

Design Verification: The Past, Present and Future

Sean Smith, Juniper Networks, RTP,
NC



Agenda

- **Introduction to Sean & Juniper**
- **A little background on DV Past**
- **An overview of where DV is today?**
- **Where is DV Going?**
- **Summary and Q&A**

Who is Juniper at a Glance?

Snapshot:

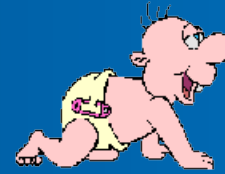
Founded: 1996
NASDAQ: JNPR
Headquarters:
Sunnyvale, CA

Employees: 7,000
employees in
offices in 47
countries.

Customer Profile:
We serve the top
100 global service
providers more
than 30,000
enterprises, as well
as hundreds of
government
agencies and



DV: The Past



- **DV has been a tremendously evolutionary field**
 - As recent as 15 years ago
 - There was little research on DV
 - There were no published texts on DV (until 2000)
 - A large void compared to logic design.
 - Considered by many to be the n00b job

- **Common DV Techniques**
 - Visual inspection
 - Diff the golden log files
 - Directed self checking tests
 - Code coverage was in its infancy
 - C++ was a common choice in the high end

DV: The Present



- **Wow! DV has Evolved!**
 - Lots of research on DV including DV centric conference
 - Dozens of published texts
 - Still a void compared to logic design but narrowing
 - Not necessarily a newbie job but there is a delta in compensation & respect

- **Common DV Techniques**
 - Self checking constrained random
 - Objective metrics for measuring completeness
 - Static verification has become almost mainstream
 - SystemVerilog provided the unification

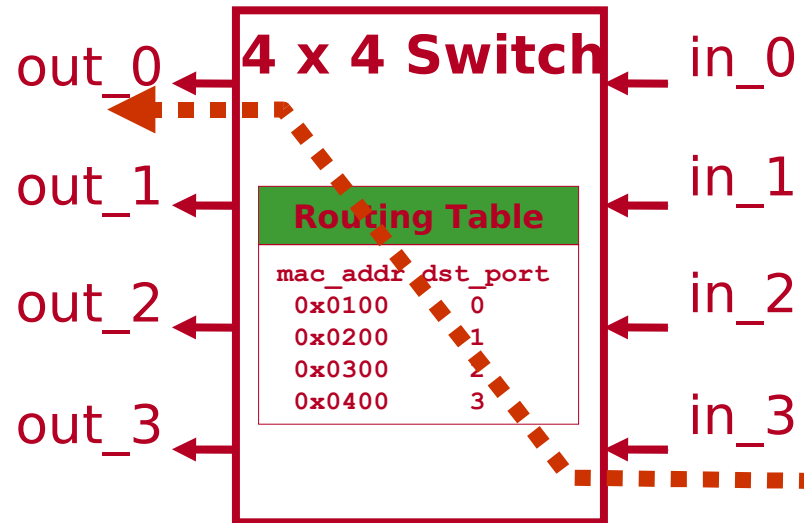
DV: The Future



- **How are we going to deal with capacity?**
 - Current simulation techniques are running out steam?
 - Emulation?
 - Formal?
- **How can we be smarter about reaching our verification goals?**
- **Are the traditional problems where we should be focusing our energy?**

Dynamic Simulation Challenges

■ Closing Coverage...



Packet Format

dst 0x100	src 0x500	len 2	data 0x0102030 4	crc 0x9bdbc90 0
---------------------	---------------------	-----------------	-------------------------------	------------------------------

Closing coverage continued...

