Designing IP and Device Driver for RISC-V

Anupam Bakshi
CEO
Agnisys, Inc.

SiFive Tech Symposium July 2019

Who all are associated with SoC Development?

Typical SoC Development is a High Risk Game!

Requires too many resources – engineers, machines, licenses, …

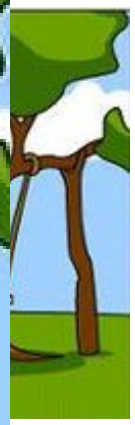
Requires too much time

Requires too many steps

Business/ Marketing specification

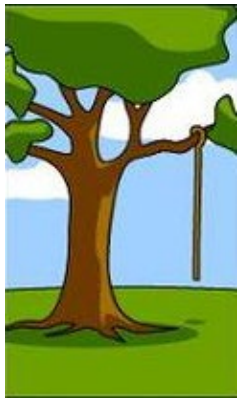
RTL Designer implements thi

Customer Requirement



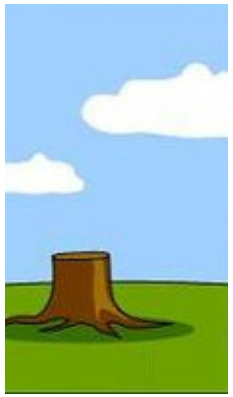
ion team es this

Firmware team writes Device Drivers



Software Team develops this

Tech Pub team documents this



Customer gets this!

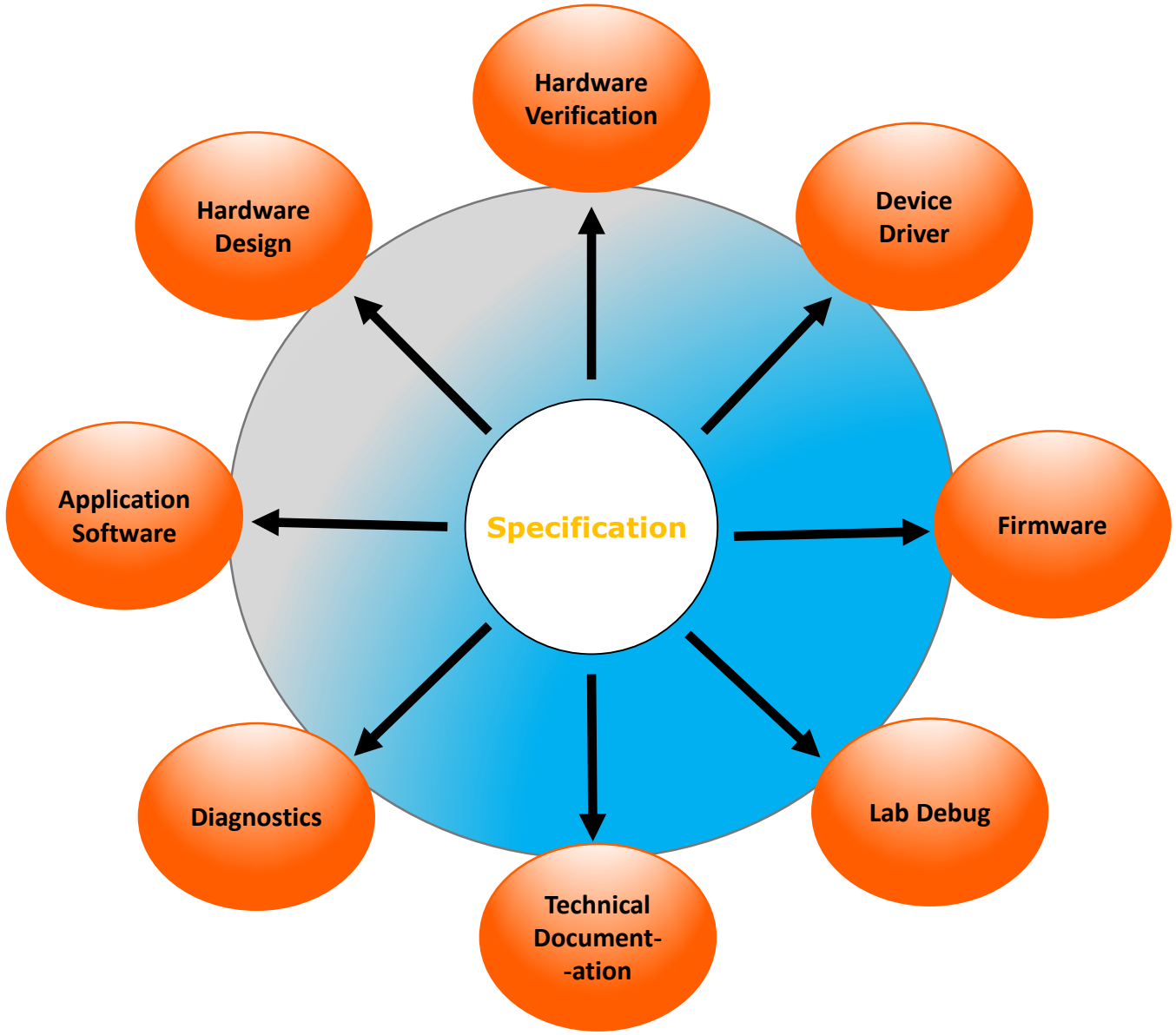
Has this ever happened to you?

