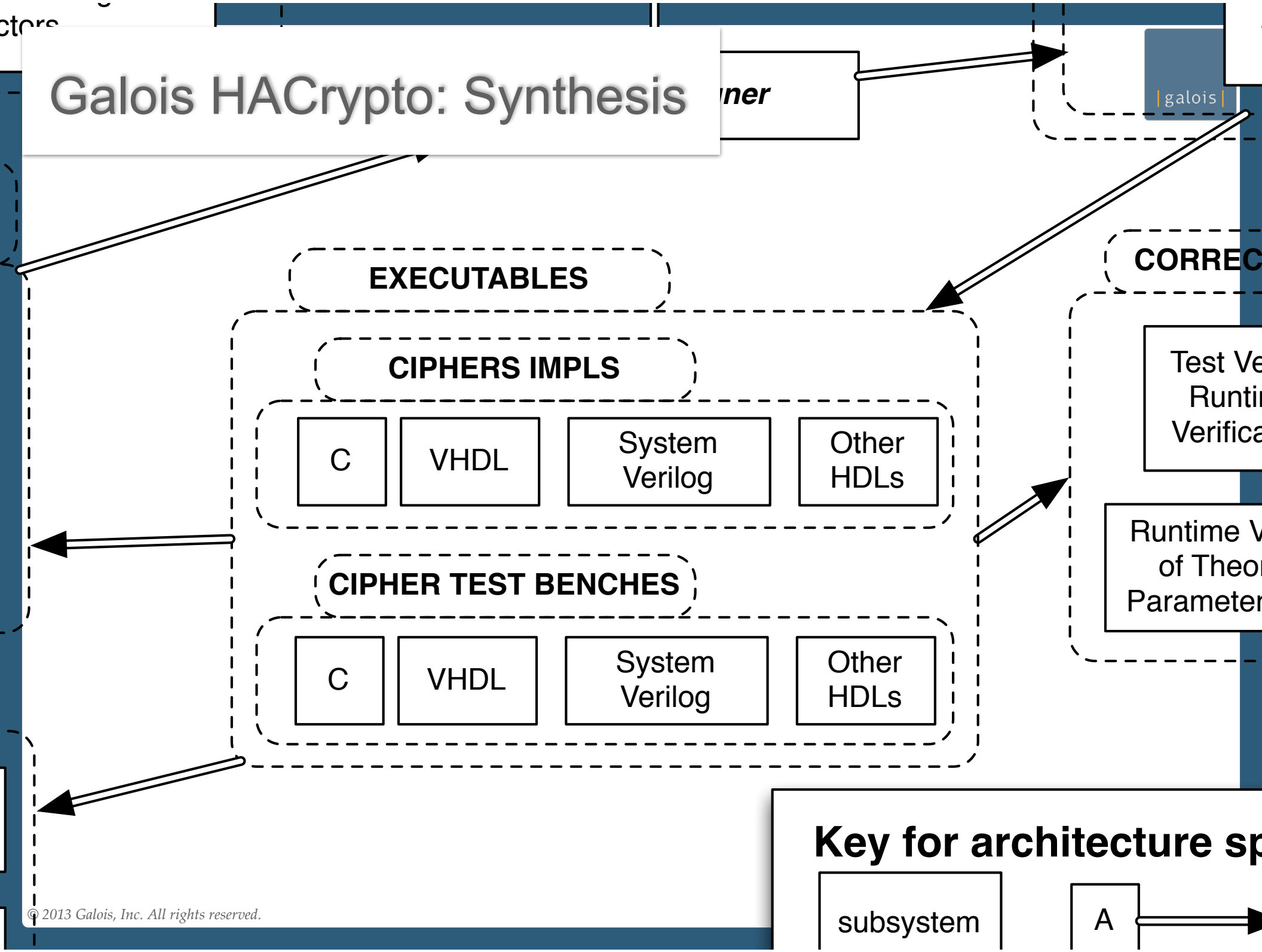
|galois|

- our new tool chain maps AIGs within SAW to (a fragment of) plain-old-Verilog or SystemVerilog (POV and SV henceforth), preserving enclosing structure
- essentially the SawCore expression language is translated to a combinational circuit in POV via an AIG
- preserved structure is module and function declarations that are mapped to SV interfaces/modules and module ports
- SawCore function application is mapped to assignments on ports (for POV) or sends on CSP channels encoded in the USC SV Channel interface
- we generate validation and verification artifacts for both software and hardware by translating Cryptol properties into validation tests and verification conditions
- use symbolic interpretation of properties to partially evaluate top-level functions used in properties to derive function and module tests and verification conditions

