AGNISYS

SYSTEM DEVELOPMENT WITH CERTAINTY

Creating portable UVM sequences with ISequenceSpec™

Neena Chandawale
(Host)

Amanjyot Kaur
(Presenter)

Agenda

- Introduction
- Challenges faced and their solution
- ISequenceSpec™ overview
  - Introduction
  - Input flavors- spreadsheet and text-based format
  - Common configuration
- Deep Dive into ISequenceSpec™
  - Sequence language features- conditional and looping statements, function calls, structures, randomization, constraints, assertions, etc.
  - Sequence properties
  - Check- list of checks
- Practical examples
  - Handling indirect registers
  - RISC-V SweRV™
- Benefits
- Conclusion

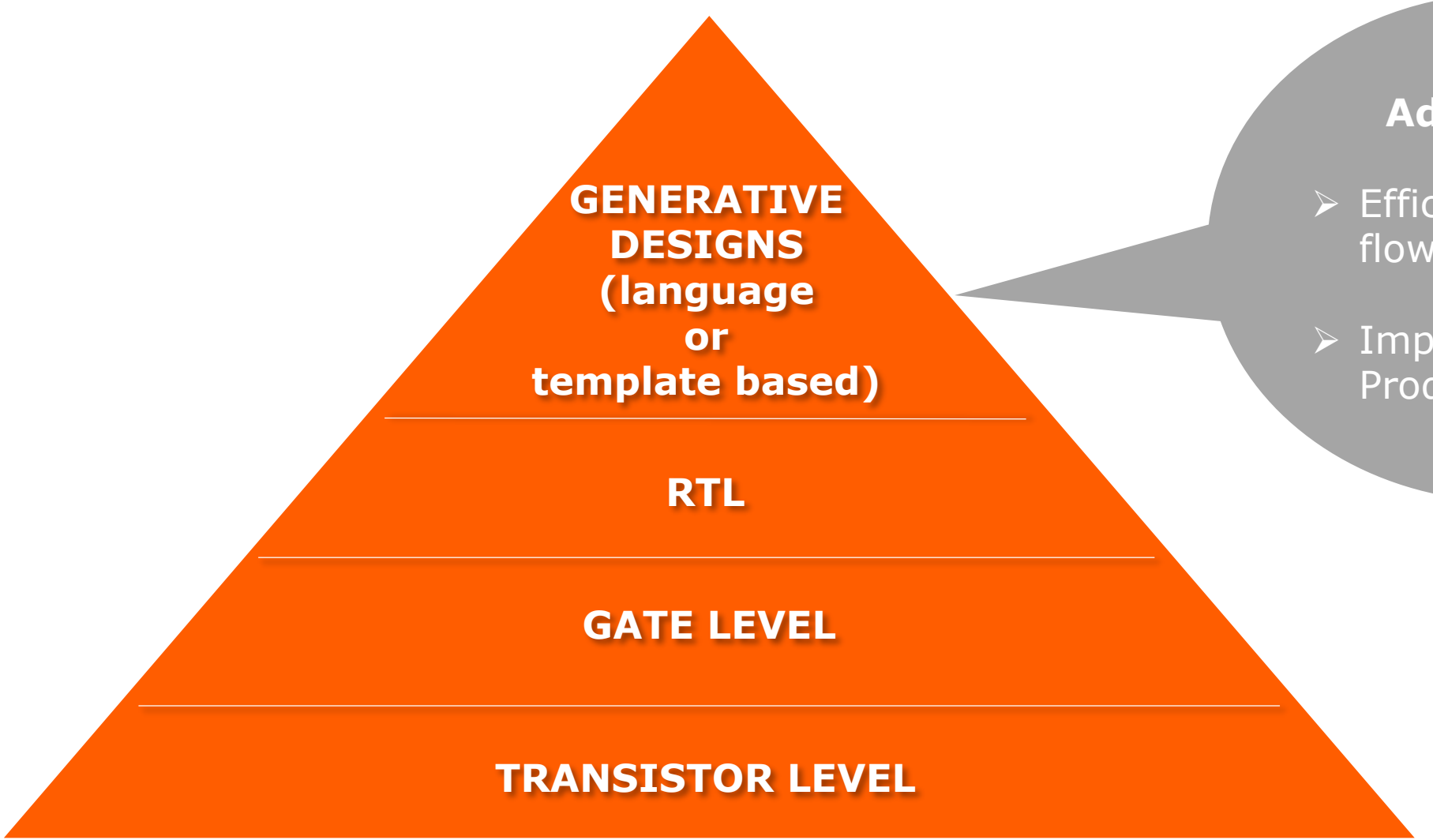
AGNISYS
SYSTEM DEVELOPMENT WITH CERTAINTY

Introduction

History of EDA tools and design methodologies

- Designs evolved from hundreds of transistors to hundreds of billions over last several decades
- Shrinking transistor and increasing transistor count drove various design and verification methodologies
- Abstraction: Key tool in managing ever-increasing complexity and scale of ASIC/SoC designs

Design Abstraction



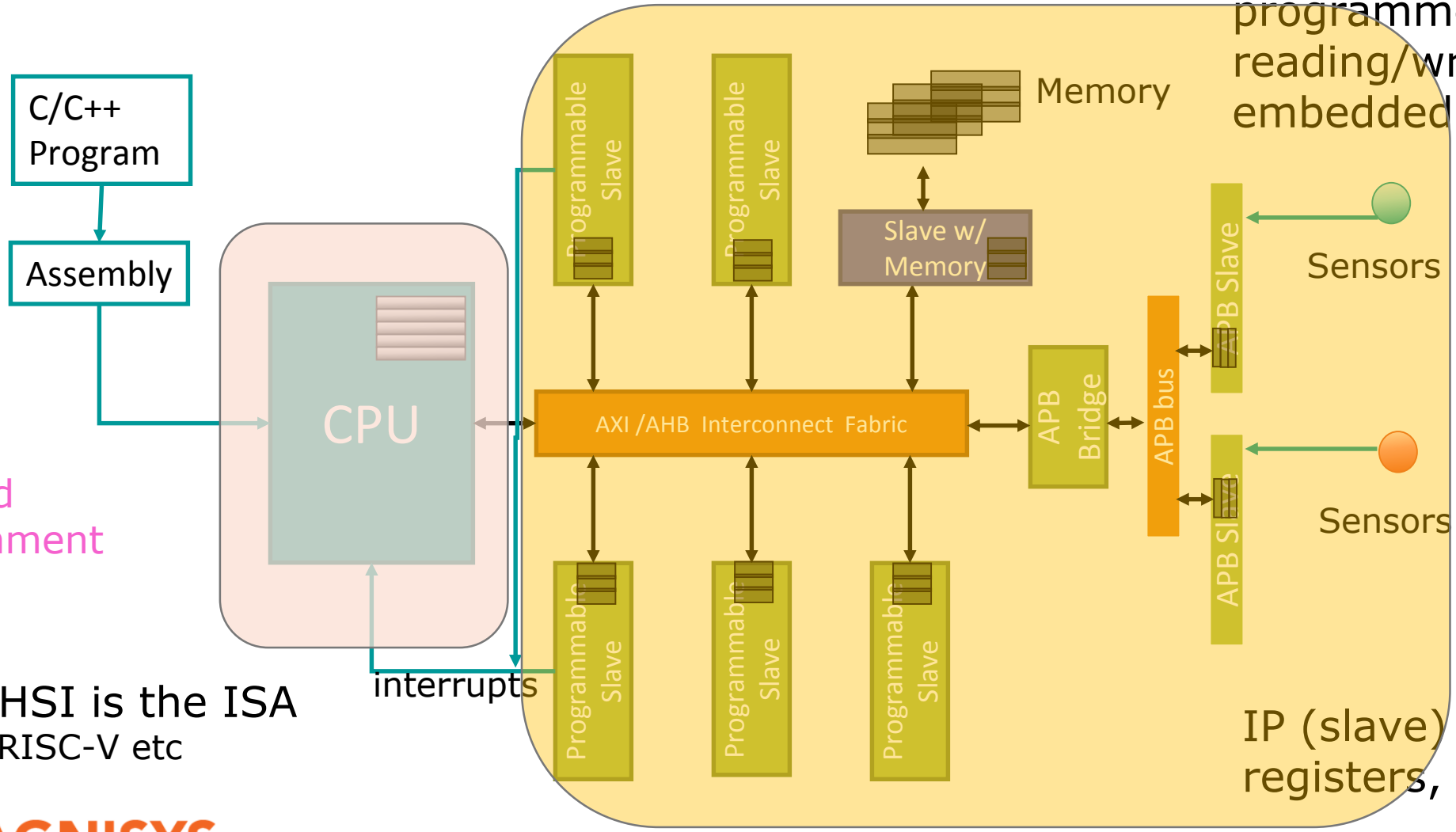
Advantages

- Efficient design flows
- Improves Productivity

AGNISYS
SYSTEM DEVELOPMENT WITH CERTAINTY

CPU & IP Hardware Software Interface

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



C based environment

SV-UVM environment

CPU HSI is the ISA
ARM, RISC-V etc

IP (slave) HSI are the registers, interrupts

AGNISYS
SYSTEM DEVELOPMENT WITH CERTAINTY

What's a Sequence?