AGNISYS
SYSTEM DEVELOPMENT WITH CERTAINTY

Ways to reduce Time to Market for SoC

- Use proven technology
  - Stand on the shoulders of giants – like RISC-V, SiFive
  - Use proven IP
- Use proven methodology
  - Like template based design
- Automate as much as possible
- Eliminate wasteful work
  - Manual work
  - Duplication of work
- Use Single Source of Information
  - Do not duplicate sources of information

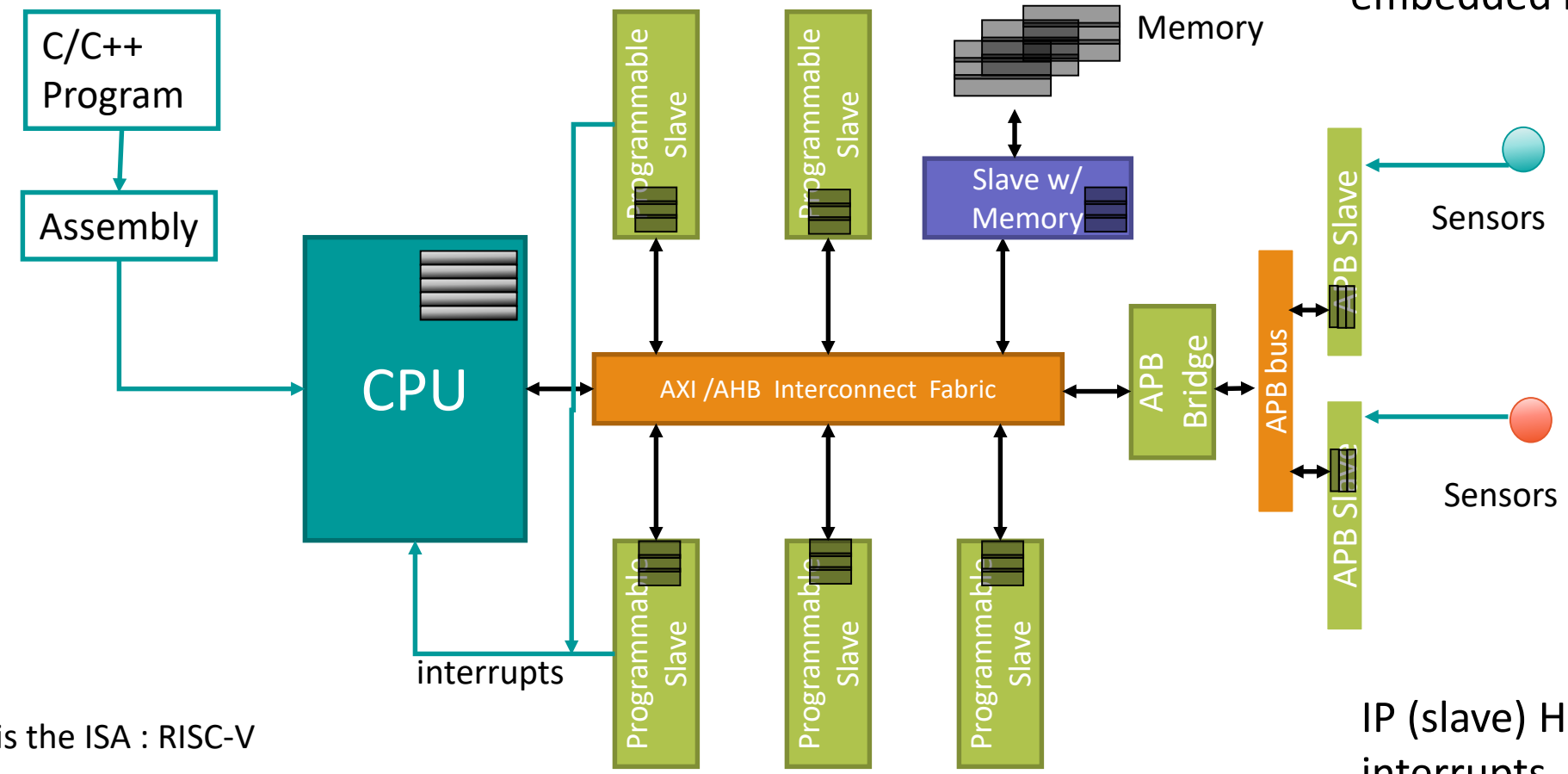
Single Source for Information



AGNISYS
SYSTEM DEVELOPMENT WITH CERTAINTY

Components of a Typical SoC

The slaves are programmed by reading/writing to the embedded register



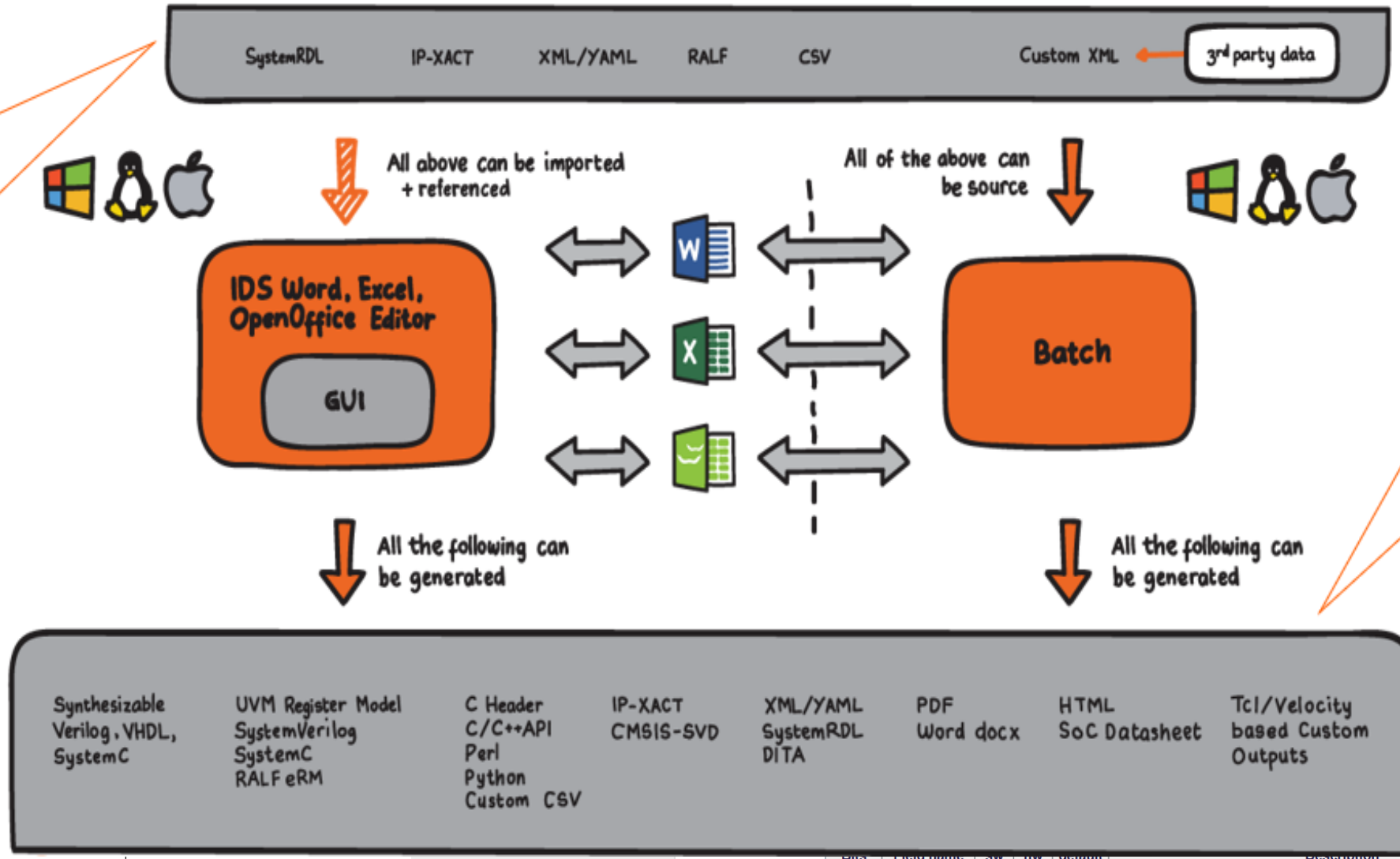
CPU HSI is the ISA : RISC-V

