2023 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

SAN JOSE, CA, USA
FEBRUARY 27-MARCH 2, 2023

Hardware/Software Interface Formats
A Discussion

Richard Weber, Jamsheed Agahi, Josh Rensch, Eric Sherk

Semifore
An Arteris Business

accellera SYSTEMS INITIATIVE

Agenda

- Introduction
- Non-authoring languages
- Custom purpose-built HSI authoring solutions
- General purpose-built HSI authoring solutions
- Q&A

Introduction

- Goal: work smarter, not harder

- Problem:
  - More work than time or people
  - Conflicting needs
  - Many languages
  - Standards are complex

- Comparative and subjective discussion

Jamie, Lead Engineer

Non-authoring languages
UVM RAL and PSS

Non-authoring languages – UVM RAL

- Standard base class library and API
- Verification view of the address map
- Strengths
  - Uniform methodology
  - Front- and back- door access
  - Addressmap hierarchy & threads
  - Test generation
  - VIP reuse

Non-authoring languages – UVM RAL

- Weakness - not suitable for authoring
  - Low abstraction level
  - Limited view generation capability

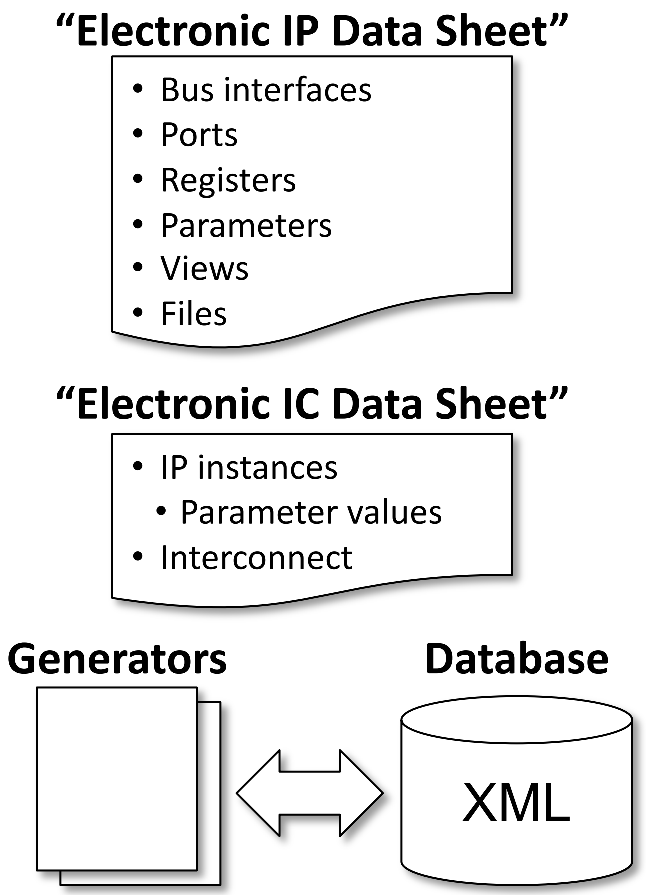
Non-authoring languages – Portable Stimulus

- Packed struct
  - Standardized struct
  - Fields of register
  - Structure only, no behavior

Non-authoring languages
IP-XACT

IP-XACT, a standard structure for packaging, …

- XML Schema Definition (XSD)
  - component, design, …
- Semantic Consistency Rules (SCR)
  - overlapping registers, multiple drivers, …
- Tight Generator Interface (TGI)
  - API provided by IP-XACT Design Environment



IP-XACT development

• IP-XACT 1.0	December 2004
• IP-XACT 1.1	June 2005
• IP-XACT 1.2	April 2006
• IP-XACT 1.4	March 2008
• IEEE Std. 1685-2009	December 2009
• IEEE Std. 1685-2014	June 2014
• IEEE Std. 1685-2022	September 2022