- A list of steps that need to be executed in order
- A flow-chart or an algorithm
- Used for configuration setting (programming) or test

Functional

Reset
Power up
Low power
Initialization
Functional modes
Filter coefficients
Eeprom settings
Memory descriptors

Test

Parameterized
Randomized
Constrained

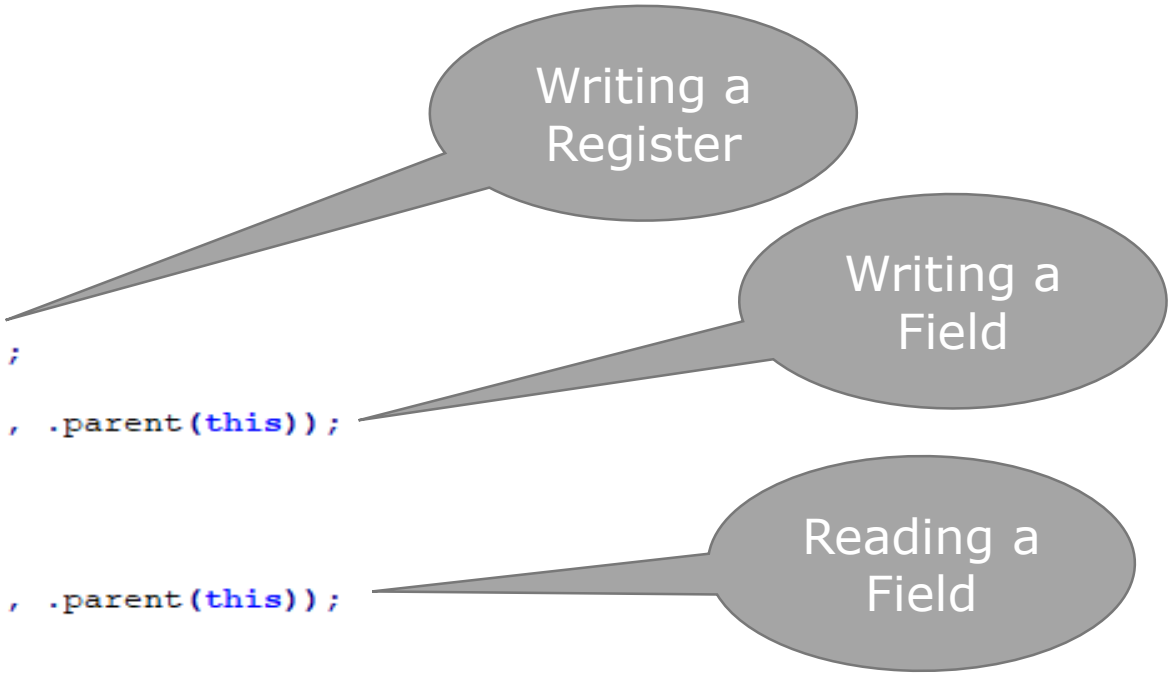
- Sequences are created at various stages of the Design Process
  - Frontend Design Verification SV/UVM
  - Embedded code C/C++
  - Backend Configuration of chips C/C++
  - Backend Testing and Validation C/C++

AGNISYS
SYSTEM DEVELOPMENT WITH CERTAINTY

Example Sequence with HSI

As an example, the code below is a SV task that is manually coded by the user. It shows that HSI is a critical part of a sequence to achieve a certain behavior in the target device.

```
task xmit( int noOfTxTrans);

    for ( int count = 0 ; count < noOfTxTrans;count++ )
    begin

        if (1 && count == LineRate && rdValue == ClockFreq)
            begin
                lvar = InitialWriteData + count;

                rm.TXDATA.write(status, lvar, .parent(this));

                rm.CONTROL.TXEN.write(status, uartControl[1], .parent(this));
            end
            while  (rdValue == 0)
            begin

                rm.STATUS.TXDONE.read(status, STATUS_TXDONE , .parent(this));

                rdValue=STATUS_TXDONE;
            end
        end

endtask
```



AGNISYS
SYSTEM DEVELOPMENT WITH CERTAINTY

Sequences are built on registers, memories, pins

- Sequences contain
  - Register / Field Writes
  - Register / Field Reads
  - Pin Manipulation Commands
  - Wait / Function calls, sub sequence calls

- Information about Registers/Memories can be in any format
  - IP-XACT
  - SystemRDL
  - Word / Excel
  - Text files

Challenges faced and their solution

Challenges faced with Sequences

- Duplicate work in Verification, Firmware & Validation groups
- A sequence works on one platform and not on other
- No way to create the same debug environment on multiple platforms
- Sequence is not clear or may not be well documented
- Sequence contain register data that can be in any standard or custom format

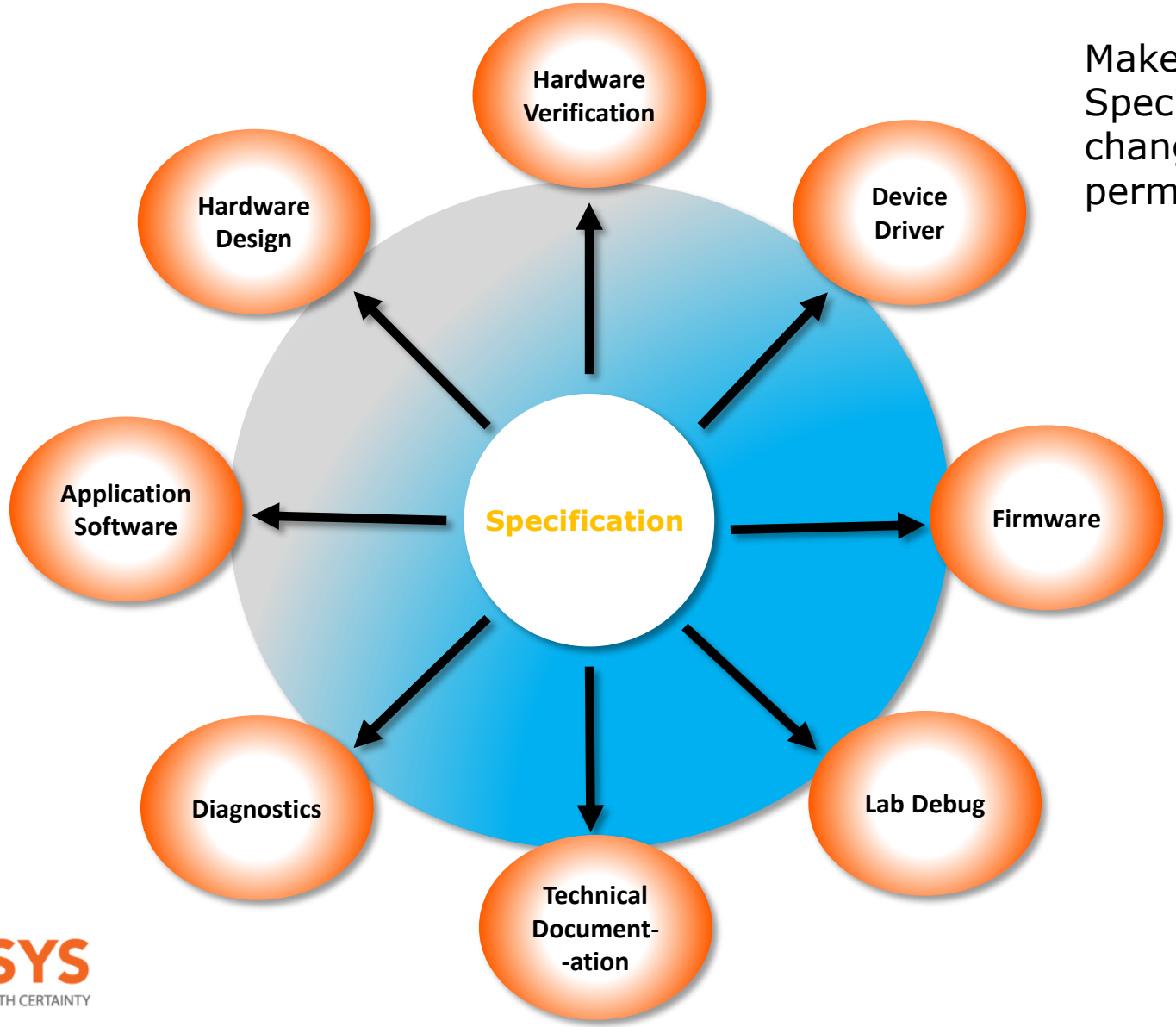
AGNISYS
SYSTEM DEVELOPMENT WITH CERTAINTY

Proposed Solution

Create a Golden Spec for Implementation-Level Sequences and Auto-Generate the Code

- Capturing the golden specification for sequences will need the following capabilities:
  - Control flow
  - High level of abstraction devoid of implementation detail
  - Access to hierarchical register data for SoC, Subsystem and IPs
  - Access to pins, signals and interfaces
  - High level execution of arbitrary transactions
  - Deal with timing differently based on the target
  - Hierarchy of sequence and base address of the DUT

Specification Driven Development



Make changes to the Specification and have the change automatically permeate to all views

AGNISYS
SYSTEM DEVELOPMENT WITH CERTAINTY

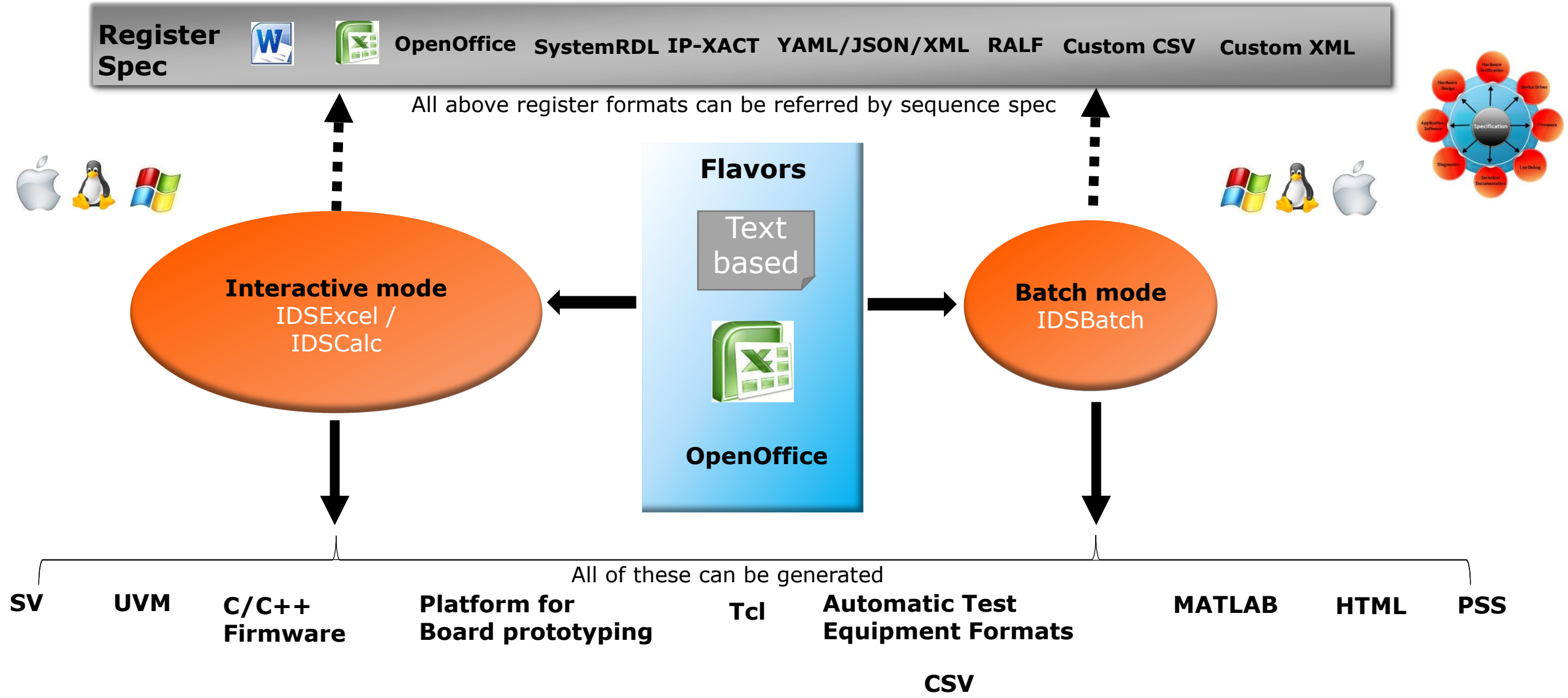
What does a sequence generation need

- Create a variety of output formats
- Flexibility in how Read/Writes are generated
- Output specific
    - UVM : font door/back door / peekpoke
    - C/C++ : Consolidated read/write
    - Test/Validation : Multiple test sites – for testing multiple chips simultaneously
    - Target platform may not support hierarchy, loops, variables