IP (slave) HSI are the registers & interrupts

AGNISYS

Solution for IP Design

Specifications can be written in **MS Word, MS Excel, LibreOffice** or text-based industry standard formats such as **SystemRDL, RALF or IP-XACT.**



IDesignSpec captures simple as well as **special registers, signals, interrupts, and generates synthesizable RTL, UVM model, C/C++ Headers, HTML or PDF.**

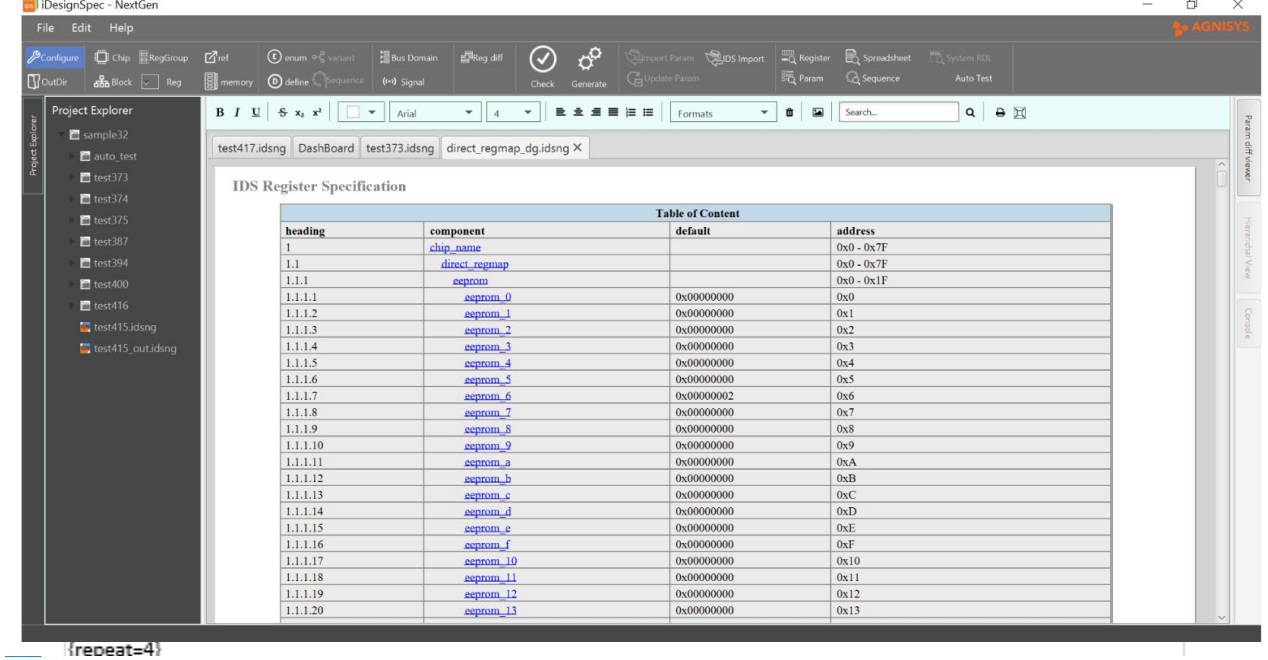
BUS

Bits	Field name	sw	hw	default	Description
0:0	polarity	rw	rw	0x0	External interrupt polarity for interrupt source ID S: 0: Active-high interrupt 1: Active-low interrupt
1:1	type1	rw	rw	0x0	External interrupt type for interrupt source ID S: 0: Level-triggered interrupt 1: Edge-triggered interrupt