IP-XACT Purpose

- IP-XACT is
  - for IP interchange between IP providers and consumers
  - for interchange between tools
  - documentation of physical interfaces
  - documentation of software interfaces
- IP-XACT is NOT
  - an authoring format
  - another model of hardware behavior

IP-XACT HSI Structure

- MemoryMap
  - Container for AddressBlocks
- AddressBlock
  - Container for register data – registerfile or registers
  - Memory
  - Memory containing virtual register data
- Registerfile
  - Container for register data
- Register
  - Container for fields

IP-XACT HSI Features

- Offsets of registers and register containers
- Positions of fields
- Software access behavior for fields
- Reset values for fields
- Volatile
- New in 2022 working backdoor or HDL paths
- New in 2022 mode dependent software access behavior

Custom HSI Authoring Solutions

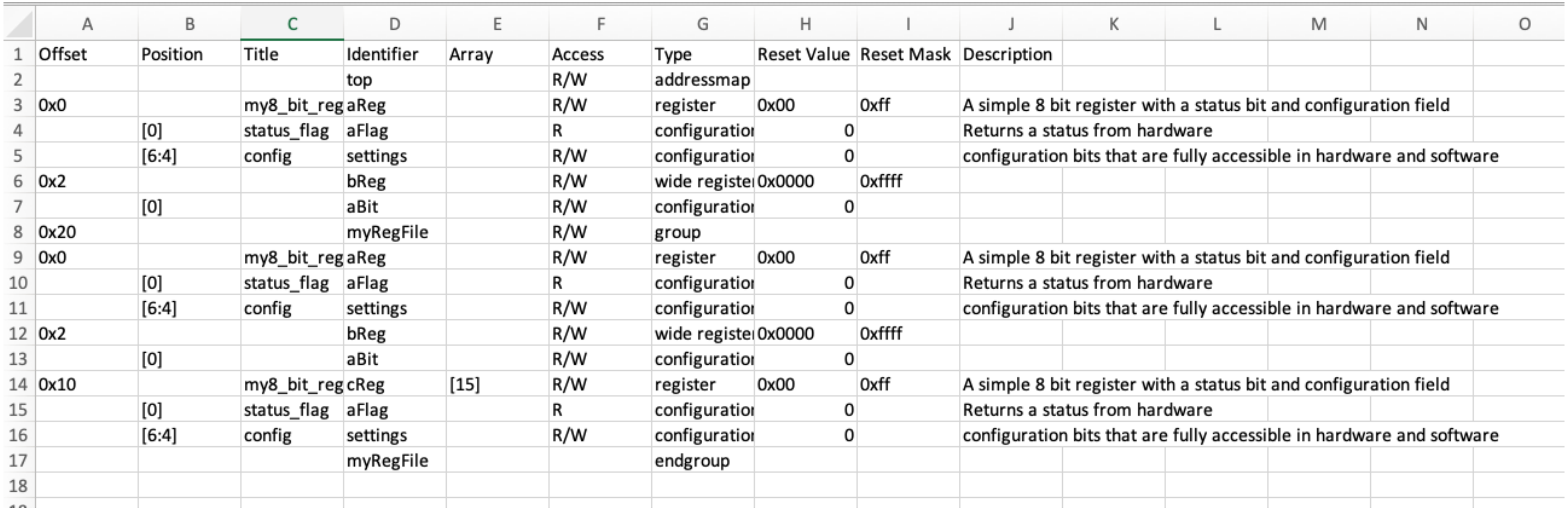
Jamie, Lead Engineer

Custom HSI Authoring Solutions

- company, team, project specific
- general purpose language
- custom data structure
- limited input and output processing
- high maintenance
- limited scalability
- limited sharing
- limited reuse