ISequenceSpec™ overview

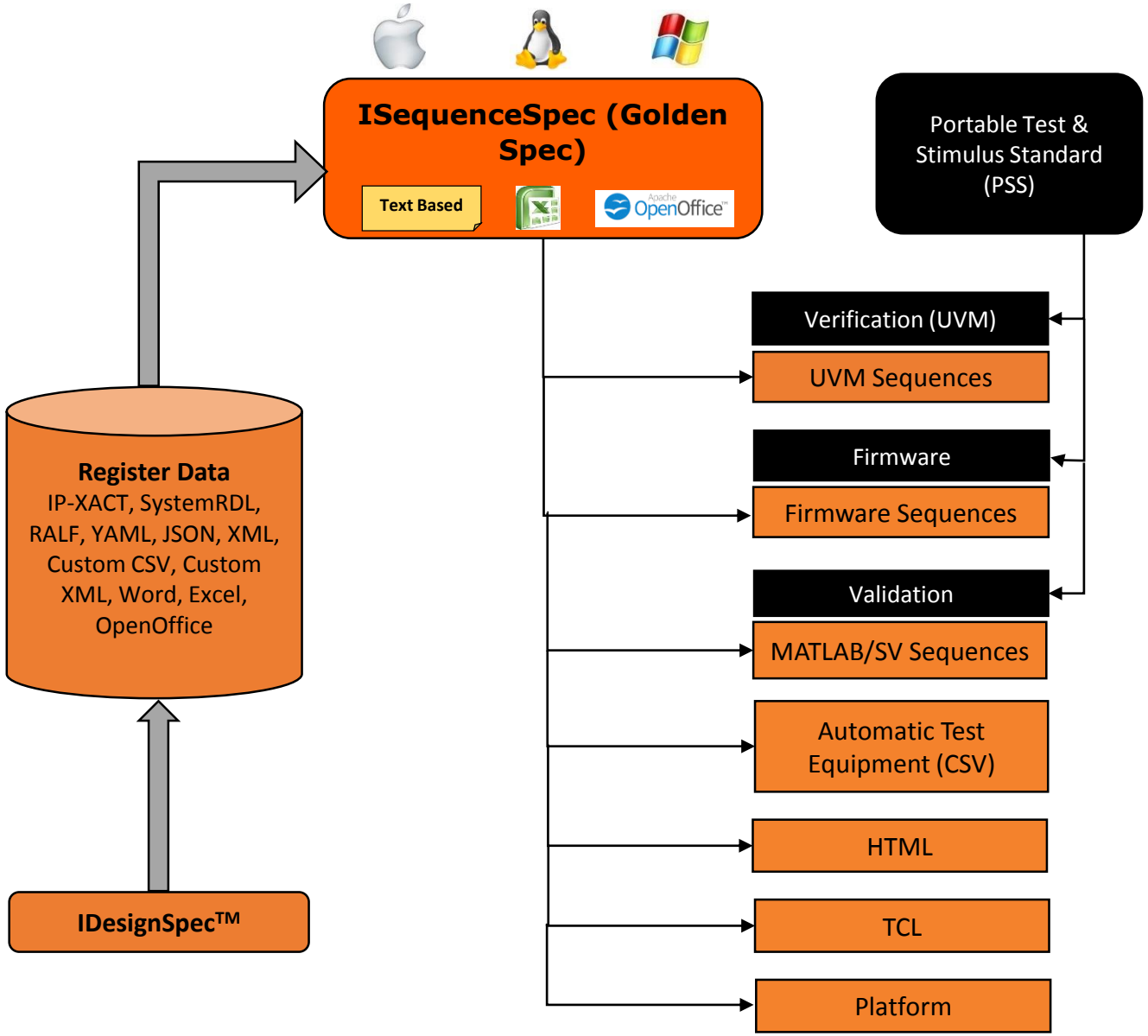
Introducing ISequenceSpec™

AGNISYS
SYSTEM DEVELOPMENT WITH CERTAINTY



Introducing ISequenceSpec™ - Continued

- ISequenceSpec enables users to describe the programming and test sequences of a device and automatically generate sequences ready to use from an early design and verification stage to post silicon validation
- Centralize creation of sequences from a single specification and generate various output formats for multiple SoC teams
  - UVM, System Verilog, C, TCL, CSV or MATLAB
  - HTML
  - Platform
- Specify portable sequences for multiple IPs at a higher level in-sync with the register specification
- Use register descriptions in standard formats such as IP-XACT, SystemRDL, RALF or leverage IDesignSpec™ integrated flow to use the register data
- Add-on to IDesignSpec



AGNISYS
SYSTEM DEVELOPMENT WITH CERTAINTY

Input Flavors

- Spreadsheet format
  - OpenOffice
  - IDSExcel
- Text based format
  - Python

riscv.py

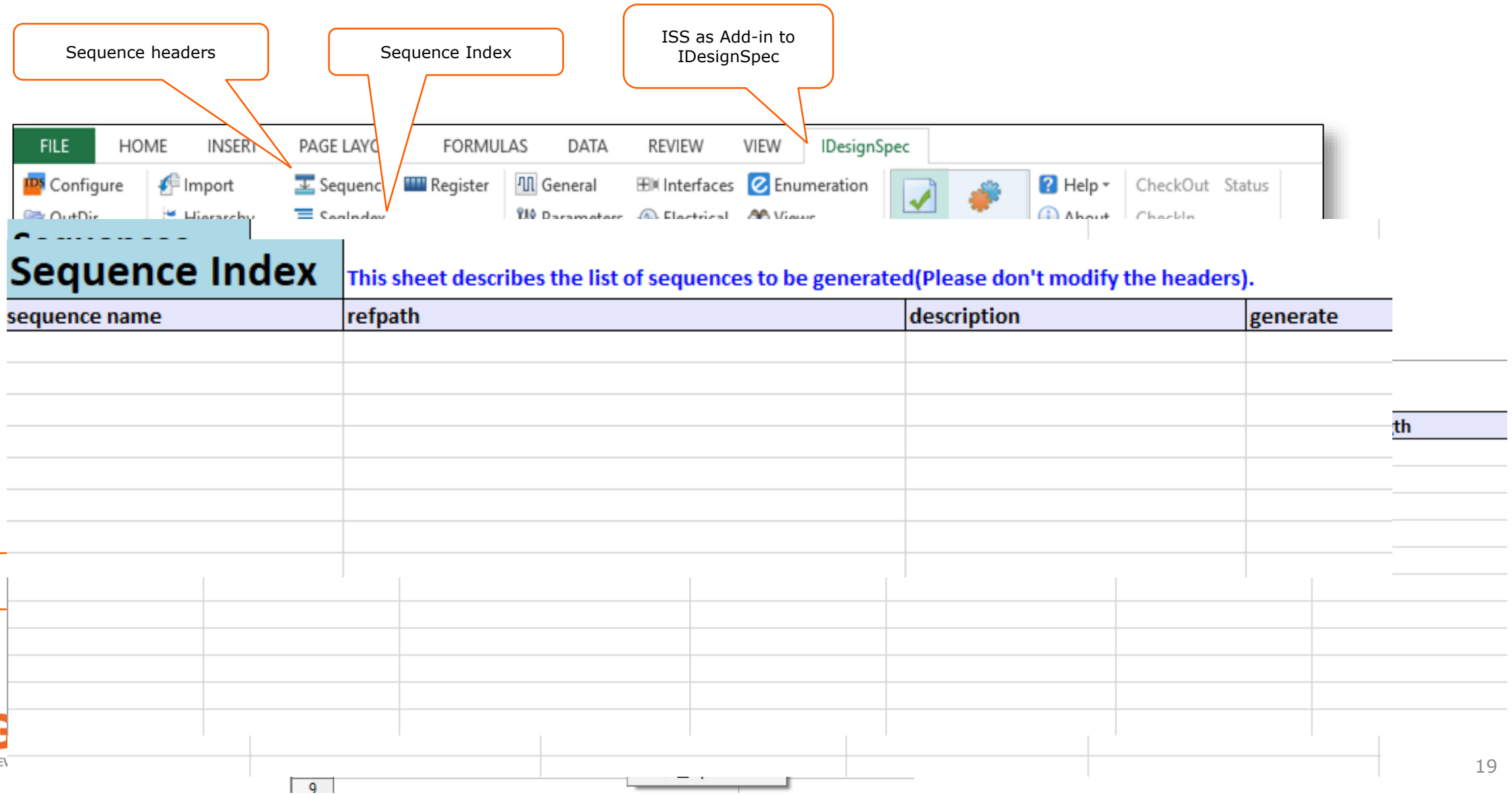
```
from regMap.registermap.registermap import ISequenceSpec as iss
from regMap.block import block
block =block()
iss = iss()

class sequences:
    def initial_seq(self,ip = 'riscv_ip.docx',desc = ''):
        set_value = iss.argument(1,'set_value','set value')
        clear_value = iss.argument(0,'clear_value','clear value')
        a_size = iss.argument(2,'a_size','size of external sources')
        base_addri = iss.constant(0x0,'base_addr','base address')
        ext_srci = iss.variable({0,0},'ext_src','external sourece array')
        reset_sigi = iss.variable(1,'reset_sig')
        exintsrc_reqi = iss.variable(0,'exintsrc_req')
        if (reset_sigi==1):
            iss.write(mpiccfg.priord,1,'Configured the priority order')
            while (exintsrc_reqi<a_size):
                iss.write(meigwctrl[exintsrc_reqi].polarity,set_value,'"polarity" field set of "meigwctrl" register')
                iss.write(meigwctrl[exintsrc_reqi].typel,set_value,'"type" field set of "meigwctrl" register')
                iss.write(meigwclr[exintsrc_reqi],clear_value,'Cleared the IP bit by writing to the gateways "meigwclr"')
                iss.read(ext_srci[exintsrc_reqi],1)
                if (ext_srci[exintsrc_reqi]==1):
                    iss.write(meie[exintsrc_reqi].inten,set_value,'Enabled interrupts for the appropriate external interrupt sources ')
                elif (ext_srci[exintsrc_reqi]==2):
                    iss.write(meie[exintsrc_reqi].inten,set_value,'Enabled interrupts for the appropriate external interrupt sources ')
                exintsrc_reqi=exintsrc_reqi+1

            iss.write(meivt.base,base_addri,'Base address of external vectored interrupt address table is set ')
            iss.write(meipt.prithresh,1,'Priority threshold is set')
            iss.write(meicidpl.clidpri,0,'Initialized the nesting priority thresholds ')
            iss.write(meicurpl.currpri,0)

    def SweRV_seq(self,ip = 'riscv_ip.docx',desc = ''):


        set_val = iss.argument(1,'set_value','set value')
        clear_val = iss.argument(0,'clear_value','clear value')
        size = iss.argument(2,'a_size','size of external sources')
        base_addr = iss.constant(0x0,'base_addr','base address')
        edge_detect = iss.constant(1,'edge_detect','Edge detection')
```