- **Constrained random tests are generally not coverage aware so they generate a lot of duplicate transactions....**
- **New Approaches are emerging to try and automate coverage specification and closure!**
 - **Graph and Genetic based algorithms for stimulus and coverage generation.**
 - **Algorithms to mutate stimulus code**
 - **Context aware simulators**
 - **Formal Verification**

Formal Targeted ROI and Applications

Formal Application Area	Examples		Resource Impact	Risk Impact	TTM Impact	ROI
Architectural verification	<ul style="list-style-type: none"> ▪ Communications and bus architecture 	<ul style="list-style-type: none"> ▪ Cache coherency 		✓ High	✓ High	✓ High
RTL design and debug	<ul style="list-style-type: none"> ▪ Designer sandbox ▪ Register verification 	<ul style="list-style-type: none"> ▪ X-propagation detection ▪ Connectivity check 	✓ High		✓ High	✓ High
Proofs of critical functionality/ Verification	<ul style="list-style-type: none"> ▪ Packet integrity ▪ Multi-processor coherence ▪ Flow control 	<ul style="list-style-type: none"> ▪ Error correction ▪ Protocol certification ▪ Token leakage ▪ Regression test 	✓ High	✓ High	✓ High	✓ High
Verification of low-power modes	<ul style="list-style-type: none"> ▪ Power architecture ▪ Clock gating 	<ul style="list-style-type: none"> ▪ Power shutoff isolation cells 		✓ High		✓ Medium
Post-silicon debug	<ul style="list-style-type: none"> ▪ Root cause bugs 	<ul style="list-style-type: none"> ▪ Validate fixes 	✓ High		✓ High	✓ Varies



- **But are there bigger looming challenges?**
- **We need to step back and consider the big picture.**
 - The goal of verification is deliver a quality product to market in a timely fashion.
 - There is an increasing software component of silicon and systems we build.
 - There is also the redundancy of pre versus post silicon verification.
- **The future of DV is much broader than the current focus!**

Reuse Starts with Specification

- **IP blocks need consistent specification format**
 - Ensure quick and easy integration into SOC level reference manual, users guides, and programmers guides
 - Modular & consistent document structure, formatting, look and feel are key
 - Need to minimize hand editing when going from block to SOC
- **Standards like IP-Xact and SystemRDL from the SPIRIT Consortium are driving the future.**