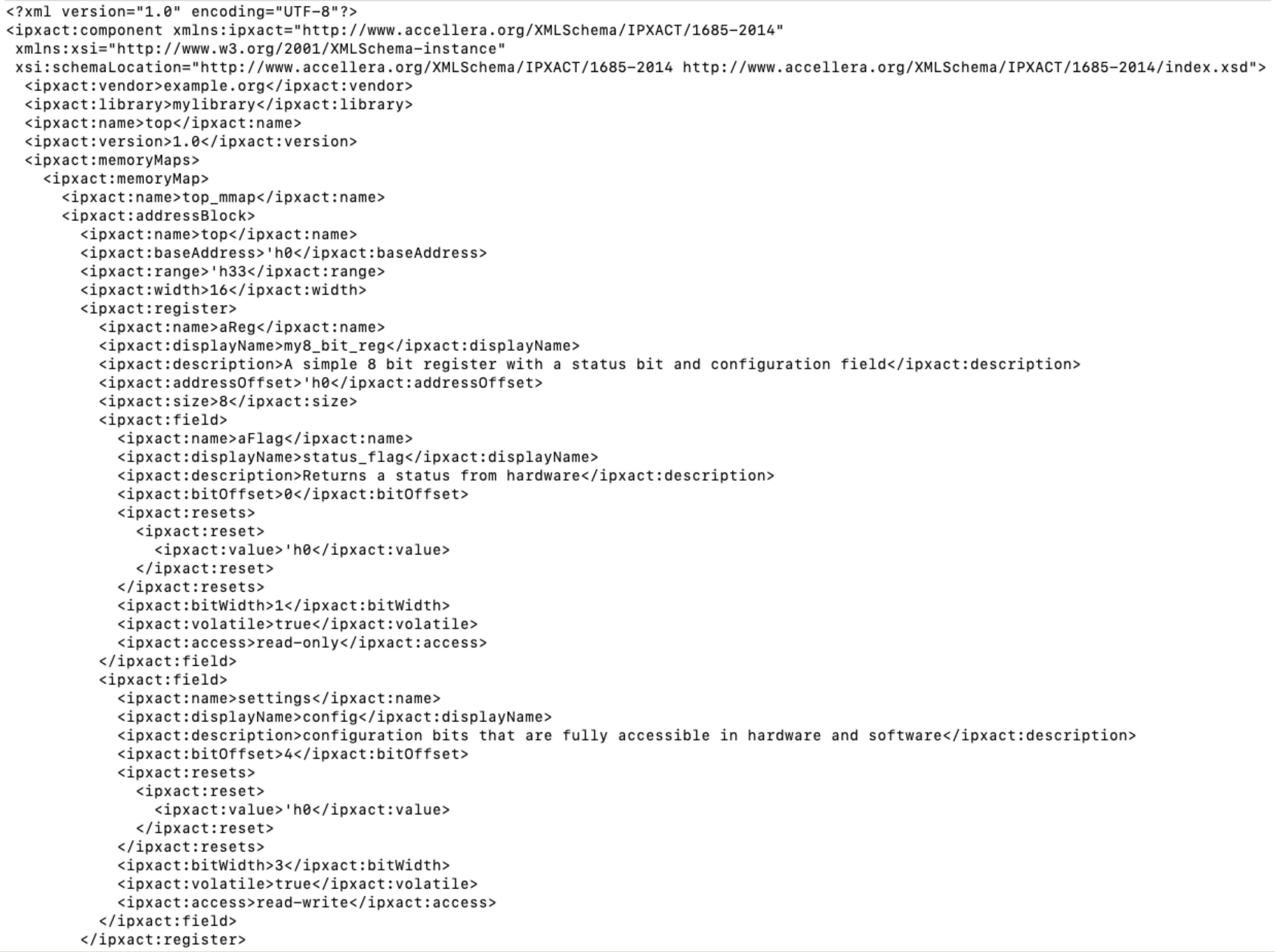
Jamie, Lead Engineer

Spreadsheet Example



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	Offset	Position	Title	Identifier	Array	Access	Type	Reset Value	Reset Mask	Description					
2				top		R/W	addressmap								
3	0x0		my8_bit_reg	aReg		R/W	register	0x00	0xff	A simple 8 bit register with a status bit and configuration field					
4		[0]	status_flag	aFlag		R	configuratio	0		Returns a status from hardware					
5		[6:4]	config	settings		R/W	configuratio	0		configuration bits that are fully accessible in hardware and software					
6	0x2			bReg		R/W	wide registe	0x0000	0xffff						
7		[0]		aBit		R/W	configuratio	0							
8	0x20			myRegFile		R/W	group								
9	0x0		my8_bit_reg	aReg		R/W	register	0x00	0xff	A simple 8 bit register with a status bit and configuration field					
10		[0]	status_flag	aFlag		R	configuratio	0		Returns a status from hardware					
11		[6:4]	config	settings		R/W	configuratio	0		configuration bits that are fully accessible in hardware and software					
12	0x2			bReg		R/W	wide registe	0x0000	0xffff						
13		[0]		aBit		R/W	configuratio	0							
14	0x10		my8_bit_reg	cReg	[15]	R/W	register	0x00	0xff	A simple 8 bit register with a status bit and configuration field					
15		[0]	status_flag	aFlag		R	configuratio	0		Returns a status from hardware					
16		[6:4]	config	settings		R/W	configuratio	0		configuration bits that are fully accessible in hardware and software					
17				myRegFile			endgroup								
18															

Partial IPXACT Example



```
<?xml version="1.0" encoding="UTF-8"?>
<ipxact:component xmlns:ipxact="http://www.accellera.org/XMLSchema/IPXACT/1685-2014"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://www.accellera.org/XMLSchema/IPXACT/1685-2014 http://www.accellera.org/XMLSchema/IPXACT/1685-2014/index.xsd">
  <ipxact:vendor>example.org</ipxact:vendor>
  <ipxact:library>mylibrary</ipxact:library>
  <ipxact:name>top</ipxact:name>