Python file | length : 10,678 lines : 266 | Ln : 202 Col : 35 Sel : 0 | 0 | Unix (LF)

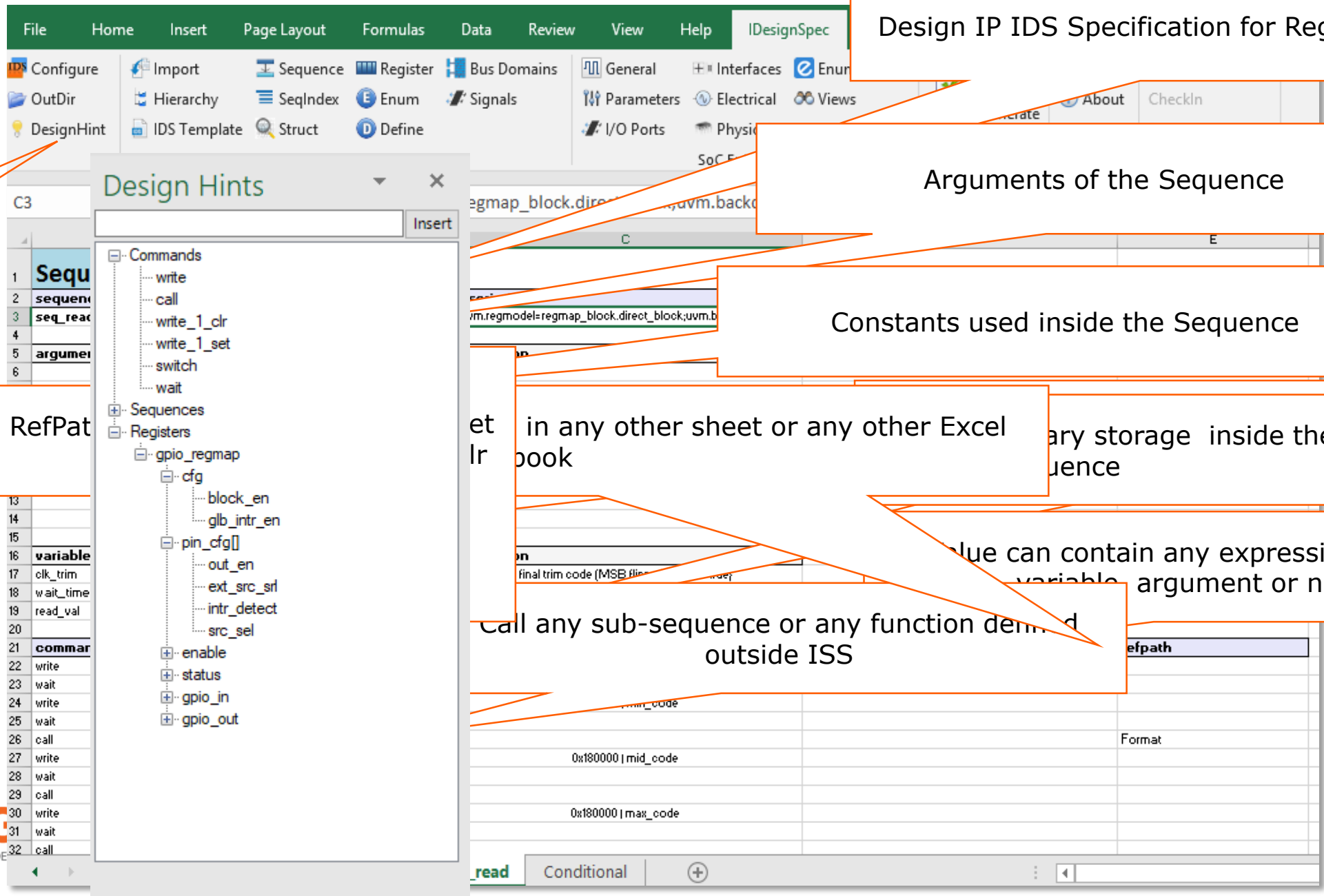
AGNISYS
SYSTEM DEVELOPMENT WITH CERTAINTY

Input Flavors - Continued

GUI : Add-in to Excel Or OpenOffice



Input Flavors - Continued



Design IP IDS Specification for Register Map data

Design Hint

Arguments of the Sequence

Constants used inside the Sequence

RefPat

in any other sheet or any other Excel book

ary storage inside the uence

tion

lue can contain any expression, constant, argument or number

Call any sub-sequence or any function defined outside ISS

Input Flavors - Continued

Text Based (Python) input

Sequence

```
class sequences:
    def initial_seq(self,ip = 'riscv_ip.docx',desc = ''):
        set_value = iss.argument(1,'set_value','set value')
        clear_value = iss.argument(0,'clear_value','clear value')
        a_size = iss.argument(2,'a_size','size of external sources')
        base_addri = iss.constant(0x0,'base_addr','base address')
        ext_srci = iss.variable({0,0},'ext_src','external sourece array')
        reset_sigi = iss.variable(1,'reset_sig')
        exintsrc_reqi = iss.variable(0,'exintsrc_req')
        if (reset_sigi==1):
            iss.write(mpiccfg.priord,1,'Configured the priority order')
            while (exintsrc_reqi<a_size):
                iss.write(meigwctrl[exintsrc_reqi].polarity,set_value,'"polarity" field set of "meigwctrl" register')
                iss.write(meigwctrl[exintsrc_reqi].typel,set_value,'"type" field set of "meigwctrl" register')
                iss.write(meigwclr[exintsrc_reqi],clear_value,'Cleared the IP bit by writing to the gateways "meigwclr"')
                iss.read(ext_srci[exintsrc_reqi],1)
                if (ext_srci[exintsrc_reqi]==1):
                    iss.write(meie[exintsrc_reqi].inten,  value,'Enabled interrupts for the appropriate external interrupt sources ')
                elif (ext_srci[exintsrc_reqi]==2):
                    iss.write(meie[exintsrc_reqi].inten,set_va  led interrupts for the appropriate external interrupt sources ')
                exintsrc_reqi=exintsrc_reqi+1

            iss.write(meivt.base,base_addri,'Base address of external vec                ddress table is set ')
            iss.write(meipt.prithresh,1,'Priority threshold is
            iss.write(meicidpl.clidpri,0,'Initialized the nesti
            iss.write(meicurpl.currpri,0)
```

Arguments:
name = iss.argument (‘value’, 'name', 'description')

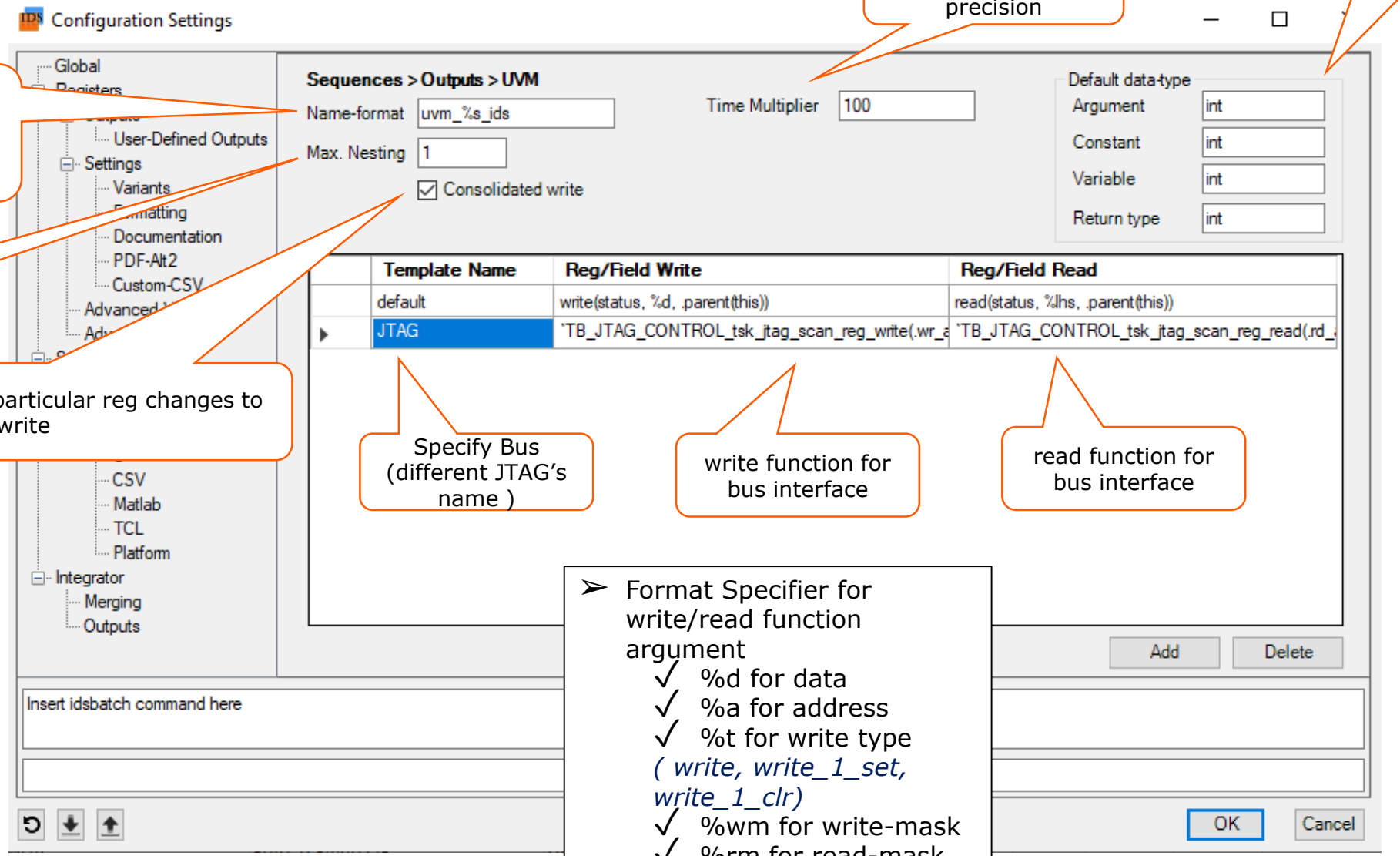
Constants:
name = iss.constant (‘value’, 'name', 'description')