AGNISYS
SYSTEM DEVELOPMENT WITH CERTAINTY

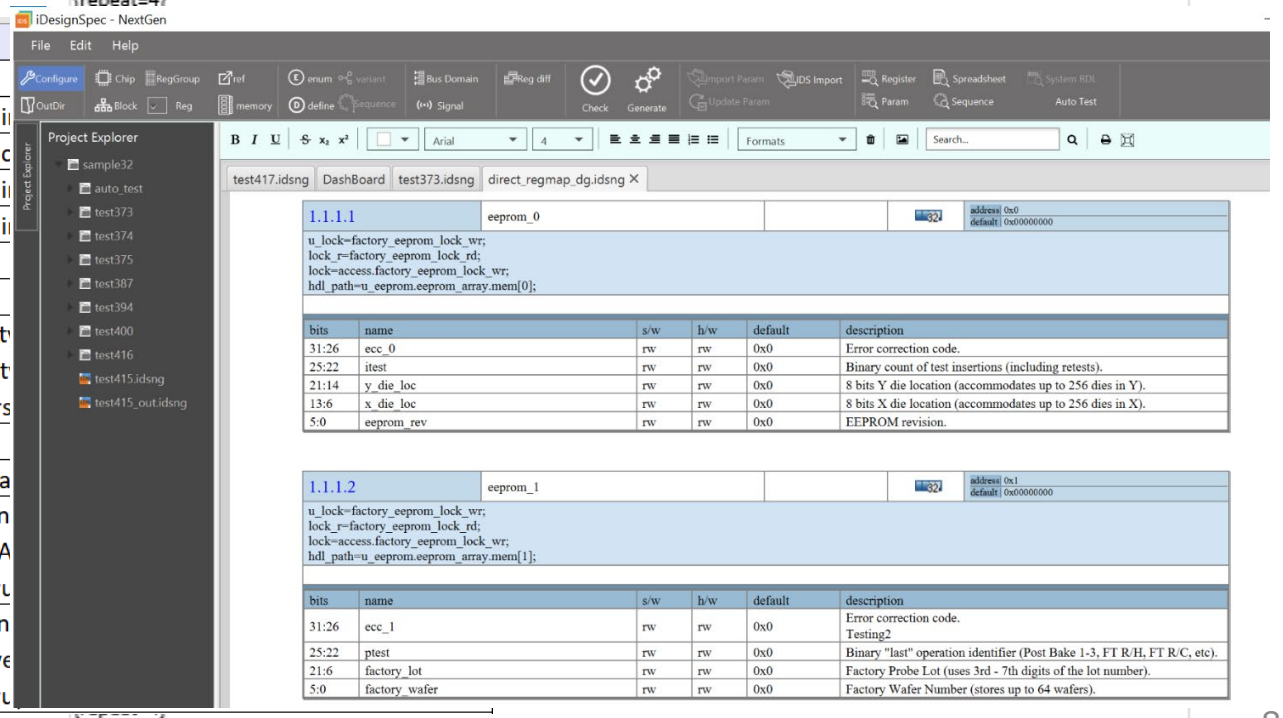
Single Spec for CSR

- Word Document
- Excel Document
- Desktop Application



heading	component	default	address
1	chip_name		0x0 - 0x7F
1.1	direct_regmap		0x0 - 0x7F
1.1.1	eeprom		0x0 - 0x1F
1.1.1.1	eeprom_0	0x00000000	0x0
1.1.1.2	eeprom_1	0x00000000	0x1
1.1.1.3	eeprom_2	0x00000000	0x2
1.1.1.4	eeprom_3	0x00000000	0x3
1.1.1.5	eeprom_4	0x00000000	0x4
1.1.1.6	eeprom_5	0x00000000	0x5
1.1.1.7	eeprom_6	0x00000002	0x6
1.1.1.8	eeprom_7	0x00000000	0x7
1.1.1.9	eeprom_8	0x00000000	0x8
1.1.1.10	eeprom_9	0x00000000	0x9
1.1.1.11	eeprom_a	0x00000000	0xA
1.1.1.12	eeprom_b	0x00000000	0xB
1.1.1.13	eeprom_c	0x00000000	0xC
1.1.1.14	eeprom_d	0x00000000	0xD
1.1.1.15	eeprom_e	0x00000000	0xE
1.1.1.16	eeprom_f	0x00000000	0xF
1.1.1.17	eeprom_10	0x00000000	0x10
1.1.1.18	eeprom_11	0x00000000	0x11
1.1.1.19	eeprom_12	0x00000000	0x12
1.1.1.20	eeprom_13	0x00000000	0x13

block	register	field	bits	sw access	hw access	field default	
SweRV							
	mip						Machi
		mceip	[30:30]	ro	ro	0	Correc
		meip	[11:11]	ro	ro	0	Machi
		mtip	[7:7]	ro	ro	0	Machi
	mpiccfg						
		priord	0	rw	rw	0	Priorit priorit Revers
	meigwctrl						{repea
		polarity	[0:0]	rw	rw	0	Extern S: 0: A interru
		type1	[1:1]	rw	rw	0	Extern 0: Leve interru



bits	name	s/w	h/w	default	description
31:26	ecc_0	rw	rw	0x0	Error correction code.
25:22	itest	rw	rw	0x0	Binary count of test insertions (including retests).
21:14	y_die_loc	rw	rw	0x0	8 bits Y die location (accommodates up to 256 dies in Y).
13:6	x_die_loc	rw	rw	0x0	8 bits X die location (accommodates up to 256 dies in X).
5:0	eeprom_rev	rw	rw	0x0	EEPROM revision.