  <ipxact:version>1.0</ipxact:version>
  <ipxact:memoryMaps>
    <ipxact:memoryMap>
      <ipxact:name>top_mmap</ipxact:name>
      <ipxact:addressBlock>
        <ipxact:name>top</ipxact:name>
        <ipxact:baseAddress>'h0</ipxact:baseAddress>
        <ipxact:range>'h33</ipxact:range>
        <ipxact:width>16</ipxact:width>
        <ipxact:register>
          <ipxact:name>aReg</ipxact:name>
          <ipxact:displayName>my8_bit_reg</ipxact:displayName>
          <ipxact:description>A simple 8 bit register with a status bit and configuration field</ipxact:description>
          <ipxact:addressOffset>'h0</ipxact:addressOffset>
          <ipxact:size>8</ipxact:size>
          <ipxact:field>
            <ipxact:name>aFlag</ipxact:name>
            <ipxact:displayName>status_flag</ipxact:displayName>
            <ipxact:description>Returns a status from hardware</ipxact:description>
            <ipxact:bitOffset>0</ipxact:bitOffset>
            <ipxact:resets>
              <ipxact:reset>
                <ipxact:value>'h0</ipxact:value>
              </ipxact:reset>
            </ipxact:resets>
            <ipxact:bitWidth>1</ipxact:bitWidth>
            <ipxact:volatile>true</ipxact:volatile>
            <ipxact:access>read-only</ipxact:access>
          </ipxact:field>
          <ipxact:field>
            <ipxact:name>settings</ipxact:name>
            <ipxact:displayName>config</ipxact:displayName>
            <ipxact:description>configuration bits that are fully accessible in hardware and software</ipxact:description>
            <ipxact:bitOffset>4</ipxact:bitOffset>
            <ipxact:resets>
              <ipxact:reset>
                <ipxact:value>'h0</ipxact:value>
              </ipxact:reset>
            </ipxact:resets>
            <ipxact:bitWidth>3</ipxact:bitWidth>
            <ipxact:volatile>true</ipxact:volatile>
            <ipxact:access>read-write</ipxact:access>
          </ipxact:field>
        </ipxact:register>
```