Variables:
name = iss.variable (‘value’, 'name', 'description')

Read-Write Statements:
Read Statement: iss.read(‘register/field name’,’variable name’,'desc')
Write Statement: iss.write((‘register/field name’,’variable/constant value’,'desc')

AGNISYS
SYSTEM DEVELOPMENT WITH CERTAINTY

Common Configuration



User defined class name in UVM output

Max nesting

Multiple field write of a particular reg changes to one write

User suitable time multiplier for precision

User defined data types

Specify Bus (different JTAG's name)

write function for bus interface

read function for bus interface

- Format Specifier for write/read function argument
  - ✓ %d for data
  - ✓ %a for address
  - ✓ %t for write type *(write, write_1_set, write_1_clr)*
  - ✓ %wm for write-mask
  - ✓ %rm for read-mask
  - ✓ %vm for volatile mask

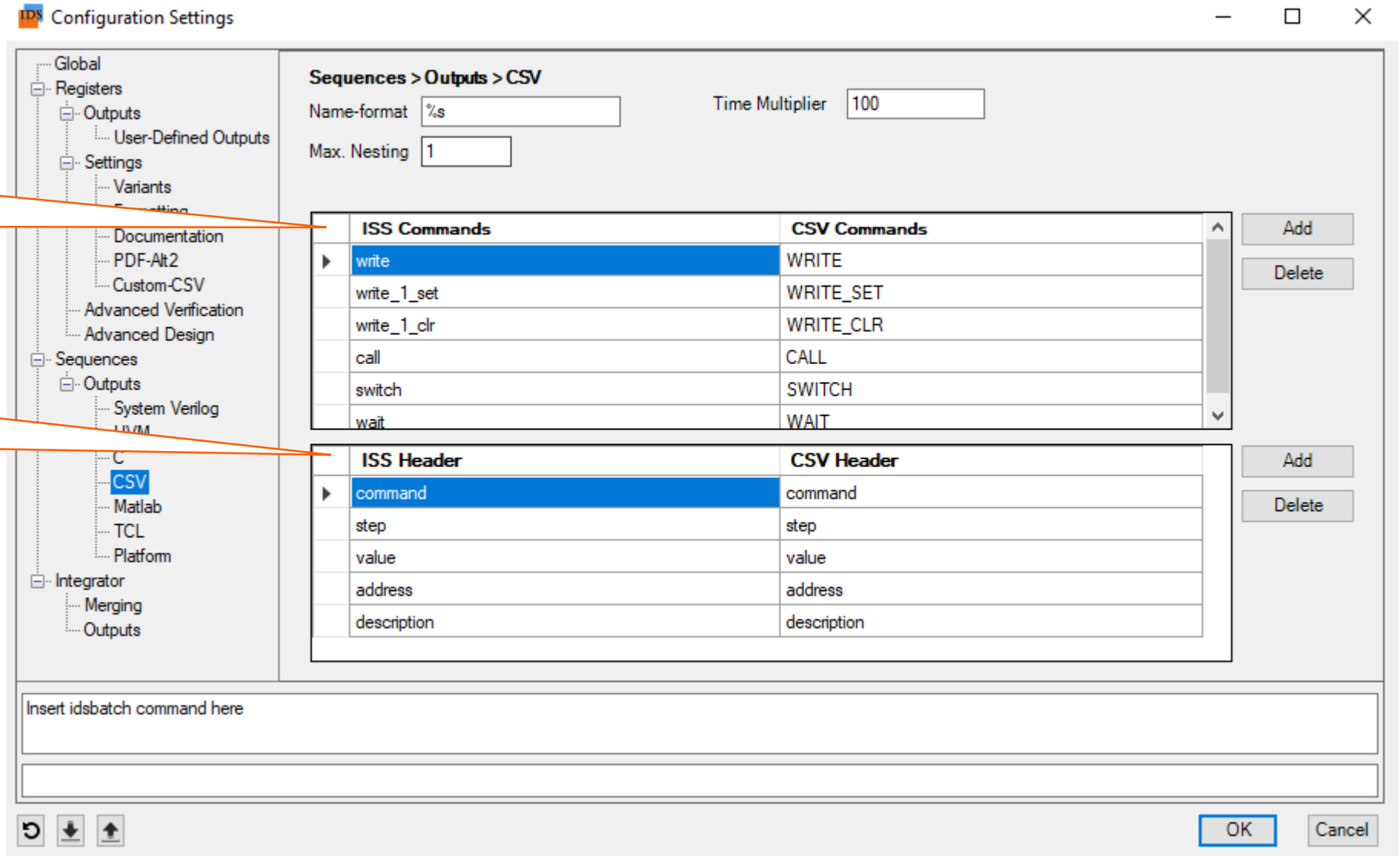
AGNISYS
SYSTEM DEVELOPMENT WITH CERTAINTY

Common Configuration- Continued

CSV Configuration zone

Map all ISS commands to custom commands for output

Map all ISS Header to Custom headers for output



AGNISYS
SYSTEM DEVELOPMENT WITH CERTAINTY

Common Configuration- Continued

Text Based (Python) input

Configure : All GUI configure options are available in textual mode too

Add configure table at the top of the file

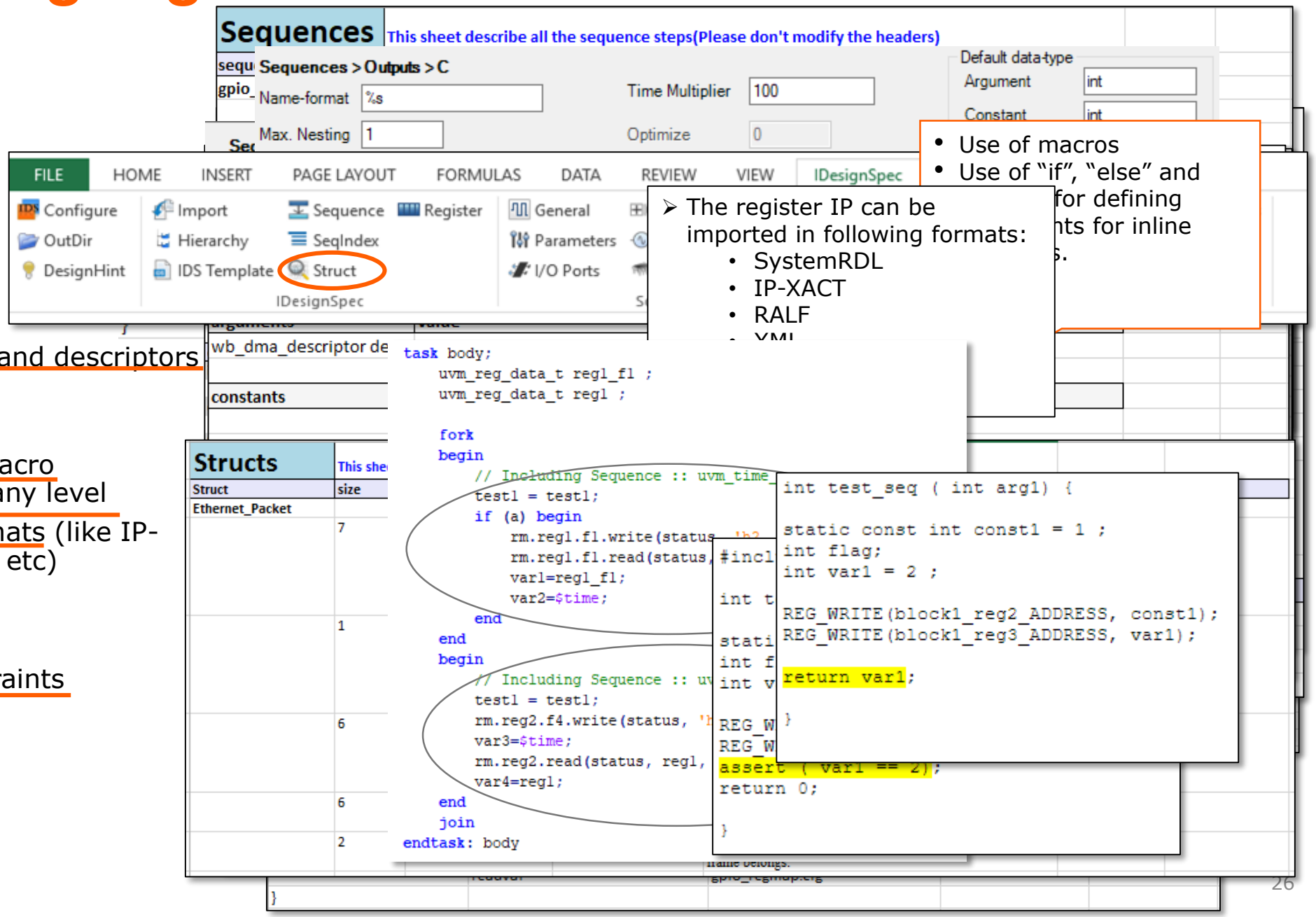
```
configure = {
        'output' :{
            'sv' : {
                'name_format' : '%s',
                'max_nesting':'2',
                'mout': 'false',
                'consolidated_write':'false',
                'Timemultiplier'    :{
                    'time_unit':'1ns',
                    'time_precision':'1ns'
                },
                'default data-type':{
                    'argument':'int',
                    'constant':'int',
                    'variable':'int',
                },
                'template' :{
                    'reg_read' : 'read_mirror(%a)',
                    'reg_write': 'write_mirror(%a, %d, 0, 0)'
                }
            },
```