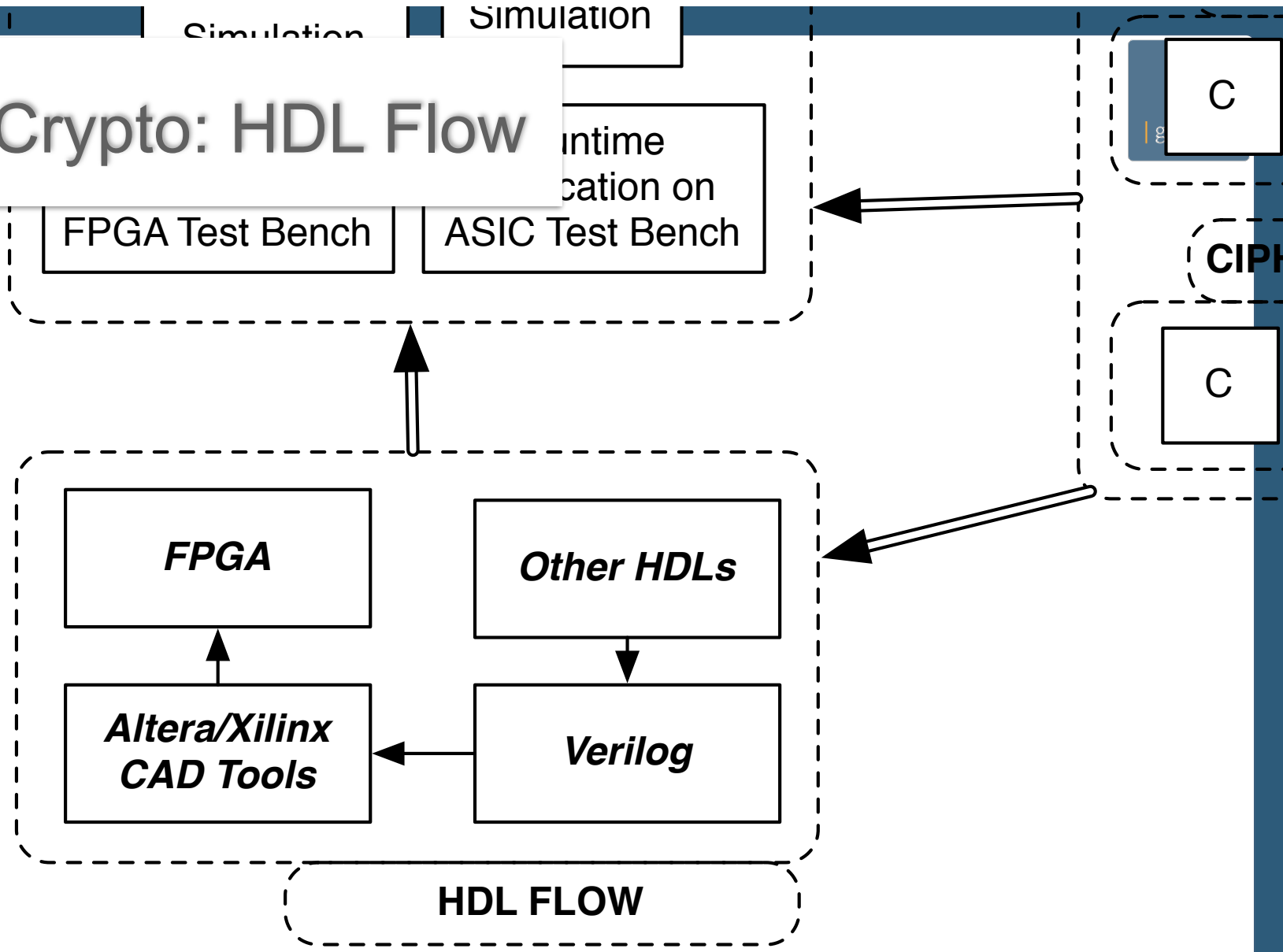
Galois HACrypto: Evidence



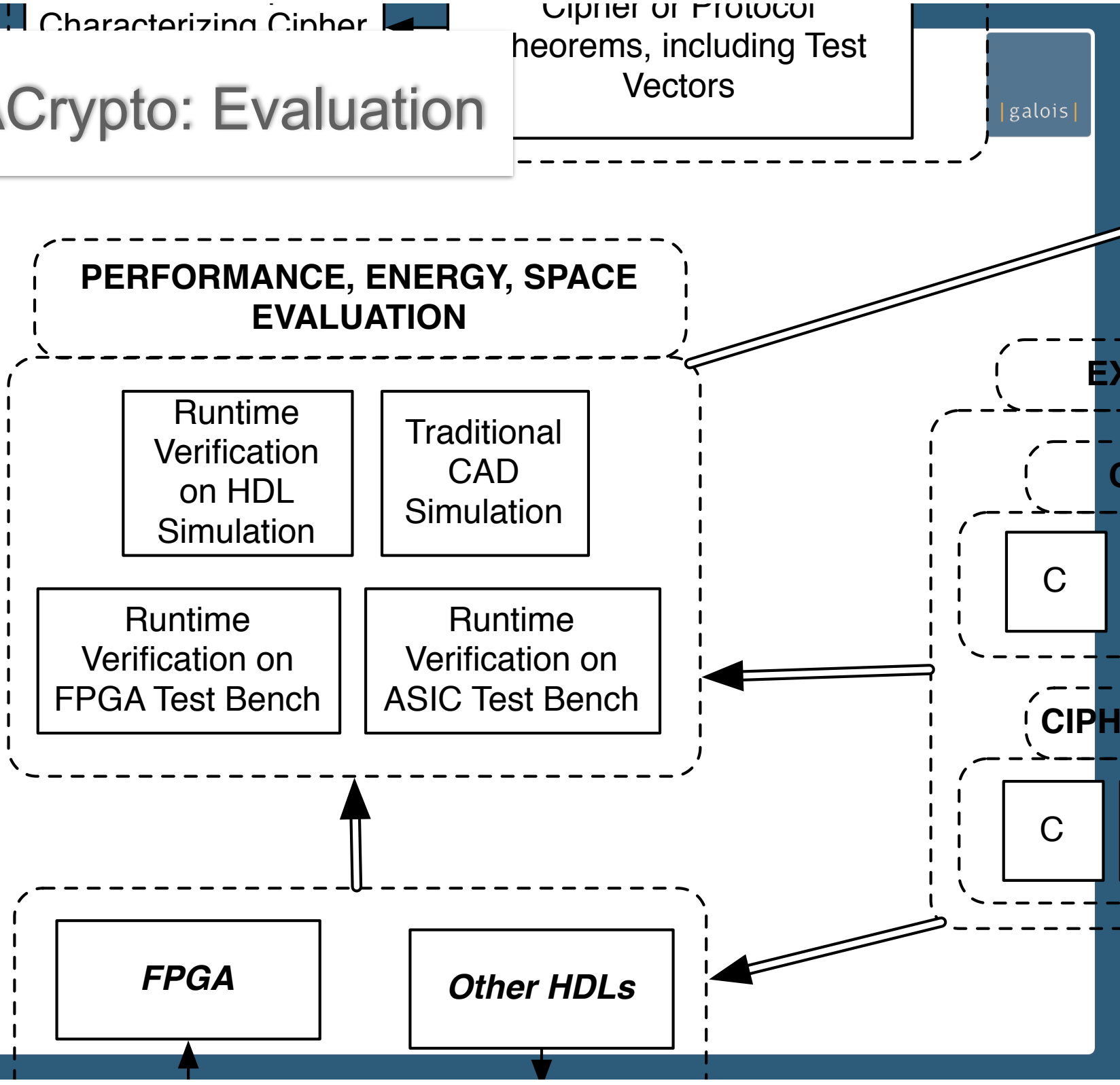
Galois HACrypto: Synthesis



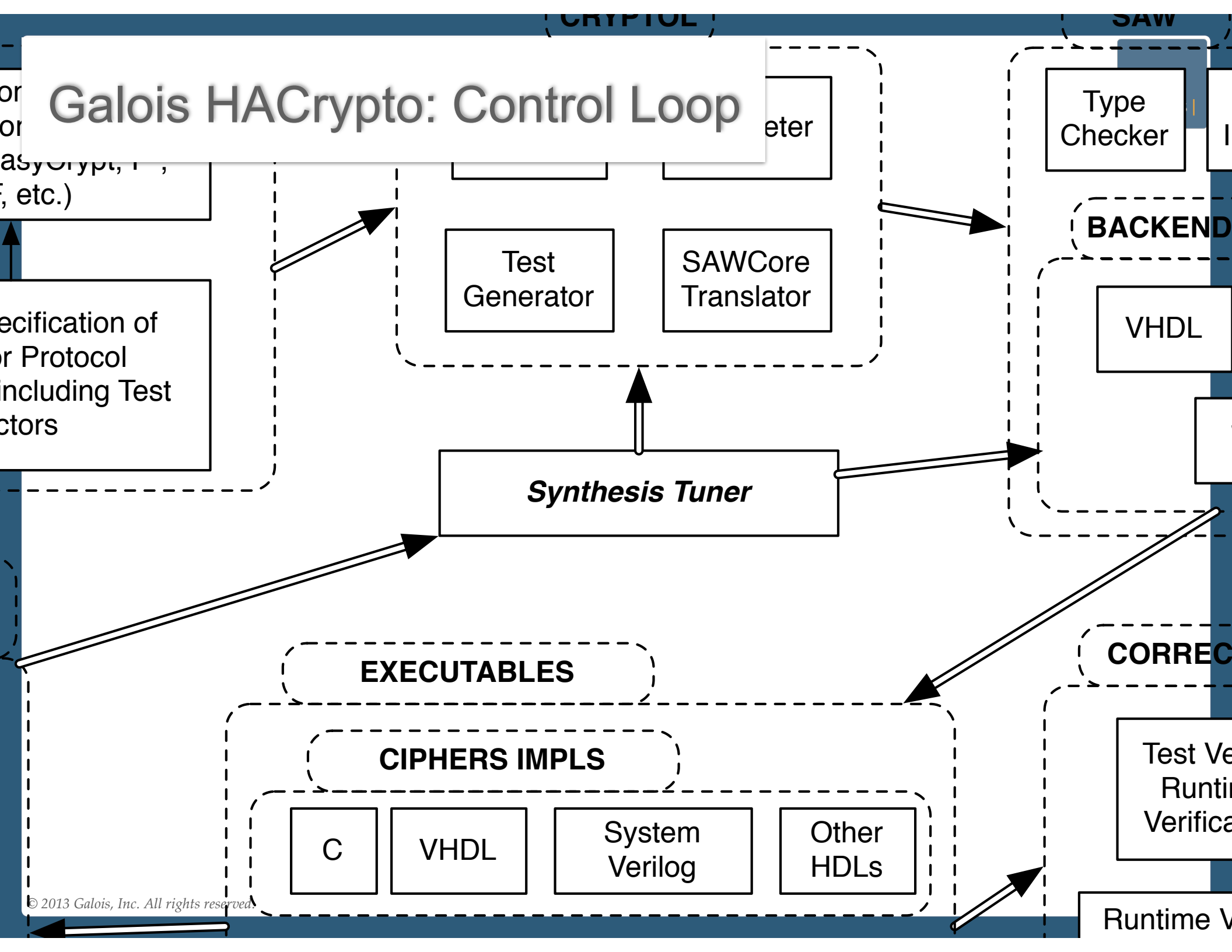
Galois HACrypto: HDL Flow



Galois HACrypto: Evaluation



Galois HACrypto: Control Loop



R&D Next Steps

|galois|

- Daniel and Trevor are working on first version of compilation of SawCore directly from Cryptol and via SAW AIGs into SV and SV-CSP
- Dan is working on module and function compilation to a CSP model to permit us to reason about systems for safety and progress properties using model checkers like FDR3
- we are assembling a tool chain which automatically explores product line space using feedback from simulation
- Trevor (with input from Brian Huffman and Daniel Wagner) is working on a new C compilation feature by augmenting our SBV backend and is exploring simulation via HCSP
- we will complete our Suite B and non-Suite B ECC Cryptol specifications (a few modes of various Suite B algorithms and some Suite B ECC are all that are left)
- finish design and specify Galois Crypto API
- evaluate or estimate performance of synthesized, pipelined **libvcrypt**
- fabricate our first **havcrypt** consumer FPGA covering Suite B and our first ASIC focusing on a subset of ciphers and the SystemVerilog-CSP approach