General Purpose-built HSI Authoring Solutions

Jamie, Lead Engineer

General Purpose-built HSI Authoring Solutions

- generalized solution
- languages developed for HSI Authoring
  - Standards – SystemRDL 2.0
  - Proprietary – CSRSpec
- industry standard input and output formats
  - reuse
  - maintenance
  - consistency across multiple views/organizations
  - RTL, documentation, DV, software

Jamie, Lead Engineer

General Purpose-built HSI Authoring Solutions

- CSRSpec and SystemRDL 2.0 common characteristics
  - human readable
  - structured
  - “programming constructs”
  - portable
  - reusable
  - “look” like a hardware language

SystemRDL 2.0 Example

```
reg a16BitReg {
   regwidth = 16;
   field {} aBit = 0;
};

addrmap top {
      reg aReg {
         name = "my8_bit_reg";
         desc = "A simple 8 bit register with a status bit and configuration field";
         regwidth = 8;

            field {
               name = "status_flag";
               desc = "Returns a status from hardware";
               fieldwidth = 1;
               sw = r;
               hw = w;
            } aFlag = 0;

            field {
               name = "config";
               desc = "configuration bits that are fully accessible in hardware and software";
               fieldwidth = 3;
               sw = rw;
               hw = rw;
            } settings[6:4] = 0;
      } aReg;

      a16BitReg   bReg;

      regfile {
         aReg        aReg;
         a16BitReg   bReg;
         aReg        cReg[15];
      } myRegFile;
};
```

CSRSpec Example

```
register a16BitReg {
    property register_size = 16 bits;
    field {
        property input_value = true;
        property output_port = true;
        property reset_value = 0;
    } aBit;
};

addressmap {
    register aReg {
        property title = "my8_bit_reg";
        property description = "A simple 8 bit register with a status bit and configuration field";
        property register_size = 1 bytes;

        field {
            property title = "status_flag";
            property description = "Returns a status from hardware";
            property width = 1;
            property write_access = false;
            property input_value = true;
            property reset_value = 0;
        } aFlag;

        field {
            property title = "config";
            property description = "configuration bits that are fully accessible in hardware and software";
            property width = 3;
            property input_value = true;
            property output_port = true;
            property reset_value = 0;
        } [6:4] settings;
    } aReg;

    a16BitReg   bReg;

    group {
        aReg        aReg;
        a16BitReg   bReg;
        aReg        cReg[15];
    } myRegFile;
} top;
```

General Purpose-built HSI Authoring Solutions

- CSRSpec vs. SystemRDL 2.0 properties

CSRSpec	SystemRDL 2.0
addressmap	addrmap
register	reg
field	field
memory	mem
group	regfile
port	signal

General Purpose-built HSI Authoring Solutions

- CSRSpec vs. SystemRDL 2.0 objects

Item	CSRSpec	SystemRDL 2.0
syntax	**property** *title* = “cool title”;	*name* = “cool title”;
properties	over 280 (like *title* above)	over 85 (like *name* above)
types	boolean, integer, number, reference, string	boolean, integer, number, reference, string
field types	configuration, constant, status, interrupt, counter, writedata	configuration, constant, status, interrupt, counter
bus	supported	none *
user-defined properties	none	supported

* possible via UDP which then becomes a “proprietary” solution

General Purpose-built HSI Authoring Solutions

- CSRSpec vs. SystemRDL 2.0 other differences

Item	CSRSpec	SystemRDL 2.0
property assignment	static	static and dynamic
constraint objects	none	supported
byte accessible bus wide registers	supported	not supported
field and signal array	supported	not supported
field alias	supported	not supported

General Purpose-built HSI Authoring Solutions

If you hand a good spec to three providers, you'll get three variations back in return.

The way you know your spec is worthwhile is that you can live with the differences between them.

If it's worth caring about, it's worth writing down.

- Seth Godin (https://seths.blog/2022/07/a-good-spec/)

Lessons Learned

Jamie, Lead Engineer

Questions ?

Attribution

- “Jamie” Photo by Mapbox on Unsplash