bits	name	s/w	h/w	default	description
31:26	ecc_1	rw	rw	0x0	Error correction code. Testing2
25:22	ptest	rw	rw	0x0	Binary "last" operation identifier (Post Bake 1-3, FT R/H, FT R/C, etc).
21:6	factory_lot	rw	rw	0x0	Factory Probe Lot (uses 3rd - 7th digits of the lot number).
5:0	factory_wafer	rw	rw	0x0	Factory Wafer Number (stores up to 64 wafers).

bits	name	s/w	h/w	default	description
0	reset	Rw	Ro	0	{rtl.hw_w1p=true} Resets the washer

Solution for IP Design for RISC-V

- Library
  - Library for standard parts
    - GPIO
    - PWM
    - Timer
    - UART
    - I2C/SPI
  - Highly customizable and configurable
  - Template based approach
  - Register Bus abstraction
    - **TileLink**, AMBA-AXI, AHB, APB, AHB3LITE, SPI, I2C, OCP, AVALON

1 washer	washer	Block	0x0
offset	external	size	

{rtl_wrapper=true}

controller_if	controller_if	Signals

{rtl_wrapper=true}

name	port type	Description
rinse[3:0]	out	
water_level[3:0]	in	
imbalance[3:0]	in	
weight[3:0][7:0]	in	
busy[3:0]	out	

Value	Category
0-16	Light
16-32	Standard
32+	Heavy

Value	Meaning
0	Machine available
1	Wash in progress

1.1 spin_count_reg	spin_count_reg	Reg.	0x0, 0x4...
offset	external	size 32	default 0x00302010

{repeat=4}{resetsignal=spin_reset}

bits	name	s/w	h/w	default	description
7:0	light	Ro	Ro	0x10	{counter=decr}
15:8	standard	Ro	Ro	0x20	{counter=decr}
23:16	heavy	Ro	Ro	0x30	{counter=decr}

1.2 dry_count_reg	dry_count_reg	Reg.	0x10, 0x14...
Offset	external	size 32	default 0x00302010

{repeat=4}{resetsignal=dry_reset}

bits	name	s/w	h/w	default	description
7:0	light	Ro	Ro	0x10	{counter=decr}
15:8	standard	Ro	Ro	0x20	{counter=decr}
23:16	heavy	Ro	Ro	0x30	{counter=decr}

AGNISYS
SYSTEM DEVELOPMENT WITH CERTAINTY

GPIO example

- Two stages : Configuration and Customization

Hyper-Parameters

Name	Default	Description
NUM_GPIO	**8**	**Number of gpio pins**
NUM_SRC	8	Number of input Sources
USE_BLOCK_EN	True	Use Block Enable

Hardware Interface

Signals

name	port type	description
clk	input	Free running clock for gpio block
GPIO[NUM_GPIO-1:0]	inout	Number of pins will be controlled by the parameter NUM_SRC
irq	output	Interrupt port
Bus ports	Input/output	Configured by the user from configuration

Register Map

User_Custom_Reg — Reg.

offset		external		size	32

This register is added by user for some custom functionality.

bits	name	s/w	h/w	default	description
0	F1	Rw	Ro	0	User can use this field in any complex "next" or "assign" expression within the GPIO block. Standard GPIO register fields can be used in expressions here. User can also use the Hyper-Parameters here e.g. to set the number of repeats for this register. For example, User can use a counter to count the number of times a certain interrupt occurs on a GPIO pin.

NUM_GPIO-1:0	data	ro	wo	0

gpio_out — Reg.

offset		external	

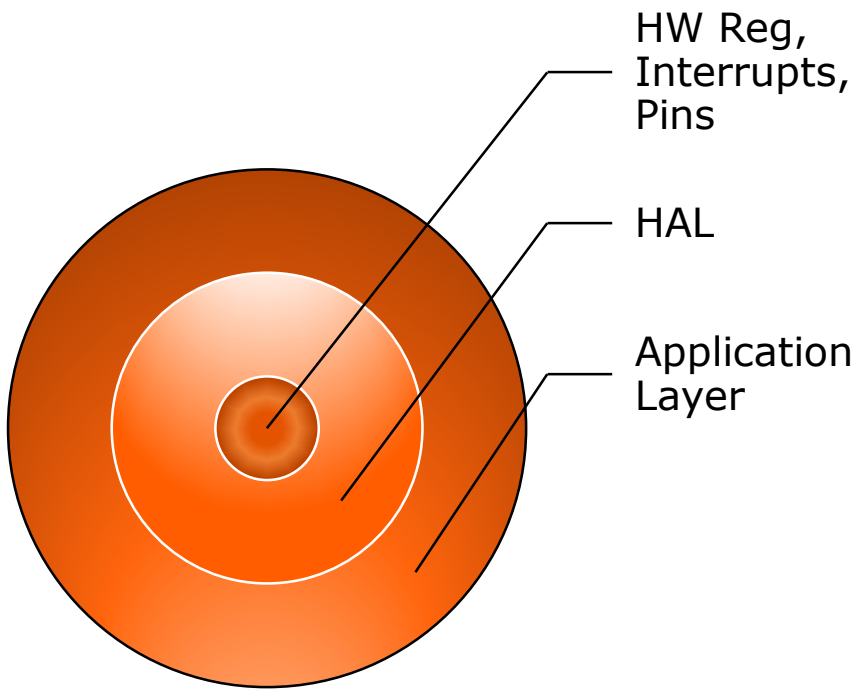
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

bits	name	s/w	h/w	default	description
NUM_GPIO-1:0	data	rw	ro	0	Drive the gpio pin when selected as output and ext_src_sel is 0