AGNISYS
SYSTEM DEVELOPMENT WITH CERTAINTY

Deep Dive into ISequenceSpec™

Sequence Language Features

- Looping
  - For loop
  - While loop
- Condition
  - If - else condition
- Wait statement
- Switch command
- External function call
- Structs to define packets and descriptors (UVM and header)
- Multiple structures
- Powerful referencing of macro sequences and IP's from any level
- IP's in different input formats (like IP-XACT, System-RDL, RALF, etc)
- Fork join
- Assertions
- Randomization and constraints
- Comments
- Returning values
- Consolidated read/write



AGNISYS
SYSTEM DEVELOPMENT WITH CERTAINTY

Sequence Language Features - Continued

Python format

- Looping
  - While loop
- Condition
  - If - else condition
- Wait statement
- External function call
- IP's in different input formats XACT, System-RDL, RALF, etc
- Fork join
- Randomization and constraint

```
while (exintsrc_reqi<a_size):

    iss.write(meigwctrl[exintsrc_reqi].polarity,set_value,'"polarity" field set of "meigwctrl" register')
    iss.write(meigwctrl[exintsrc_reqi].typel,set_value,'"type" field set of "meigwctrl" register')
    d = iss.variable(3,'b','description')
    c = iss.constant(0,'d','description')
    e = iss.variable([1,2,3,4,5])
    iss.fork_join(
    def seq_while(self,ip = 'samples.docx',desc = '{base_address=true}')
        consl=iss.constant(23,'consl','this is constant consl')
        varl=iss.variable(0,rand(),'{constraint="value<100"}')
        var3=iss.variable(1,'var2')
        var2=iss.variable(2,'var2')
        while (consl==23):
            iss.write(Regl.Fl,23,'write to reg2 field 1')
            iss.wait(50)
        iss.write(reg2.f7,100,'{uvm_door=back}')
        iss.write(reg2.f6,'varl?0:1')
        iss.write(reg2.f7,rand(),'{constraint="value<100"}')
        iss.read(varl,reg2.f6)

self.extern_func(set_val,clear_val,size)

if (reset_sigi==1):

    iss.write(mpiccfg.priord,1,'Configured the priority order')
    while (exintsrc_reqi<a_size):

                    },
        'regmodel_template' :{
                'read' : 'read(status, %lhs, .parent(this))',
                'write': 'write(status, %d, .parent(this))'
                }
```

'while' loop syntax for the input pyhton sequences. Nested loops are also supported

Rand() keyword and constraint is used in write sequence of register.

External function can be call in the sequnces.

AGNISYS
SYSTEM DEVELOPMENT WITH CERTAINTY

Sequence Properties

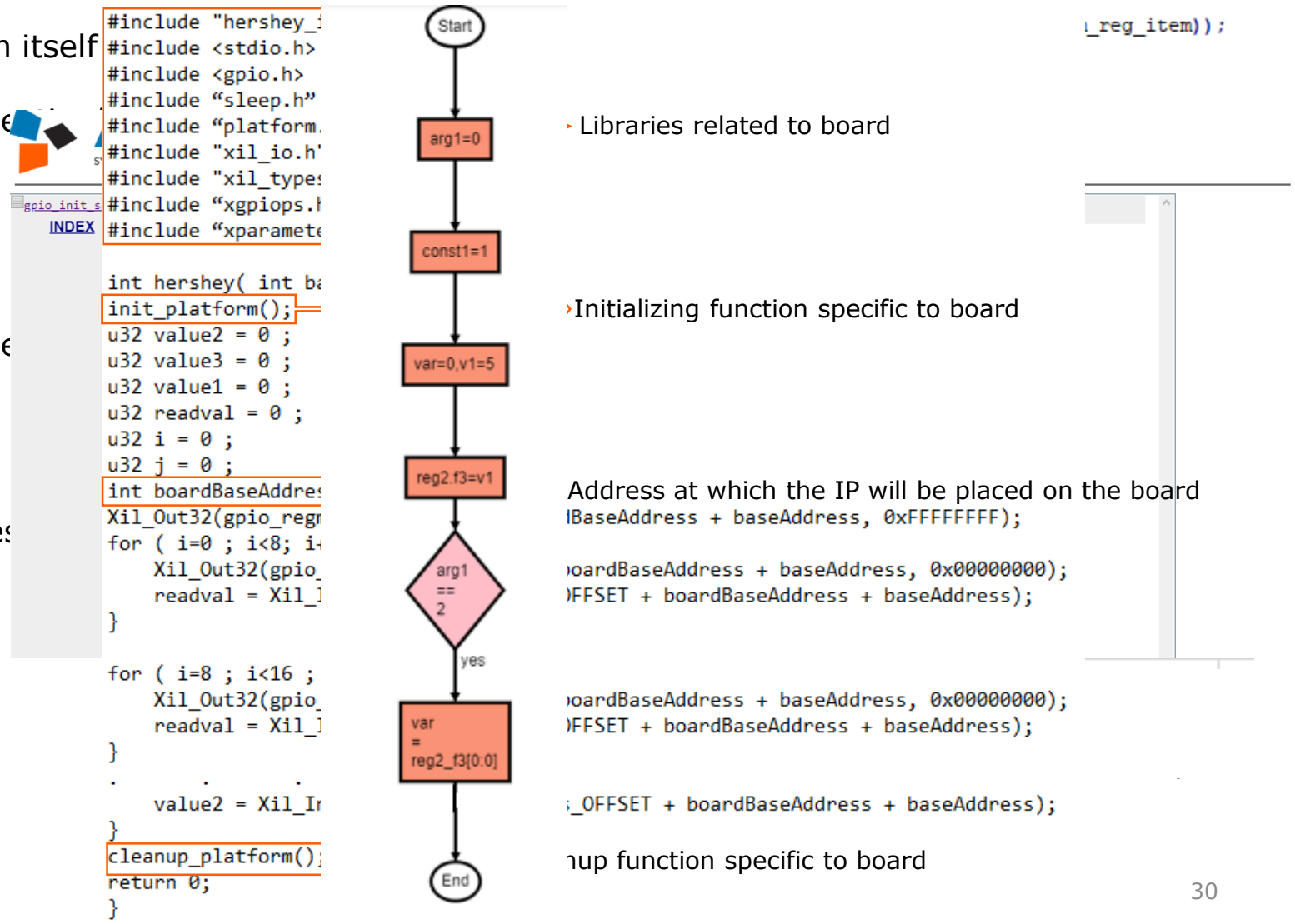
Property Name	Purpose	Usage
format	Specify the fixed-data type format of a field	{format=fixdt(<signed>,<word length> ,<fractional bits>)}
uvm_door	UVM frontdoor and backdoor writes to registers and peek, poke functionality	{uvm_door=frontdoor/backdoor/peekpoke}
uvm.regmodel	Specify the upper hierarchy into the generated regmodel	{uvm.regmodel = <path of register model>}
uvm_reg_map	Accommodates the requirement of multiple reg maps in UVM sequences	{uvm_reg_map=<map name>}
num_sites	Specify the number of sites(ICs) used in the LAB in parallel	{num_sites=<number of sites>}
site	Defines the variable as site-based	{site= true/false}
extern	Specify the variable as Global Variable	{extern=true/false}
base_address	Specify the base address of the memory in C	{base_address=true/false}
board_type	Defines the board in which the user wants to run the sequences for which Signals are used	{board_type=zboard}
signal_map	Defines which signal is connected to specific pins of the defined board	{signal_map = <signal name> : <pin number>}
firmware_guard_band	Header files for sequences and API package will be generated with guard band	{firmware_guard_band=true}
iss_uvm.package	Specify the name of package that is being generated for sequences	{iss_uvm.package=<package name>}
pss_action, pss_component	In case user needs to add extra functionality/scenario in the 'action' block of PSS then user can extend that 'action' block to add that scenario	{pss_action=<action_name>}{pss_component= <component_name>}
concat	To write combination of variables into register/ field	concat(<comma_separated_variables>)
time()	To get value of current time, time() keyword is used	time() keyword under the ‘value’ column
constraints	To apply constraints on variables and registers	{constraint=“value>5”}

Check – list of checks

- Register Map checking
  - Overlaps, name conflicts, address conflicts, and many more
- Register/field name exists
- Value can fit in to the available size of the register/field
- Format is correct
- Correct field access (ro/wo etc.)

Output Formats

- UVM
  - Arguments - Class parameters; user can specify type other than default in the Configuration and Inline
  - Constants - Resolved in the specification itself
  - Variables - Appear in the output
  - Calls - Flattened as indicated by "Max Ne
  - Structs (for UVM)
- Firmware
  - Arguments - Appear in the output
  - Constants - Resolved
  - Variables - Resolved
  - Calls - Flattened as indicated by "Max Ne
- CSV
  - Arguments - Appear in the output
  - Constants - Appear in the output
  - Variables - Resolved
  - Calls - Flattened if indicated by "Max Nes
- System Verilog
- PSS
- MATLAB
- HTML
- Platform



AGNISYS
SYSTEM DEVELOPMENT WITH CERTAINTY

Practical examples

Handling Indirect Registers

Some registers are not directly accessible via a dedicated address. Indirect access of an array of such registers is accomplished by first writing an “index” register with a value that specifies the array's offset, followed by a read or write of a “data” register to obtain or set the value for the register at that specified offset.