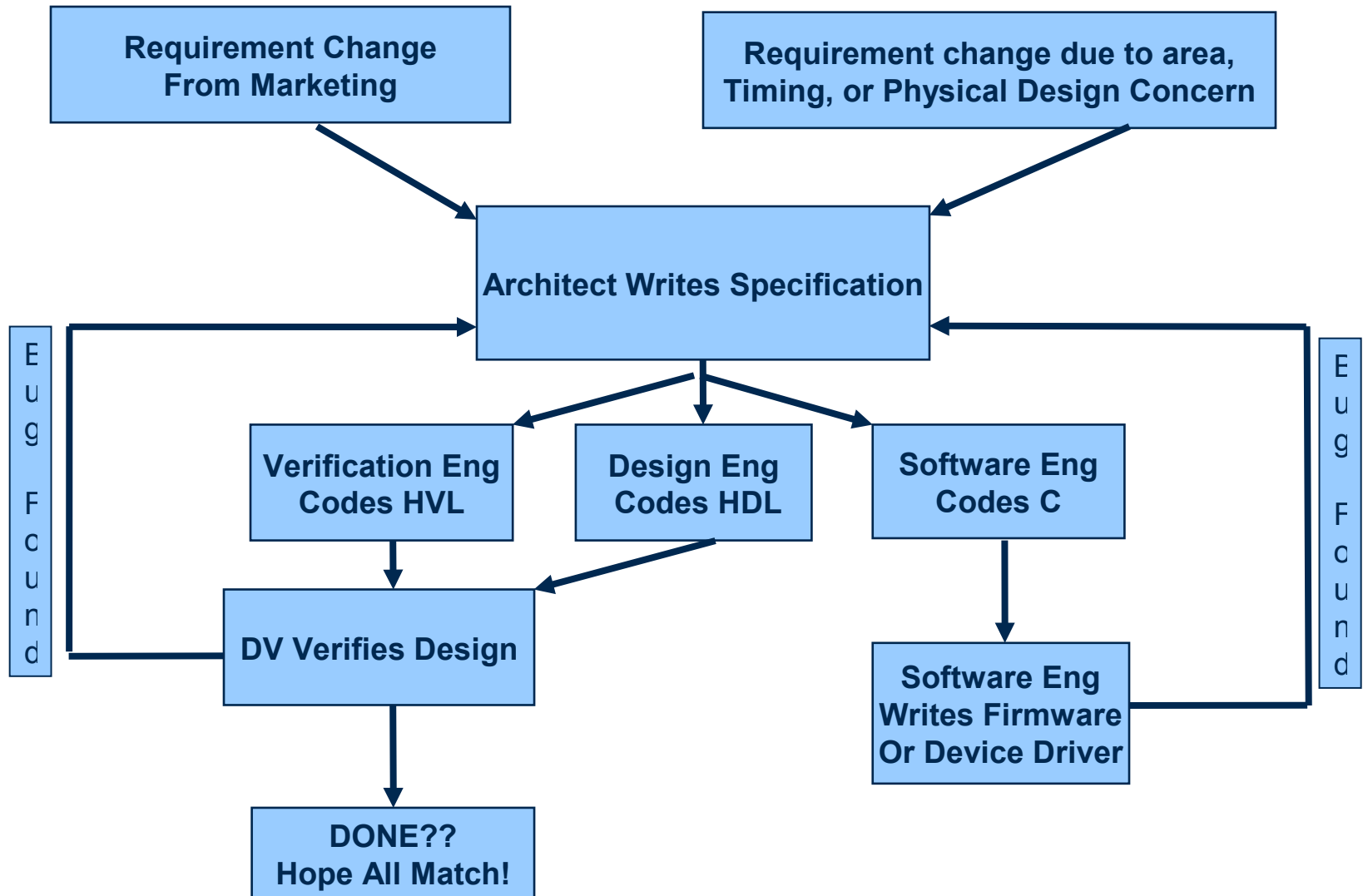
SPIRIT and SystemRDL

- **SystemRDL is a language for specifying register architectures and implementations.**
- **SPIRIT Consortium has accepted the donation of SystemRDL on May 21, 2007**
- **On May 18th 2009, The SystemRDL 1.0 standard was released.**