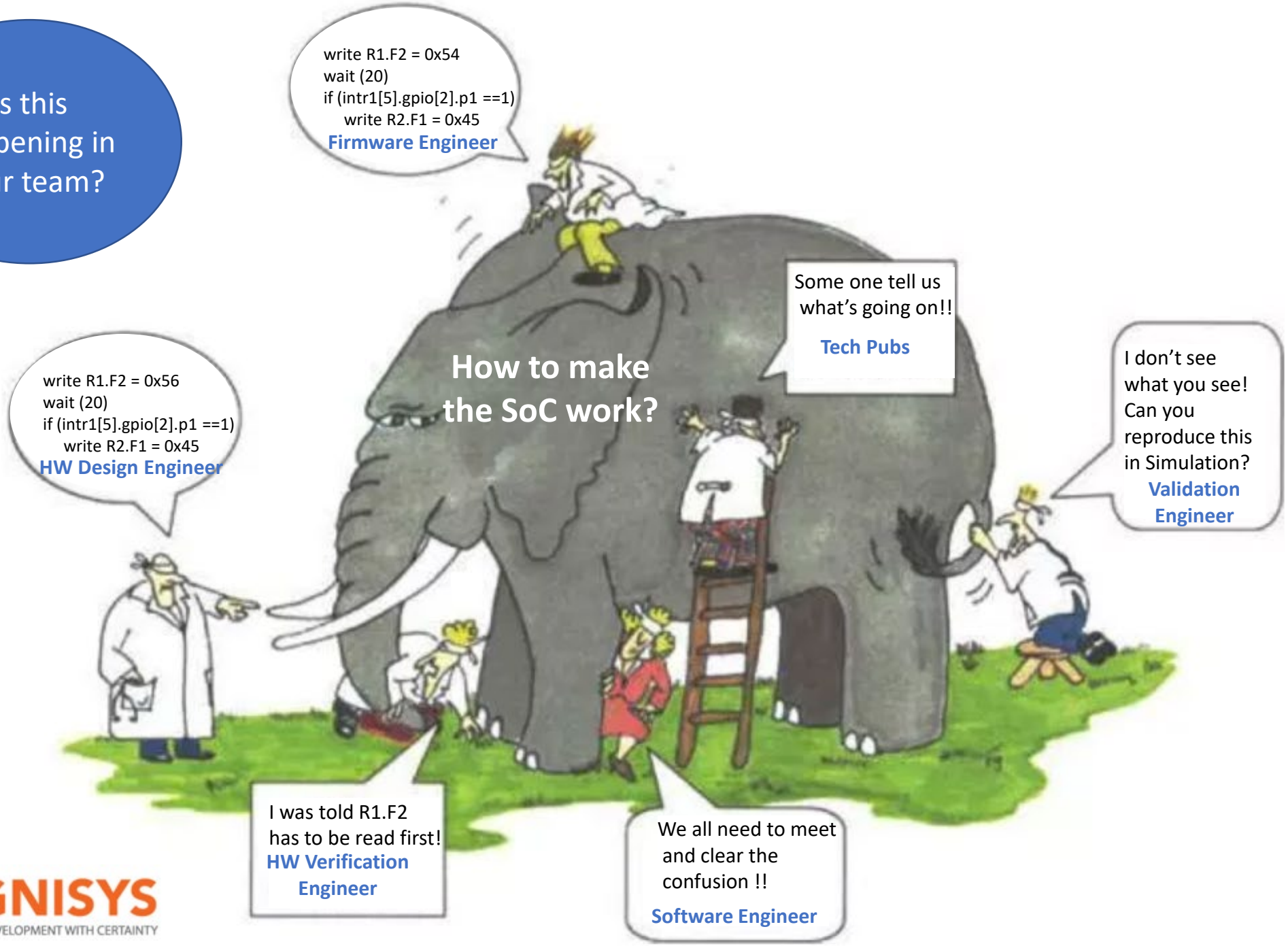
AGNISYS
SYSTEM DEVELOPMENT WITH CERTAINTY

Solution for Device Driver

- How does the CPU interact with the IPs?
  - Through the Hardware Abstraction Layer (HAL)
- Creating Device Driver is always very tedious
- Device Drive Languages
  - C
  - ASM
- Functionality of the SoC Designed by HW team
  - But used By
    - Verification/Emulation Team
    - Firmware Team
    - Validation Team