Register Data:

1.1 Index	Index	Reg.	0x0

Offset		external		size	32	default	0x00000000

bits	name	s/w	h/w	Default	description
7:0	Index_fld	Rw	Rw	0	

1.2 Data	Data	Reg.	0x4

Offset		external		size	32	default	0x00000000

{index_reg=Index }{ depth=256}

bits	name	s/w	h/w	Default	description
7:0	Data_fld	Rw	Rw	0	

AGNISYS
SYSTEM DEVELOPMENT WITH CERTAINTY

Writing sequences for Indirect Register

Sequences	This sheet describe all the sequence steps(Please don't modify the headers)			
sequence name	**ip**	**description**		
indirect_seq	indirect.docx			
arguments	**value**	**description**		
Arg1	23			
constants	**value**	**description**		
Cons1	8			
variables	**value**	**description**		
Var1	2			
command	**step**	**value**	**description**	**refpath**
write	Data[3]	8		
write	Data[2]	2		
		Data[2]		

AGNISYS
SYSTEM DEVELOPMENT WITH CERTAINTY

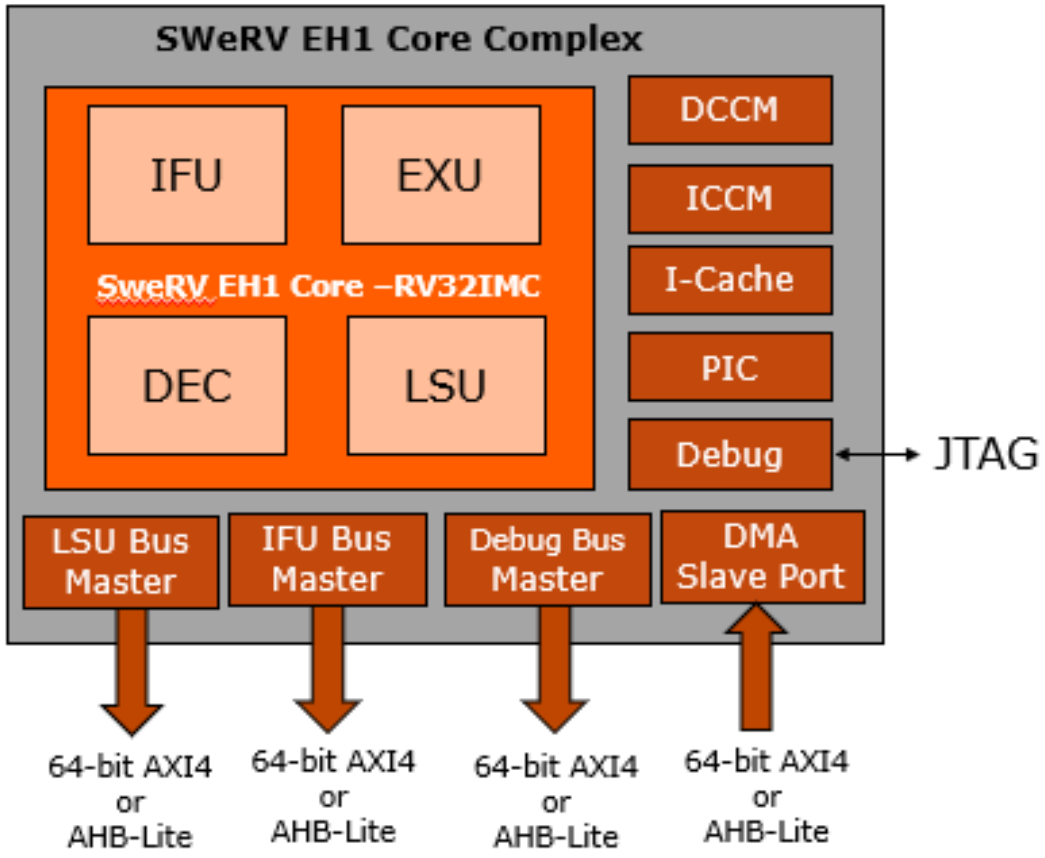
ISS Generated output - Firmware

```
int  indirect_seq( int Arg1 ) {

const int Cons1 = 8 ;
int  block_name_Data;
int  flag_block_name_Data;
int flag;
int Var1 = 2 ;



REG_WRITE(block_name_Index_ADDRESS,12);

REG_WRITE(block_name_Data_ADDRESS,8);

REG_WRITE(block_name_Index_ADDRESS,8);

REG_WRITE(block_name_Data_ADDRESS,2);

REG_WRITE(block_name_Index_ADDRESS,8);
block_name_Data= REG_READ(block_name_Data_ADDRESS);

return 0;
}
```

AGNISYS
SYSTEM DEVELOPMENT WITH CERTAINTY

RISC-V SweRV™

- RISC-V SweRV™ has been contributed by Western Digital Corporation, Licensed under Apache-2.0
- Following block diagram depicts the core complex and its functional blocks.



AGNISYS
SYSTEM DEVELOPMENT WITH CERTAINTY

Source: https://github.com/westerndigitalcorporation/swerv_eh1/blob/master/docs/RISC-V_SweRV_EH1_PRM.pdf

RISC-V SweRV™ - Continued

5.5 Theory of Operation

5.5.1 Initialization

The control registers must be initialized in the following sequence:

1. Configure the priority order by writing the *priord* bit of the `mpiccfg` register.
2. For each configurable gateway *ext_src*, set the polarity (*polarity* field) and type (*type* field) in the `meigwctrlS` register and clear the IP bit by writing to the gateway's `meigwclrS` register.
3. Set the base address of the external vectored interrupt address table by writing the *base* field of the `meivt` register.
4. Set the priority level for each external interrupt source *ext_src* by writing the corresponding *priority* field of the `meiplS` registers.
5. Set the priority threshold by writing *prithresh* field of the `meipt` register.
6. Initialize the nesting priority thresholds by writing '0' (or '15' for reversed priority order) to the *clidpri* field of the `meicidpl` and the *currpri* field of the `meicurpl` registers.
7. Enable interrupts for the appropriate external interrupt sources by setting the *inten* bit of the `meieS` registers for each interrupt source *ext_src*.

RISC-V SweRV Register map

block	register	field	bits	sw access	hw access	field default	description
SweRV							
	mip						Machine Interrupt Pending Register
		mceip	[30:30]	ro	ro	0	Correctable error local interrupt pending
		meip	[11:11]	ro	ro	0	Machine external interrupt pending
		mtip	[7:7]	ro	ro	0	Machine timer interrupt pending
	mpiccfg						
		priord		rw	rw	0	Priority order: 0: RISC-V standard compliant priority order (0=lowest to 15=highest) 1: Reverse priority order (15=lowest to 0=highest)
	meigwctrl						{repeat=2}
		polarity	[0:0]	rw	rw	0	External interrupt polarity for interrupt source ID S: 0: Active-high interrupt 1: Active-low interrupt
		type1	[1:1]	rw	rw	0	External interrupt type for interrupt source ID S: 0: Level-triggered interrupt 1: Edge-triggered interrupt
	meivt						
		base	[31:10]	rw	rw	0	Base address of external interrupt vector table
	meipl						{repeat=2}
		priority1	[3:0]	rw	rw	0	External interrupt priority level for interrupt source ID
	meipt						
		prithresh	[3:0]	rw	rw	0	External interrupt priority threshold
	meicidpl						
		clidpri	[3:0]	rw	rw	0	Priority level of preempting external interrupt source (corresponding to source ID read from claimid field of meihap register)

AGNISYS
SYSTEM DEVELOPMENT WITH CERTAINTY

Interpretation in ISequenceSpec

sequence name	ip	description
initial_seq	SweRV	
arguments	**value**	**description**
set_val	1	Set value
clear_val	0	Clear value
size	2	Size of external sources
constants	**value**	**description**
base_addr	0x0	Base address
edge_detect	1	Edge detection
variables	**value**	**description**
ext_src[2]	{0,0}	External source
exintsrc_req	0	
intr_req	0	Interrupt request signal of interrupt source
intr_req_gate	0	Interrupt request signal of gateway
reset_sig	1	To reset entire system; if set to 1, initialization will begin
pic_clk	0	
read_var	0	
internal_intr_bit	0	Internal interrupt pending bit
comp_in1[2]	{0,0}	For first level comparator
comp_in2[2]	{0,0}	For second level comparator
comp_in3[2]	{1,1}	For second level comparator
comp_out	0	Output of Comparator
mexintirq	0	
WUN	0	Wake up Notifiacation bit

Arguments of sequences

Declaration of Constants

Declaration of Variables

AGNISYS
SYSTEM DEVELOPMENT WITH CERTAINTY

Interpretation in ISequenceSpec - Continued

command	step	value	description
if(reset_sig==1){			**Initialiazation of control registers**
write	mpiccfg.priord	1	Configured the priority order
for (exintsrc_req=0; exintsrc_req<size;exintsrc_req++){			
write	meigwctrl[exintsrc_req].polarity	set_val	"polarity" field set of "meigwctrl" register
write	meigwctrl[exintsrc_req].type1	set_val	"type" field set of "meigwctrl" register
write	meigwclr[exintsrc_req]	clear_val	Cleared the IP bit by writing to the gateway's "meigwclr" register
	ext_src[exintsrc_req]	1	
if(ext_src[exintsrc_req]==1){			
write	meipl[exintsrc_req].priority1	set_val	Priority level for each external interrupt source is set
write	meie[exintsrc_req].inten	set_val	Enabled interrupts for the appropriate external interrupt sources
}			
}			
write	meivt.base	base_addr	Base address of external vectored interrupt address table is set
write	meipt.prithresh	1	Priority threshold is set
write	meicidpl.clidpri	0	Initialized the nesting priority thresholds
write	meicurpl.currpri	0	
	reset_sig	0	

AGNISYS
SYSTEM DEVELOPMENT WITH CERTAINTY

Initialization Sequences Using Python

```
class sequences:
        def swerv_init(self, ip = 'swerv.rdl'):
            set_val = iss.argument(1,'set_val','set value')
            clear_cal = iss.argument(2,'clear_val','clear value')
            size = iss.argument(1,'size','size')

            base_addr = iss.constant(0,'base_addr','Base address')
            edge_detect = iss.constant(1,'edge_detect','edge detection')

            ext_src  = iss.variable([0,0], 'ext_src', 'external source')
            exintsrc_req  = iss.variable(0, 'exintsrc_req', 'description')
            int_req  = iss.variable(0, 'int_req', 'description')
            int_req_gate  = iss.variable(0, 'int_req_gate', 'description')
            reset_sig  = iss.variable(0, 'reset_sig', 'description')
            pic_clk  = iss.variable(0, 'pic_clk', 'description')
            read_var  = iss.variable(0, 'read_var', 'description')
            internal_intr_bit  = iss.variable(0, 'internal_intr_bit', 'description')
            comp_in1  = iss.variable([0,0], 'comp_in1', 'description')
            comp_in2  = iss.variable([0,0], 'comp_in2', 'description')
            comp_in3  = iss.variable([1,1], 'comp_in3', 'description')
            comp_out  = iss.variable(0, 'comp_out', 'description')
            maxintirq  = iss.variable(0, 'maxintirq', 'description')
            WUN  = iss.variable(0, 'WUN', 'description')

            if(reset_sig == 1):
                iss.write(mpiccfg.priord, 1,'configured the priority order')
                while(exintsrc_req < size):
                    iss.write(meigwctrl.polarity, set_val,'polarity filed set of meigwctrl reg')
                    iss.write(meigwctrl.type1, set_val,'type field set of meigwctrl reg')
                    iss.write(meigwclr, clear_val,'cleared the IP bit by writing to the gateways reg')
                    iss.write(ext_src,1,'')
                    if(ext_src == 1):
                        iss.write(meip.priority1, set_val,'priority level for each external intr source is set')
                        iss.write(meie.inten, set_val,'enabled intr for the appropriate external intr source')

            iss.write(meivt.base, base_addr,'Base address of external vectored intr address table is set')
            iss.write(meipt.prithesh, 1,'priority threshold is set')
            iss.write(meicidpl.clidpri, 0,'init the nesting priority thresholds')
            iss.write(meicurpl.currpri, 0,'')
```