Product Design Space

|galois|

- we will use Software Product Lines R&D to configure and maintain **libvcrypt** and **havcrypt**
- *both products* potentially have the following binary variance points: concurrency, behavioral runtime verification, side-channel verification, behavioral formal verification, CAVP certification, FIPS certification
- **libvcrypt** potentially has the following variability associations: operating systems, compiler sorts and configurations, and verification sorts
- **havcrypt** has the following variability associations: FPGA choices, ASIC processes, synthesis tool chains and configurations, and verification sorts
- both products can contain any subset of algorithms defined in the Galois domain model of cryptography
- *bespoke products for clients within this product space are possible*

Galois Multicore Crypto Chip (havcrypt)

|galois|

- the **havcrypt** family will support configurable high performance concurrent pipelines for each supported crypto algorithm
  - e.g., for AES several AES cores optimized for the 1- and 2-block case will run in parallel with several deep pipelines for the streaming
- supporting multiple encryption algorithms in a single chip is possible even in the FPGA setting given the size of our hardware implementations
  - it looks like high-end FPGAs have enough space to fit all algorithms customers care about (Suite B plus IRTF CFRG)
  - a single ASIC can easily contain all algorithms and all modes
- we can specify the entire chip within Cryptol, thereby simulate it and formally verify functional properties about the design prior to synthesis
- the software interface to **havcrypt** is being co-designed with the hardware, thus is elegant, maps directly to the mathematics, and formally specified

Galois Parallel High-assurance Crypto Library

|galois|

- **libvcrypt** is our software crypto library
- much of **libvcrypt** can be automatically generated by our Cryptol & SAW toolchain under development
- much like in hardware, we can map control flow and function application to either modules and function application (for a single threaded library) or CSP processes and message sends (for a concurrent library)
- the top-level API for **libvcrypt**, called the **Galois Crypto API**, is identical to that of **havcrypt**, thus reduces our maintenance cost and client dev costs
- the **Galois Crypto API** is easily mapped to existing crypto APIs via a façade, thus we can build a drop-in replacement for the likes of OpenSSL & BouncyCastle

Hardware Technical Details

|galois|

- there is no reason to use QDI for async process; bundled data is optimal for crypto
- less than a handful of logical operations means that we should inline so that our CSP processes are not too fine-grained
- we should map Cryptol/SawCore arrays to Verilog arrays and let the backends optimize access
- the SawCore control flow graph maps directly to the CSP design, thus we need a small CFG module for SawCore and a CSP channel emitter for function application
- a large (>16) fan in/out in the data flow means that we must replicate processes to avoid contention (more space, better performance); we have yet to see a CFG for crypto that has much fan in/out (such is by design)
- bitsize of types does not matter at all except at the chip interface (pin count)
- new annotations are necessary at the function-level to encode forward/backward latency, area, and power for a given simulator and target chip/process

Business Decisions

|galois|

- Do we pursue a KickStarter campaign focusing on **havcrypt** on FPGAs for embedded systems?
- What is the appropriate product configuration for **havcrypt** for various business use cases?
  - We should profile OpenSSL use on database server, web server, compute server, etc.
- Should we raise money to kickoff **havcrypt** ASIC design and evaluation or can we bootstrap though customer?
- Is a drop-in replacement for de facto crypto libraries like OpenSSL and BouncyCastle necessary for **libvcrypt** adoption?

Technical Results

|galois|

- **Old toolchain evaluation:** What is the performance and behavior of the synthesized output from Cryptol version 1 on modern FPGAs (measured) and ASICs (estimated)?
- **BlueSpec evaluation:** How can BSV and BlueSpec tools be used in our high-assurance toolchain?
- **USC evaluation:** How might we use a CSP/System Verilog based toolchain for high-assurance synthesis?
- **New Galois toolchain:** How does SawCore map to CSP/System Verilog? How might we verify CSP designs? What is the assurance for such a tool chain?
- **Galois Multicore Crypto Chip (havcrypt):** What is the architecture of a multi-algorithm, multicore chip? How might it look in an FPGA vs. an ASIC setting?
- **Galois Parallel High-assurance Crypto Library (libvcrypt):** How does the architecture of a parallel, multi-algorithm, high-assurance crypto software library look and how does it relate to the hardware product? What parts of the toolchain are reused?
- **Product design space:** What are the obvious products in the design space of Galois Crypto Chips? What does the low-end offering look like? High-end? What guarantees and warrantees might we be able to make about each?