<http://www.spiritconsortium.org/>



Traditional CSR Flow

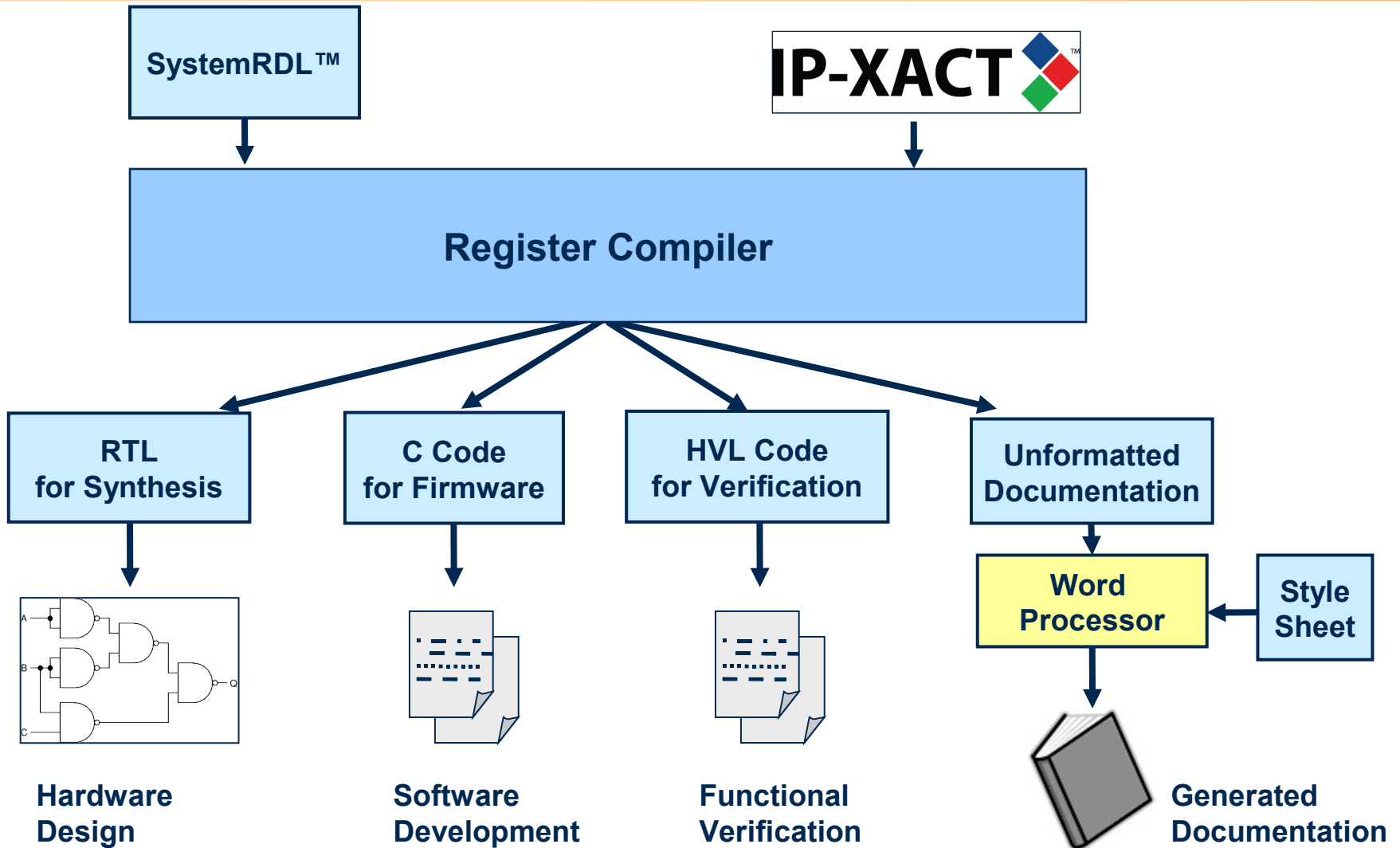


Challenges with Traditional CSR Flow

- **Resource intensive**
 - Multiple engineers creating same/similar content
 - Very tedious and time consuming work
- **Difficult to manage changes in architecture**
- **Doesn't enable common look and feel**
- **Manual nature of process lends to low quality**
 - Specification != HDL != HVL
 - Manual test cases creation for DV
 - Doc updates postponed due to pain factor

Low Reuse + Tedious Work = Low Quality/Productivity

Register Compiler Architecture



■ Simple high level syntax defines registers

```
// This register is a inline register definition.
// It defines a simple ID register.  A flip-flop is implemented
//
reg chip_id_r {
    name = "This chip part number and revision #";
    desc = "This register contains the part # and revision # for
          XYZ ASIC";
    field {
        hw    = rw; // This combination of attr creates a flip flop
        sw    = r;
        desc = "This field represents the chips part number";
    } part_num[28] = 28'h12_34_56_7;
};
```

Defining reusable components in SystemRDL

```
field intr_field_f {  
    desc = "Some Status Field";  
    level intr; // Level Sensitive Interrupt  
};  
  
field count_field_f { // Anonymous Generic Counter.  
    hw    = w; sw = rw; rclr; counter;  
    desc = "Number of certain packet type seen";  
};
```