Arguments

Constants

Variables

Initialization sequences

AGNISYS
SYSTEM DEVELOPMENT WITH CERTA

Generated Sequences in the Target Format : UVM

```
class uvm_initial_seq_seq extends uvm_reg_sequence#(uvm_sequence#(uvm_reg_item));
    `uvm_object_utils(uvm_initial_seq_seq)
    uvm_status_e status;
    SweRV_block rm ;
    function new(string name = "uvm_initial_seq_seq") ;
        super.new(name);
        this.init();
    endfunction
    int set_value=1;
    int clear_value=0;
    int a_size=2;
    function init(int set_value=1,int clear_value=0,int a_size=2);
        this.set_value = set_value;
        this.clear_value = clear_value;
        this.a_size = a_size;
    endfunction
    const int base_addri = 'h0 ;
    int ext_srci[2] = {0,0} ;
    int reset_sigi = 1 ;
    int exintsrc_reqi = 0 ;
    task body;
        if(!$cast(rm, model)) begin
            `uvm_error("RegModel : SweRV_block","cannot cast an object of type uvm_reg_sequence to
            rm");
        end
        if (rm == null)  begin
            `uvm_error("SweRV_block", "No register model specified to run sequence on, you should
            specify regmodel by using property 'uvm.regmodel' in the sequence")
            return;
        end
```

UVM generated sequences

```
if (reset_sigi == 1) begin
    //-------------------------------------------------------------------------------
    /*---- Configured the priority order*/
    //-------------------------------------------------------------------------------
    rm.mpiccfg.priord.write(status, 'h1, .parent(this));
    for ( int exintsrc_reqi = 0 ; exintsrc_reqi < a_size;exintsrc_reqi++ )
    begin
        //-------------------------------------------------------------------------------
        /*---- polarity field set of meigwctrl register*/
        //-------------------------------------------------------------------------------
        rm.meigwctrl[exintsrc_reqi].polarity.set(set_value);
        //-------------------------------------------------------------------------------
        /*---- type field set of meigwctrl register*/
        //-------------------------------------------------------------------------------
        rm.meigwctrl[exintsrc_reqi].typel.set(set_value);
        rm.meigwctrl[exintsrc_reqi].update(status);
        //-------------------------------------------------------------------------------
        /*---- Cleared the IP bit by writing to the gateway's meigwclr register*/
        //-------------------------------------------------------------------------------
        rm.meigwclr[exintsrc_reqi].write(status, clear_value, .parent(this));
        ext_srci[exintsrc_reqi]=1;
        if (ext_srci[exintsrc_reqi] == 1) begin
            //-------------------------------------------------------------------------------
            /*---- Priority level for each external interrupt source is set*/
            //-------------------------------------------------------------------------------
            rm.meipl[exintsrc_reqi].priorityl.write(status, set_value, .parent(this));
            //-------------------------------------------------------------------------------
            /*---- Enabled interrupts for the appropriate external interrupt sources*/
            //-------------------------------------------------------------------------------
            rm.meie[exintsrc_reqi].inten.write(status, set_value, .parent(this));
        end
    end
```

AGNISYS
SYSTEM DEVELOPMENT WITH CERTAINTY

Generated Sequences in the Target Format : Firmware

```
int initial_seq( int set_value ,int clear_value ,int a_size ) {
unsigned int consolidated_temp_value = 0;
static const int base_addri = 0x0 ;      /*---------- base address ----------*/
int flag;
int ext_srci[2] = {0,0} ;      /*---------- external source, ext_src ----------*/
int reset_sigi = 1 ;
int exintsrc_reqi = 0 ;
int dim_wr;
if( reset_sigi == 1){
    //-------------------------------------------------------------------------
    /*---- Configured the priority order*/
    //-------------------------------------------------------------------------
    consolidated_temp_value = ((0x00000001) & SWERV_MPICCFG_PRIORD_MASK) | (~(
    SWERV_MPICCFG_PRIORD_MASK) & consolidated_temp_value);
    REG_WRITE(SweRV_mpiccfg_ADDRESS,consolidated_temp_value);
    for ( exintsrc_reqi = 0 ;  exintsrc_reqi < a_size;exintsrc_reqi++ ) {
        //-------------------------------------------------------------------------
        /*---- polarity field set of meigwctrl register*/
        //-------------------------------------------------------------------------
        dim_wr = SweRV_meigwctrl_ADDRESS+SweRV_meigwctrl_SIZE*exintsrc_reqi;
        consolidated_temp_value = ((set_value << SWERV_MEIGWCTRL_POLARITY_OFFSET) &
        SWERV_MEIGWCTRL_POLARITY_MASK) | (~(SWERV_MEIGWCTRL_POLARITY_MASK) & consolidated_temp_value
        //-------------------------------------------------------------------------
        /*---- type field set of meigwctrl register*/
        //-------------------------------------------------------------------------
        dim_wr = SweRV_meigwctrl_ADDRESS+SweRV_meigwctrl_SIZE*exintsrc_reqi;
        consolidated_temp_value = ((set_value << SWERV_MEIGWCTRL_TYPE1_OFFSET) &
        SWERV_MEIGWCTRL_TYPE1_MASK) | (~(SWERV_MEIGWCTRL_TYPE1_MASK) & consolidated_temp_value);
        REG_WRITE(dim_wr,consolidated_temp_value);
        //-------------------------------------------------------------------------
        /*---- Cleared the IP bit by writing to the gateway's meigwclr register*/
        //-------------------------------------------------------------------------
        .   .   .
    }
```

Firmware generated sequences

```
    //-------------------------------------------------------------------------
    /*---- Base address of external vectored interrupt address table is set*/
    //-------------------------------------------------------------------------
    consolidated_temp_value = ((0x00000000) & SWERV_MEIVT_BASE_MASK) | (~(SWERV_MEIVT_BASE_MASK) &
    consolidated_temp_value);
    REG_WRITE(SweRV_meivt_ADDRESS,consolidated_temp_value);
    //-------------------------------------------------------------------------
    /*---- Priority threshold is set*/
    //-------------------------------------------------------------------------
    consolidated_temp_value = ((0x00000001) & SWERV_MEIPT_PRITHRESH_MASK) | (~(
    SWERV_MEIPT_PRITHRESH_MASK) & consolidated_temp_value);
    REG_WRITE(SweRV_meipt_ADDRESS,consolidated_temp_value);
    //-------------------------------------------------------------------------
    /*---- Initialized the nesting priority thresholds*/
    //-------------------------------------------------------------------------
    consolidated_temp_value = ((0x00000000) & SWERV_MEICIDPL_CLIDPRI_MASK) | (~(
    SWERV_MEICIDPL_CLIDPRI_MASK) & consolidated_temp_value);
    REG_WRITE(SweRV_meicidpl_ADDRESS,consolidated_temp_value);
    consolidated_temp_value = ((0x00000000) & SWERV_MEICURPL_CURRPRI_MASK) | (~(
    SWERV_MEICURPL_CURRPRI_MASK) & consolidated_temp_value);
    REG_WRITE(SweRV_meicurpl_ADDRESS,consolidated_temp_value);

}
return 0;

}
```

Benefits of ISequenceSpec

- Reduce Time and Cost of development
- Eliminate Duplication of sequence implementation
- Improve Quality
- Let ISS do the mundane work of ensuring correctness, while you focus on the algorithms

AGNISYS
SYSTEM DEVELOPMENT WITH CERTAINTY

Resistance to change

- Some may say …
  - Why do something new when everything is working fine
    - It saves time across teams, reduces development and debug time
  - We love our UVM or C/C++ language – don’t want to change
    - You can still write in your favorite language, just use the auto generated low level sequences
  - Reduces flexibility
    - But reduces workload as well

- We feel automating sequences is a low hanging fruit
  - Not much risk to it
  - And potential for high Return on Investment

Conclusion

- Today, sequences are at a point where registers were a decade ago
- There are a lot of challenges in dealing with sequences
- Also a huge opportunity for automation
- Get it right to get huge productivity gains, get it wrong, and lose a lot of cycles debugging
- Agnisys is leading the drive to automate sequences

About Agnisys

- The EDA leader in solving complex design and verification problems associated with HW/SW interface
- Formed in 2007
- Privately held and Profitable
- Headquarters in Boston, MA
  - ~1000 users worldwide
  - ~50 customer companies
- Customer retention rate ~90%
- R&D centers (US and India)
- Support centers - Committed to ensure comprehensive support
  - Email : support@agnisys.com
  - Phone : 1-855-VERIFYY
  - Response time within one day; within hours in many cases
  - Time Zones (Boston MA, San Jose CA and Noida India)

IDESIGNSPEC™ (IDS) EXECUTABLE REGISTER SPECIFICATION

Automatically generate UVM models, Verilog/VHDL models, Coverage Model, Software model etc.

AUTOMATIC REGISTER VERIFICATION (ARV)

ARV-Sim™ : Create UVM Test Environment, Sequences, Verification Plans and instantly know the status of the verification project.

ARV-Formal™ : Create Formal Properties and Assertions, and Coverage Model from the specification.

ISEQUENCESPEC™ (ISS)

Create UVM sequences and Firmware routines from the specification.

DVinsight™ (DVi)

Smart Editor for SystemVerilog and UVM projects.

IDS – Next Generation™ (IDS-NG)

Comprehensive SoC/IP Spec Creation and Code Generation Tool

AGNISYS
SYSTEM DEVELOPMENT WITH CERTAINTY