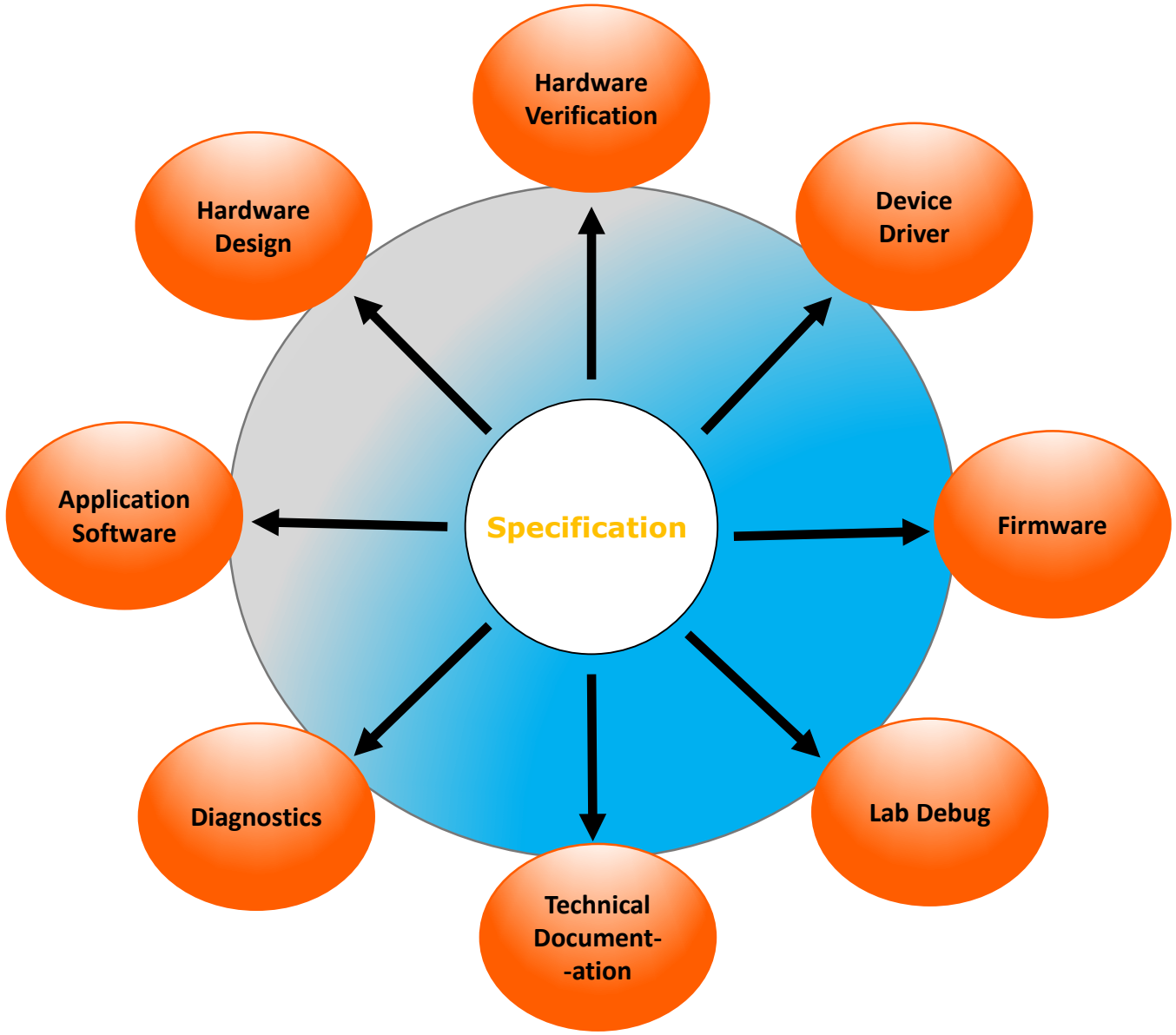


Is this happening in your team?



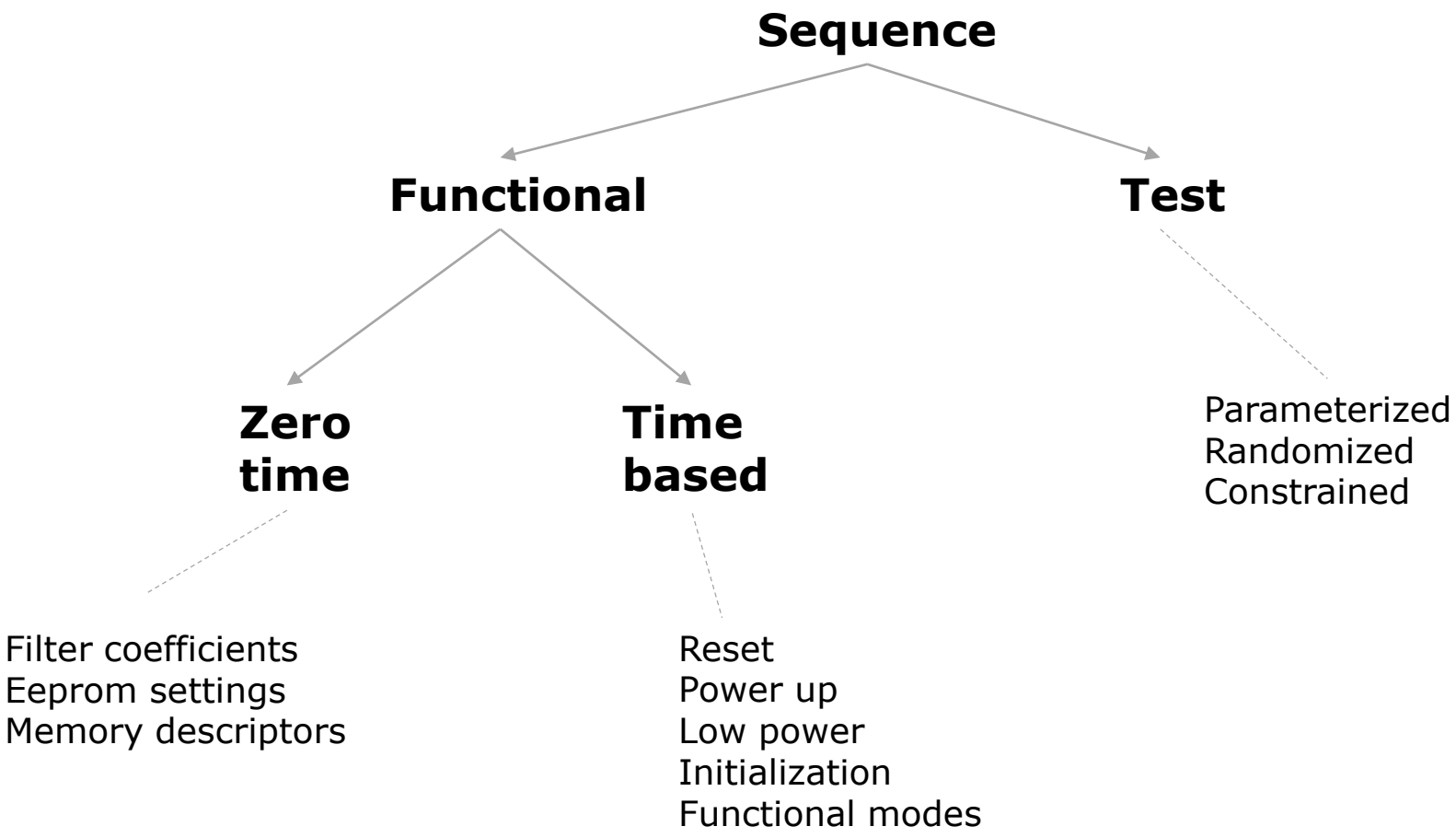
AGNISYS
SYSTEM DEVELOPMENT WITH CERTAINTY