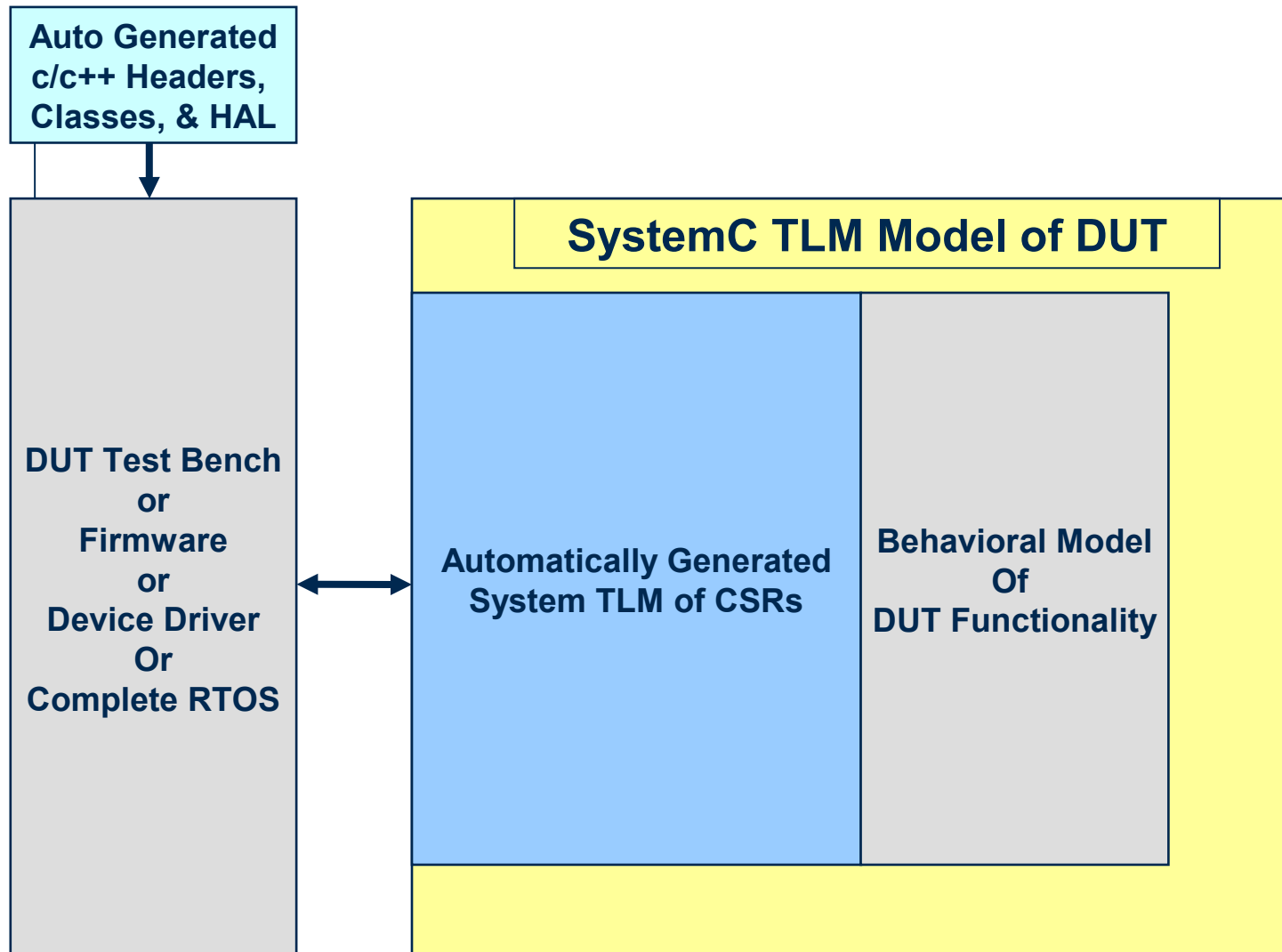
Here we define a couple of field types that we will use on the next slide. This can be used to create templates to ensure common architecture and reuse.

```
reg vc_pkt_count_r {
    count_field_f vc_count[29:0]=0x0;
    field { desc="VC is Active"; } active;
    intr_field_f cntr_overflow=0;
    cntr_overflow->next = vc_count->overflow;
};

addrmap some_register_map {
    chip_id_r chip_id @0x0110;

    external vc_pkt_count_r vc_pkt_count[256]
        @0x200 +=0x10; // External Interface
}; // End some_register_map
```

Automate ESL/TLM Creation



Closing Thoughts.....

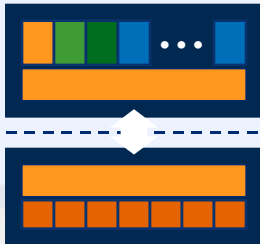
- **Verification has come a long way and has a promising future.**
- **Speed and Capacity are becoming real bottlenecks for traditional approaches.**
- **More focus needs to be placed on co-verification and verification portability.**
- **SystemRDL and IP-Xact are pragmatic languages that provide benefits to many aspects of ASIC/SOC creation.**

Juniper advantage

The POWER of



Architecture



Operating System



Open Network