Single Source for Information



AGNISYS
SYSTEM DEVELOPMENT WITH CERTAINTY

Types of Sequence



AGNISYS
SYSTEM DEVELOPMENT WITH CERTAINTY

Solution for Device Drive for RISC-V

- Have a single source for Sequences
- Create low level API using a format everyone understands
  - Python
  - Excel
- Generate all artifacts from the source
  - Verification : UVM sequences, PSS
  - Emulation/Prototype : SystemVerilog tasks, Matlab
  - Firmware : C API , RISC-V ASM
  - Validation : C or SV
- Support for a wide range of ISA base and extensions
  - I/E for integer instructions
  - M for multiplication and division
  - C for compact instruction

Single Source configuration of GPIO

- Configuring GPIO registers using Python
- Equivalent Configuration sequences for multiple environments
    - SV-UVM for Verification
    - C for Firmware
    - Portable Stimulus

```python
from spec.reg

class gpio_dr

    def drive

        gpio_

        out_d

        data_

        iss.w

        iss.w

        gpio_

        iss.w

        iss.r
```

```
class uvm_gpio_cfg_seq extends uvm_reg_sequence#(uvm_sequence#(uvm_reg_item));
    `uvm_object_utils(uvm_gpio_cfg_seq)

int  gpio_cfg( ) {

unsigned int consolidated_temp_value = 0;

int  GPIO_REG_gpio_in;
int flag;
int port_enb = 1 ;
int out_data = 0xAAAA5555 ;
int data_in = 0 ;

//-----------------------------------------------------------------------------
/*---- enable pin as output*/
//-----------------------------------------------------------------------------

consolidated_temp_value = ((port_enb << GPIO_REG_GPIO_PIN_CFG_GPIO_OUT_EN_OFFSET) & GPIO_REG_GPIO_PIN_CFG_GP
consolidated_temp_value = ((GPIO_REG_GPIO_PIN_CFG_GPIO_OUT_EN_MASK) & consolidated_temp_value) | (~(GPIO_REG
REG_WRITE(GPIO_REG_gpio_pin_cfg_ADDRESS,consolidated_temp_value);

//-----------------------------------------------------------------------------
/*---- writing 0xAAAA5555 to output port*/
//-----------------------------------------------------------------------------

REG_WRITE(GPIO_REG_gpio_out_ADDRESS,out_data);

port_enb = 0;

//-----------------------------------------------------------------------------
/*---- enable pin as input*/
//-----------------------------------------------------------------------------

consolidated_temp_value = ((port_enb << GPIO_REG_GPIO_PIN_CFG_GPIO_OUT_EN_OFFSET) & GPIO_REG_GPIO_PIN_CFG_GP
consolidated_temp_value = ((GPIO_REG_GPIO_PIN_CFG_GPIO_OUT_EN_MASK) & consolidated_temp_value) | (~(GPIO_REG
REG_WRITE(GPIO_REG_gpio_pin_cfg_ADDRESS,consolidated_temp_value);

GPIO_REG_gpio_in= REG_READ(GPIO_REG_gpio_in_ADDRESS);


//-----------------------------------------------------------------------------
/*---- reading data from port*/
//-----------------------------------------------------------------------------

data_in = GPIO_REG_gpio_in;
return 0;
}
```

AGNISYS
SYSTEM DEVELOPMENT WITH CERTAINTY

Summary

- Single Source Methodology reduces risk
- Reduces Time to Market by 40-60%
- Fully customizable IPs for RISC-V are available

Go to www.agnisys.com to download SW to get started

AGNISYS
SYSTEM DEVELOPMENT WITH CERTAINTY