

2022
DESIGN AND VERIFICATION™
DVCON
CONFERENCE AND EXHIBITION
UNITED STATES

**IP/SoC Design, Co-Verify, Co-Validate,
Co-Everything in 90 minutes!**

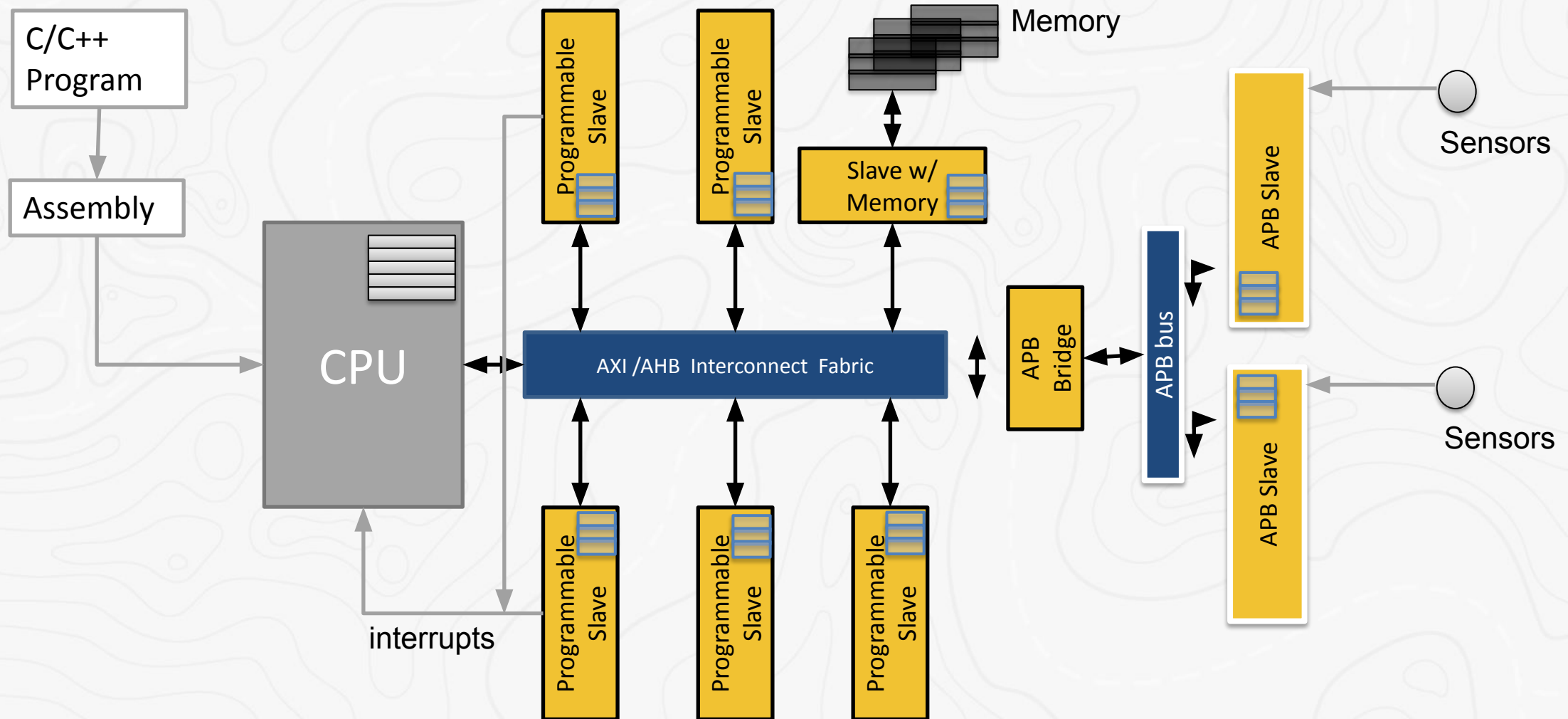
Nikita Gulliya, Agnisys
Neena Chandawale, Agnisys
Anupam Bakshi, Agnisys



Typical Chip Design

- Hardware of the SoC is designed by HW team
 - But used by
 - Verification/Emulation team
 - Firmware team
 - Validation team
 - Software team
- How does the software interact with the IPs?
 - Through the Hardware Software Interface (HSI)
- Hardware is at the core and software API is around it
- Device drivers (part of the HSI) are tedious to create
 - They are written in C and Assembly

Introduction to a Typical SoC



Challenges Faced

- Design challenges
 - Too much data
 - Changes to data cause havoc
 - Significant source of bugs
 - Reusing IP
- Verification/Validation challenges
 - Duplication of work across teams
 - Rise in complexity of designs
 - Inability to create same debug environment for multiple platforms
 - Mismatch in specification and implementation

Challenges Faced - Contd..

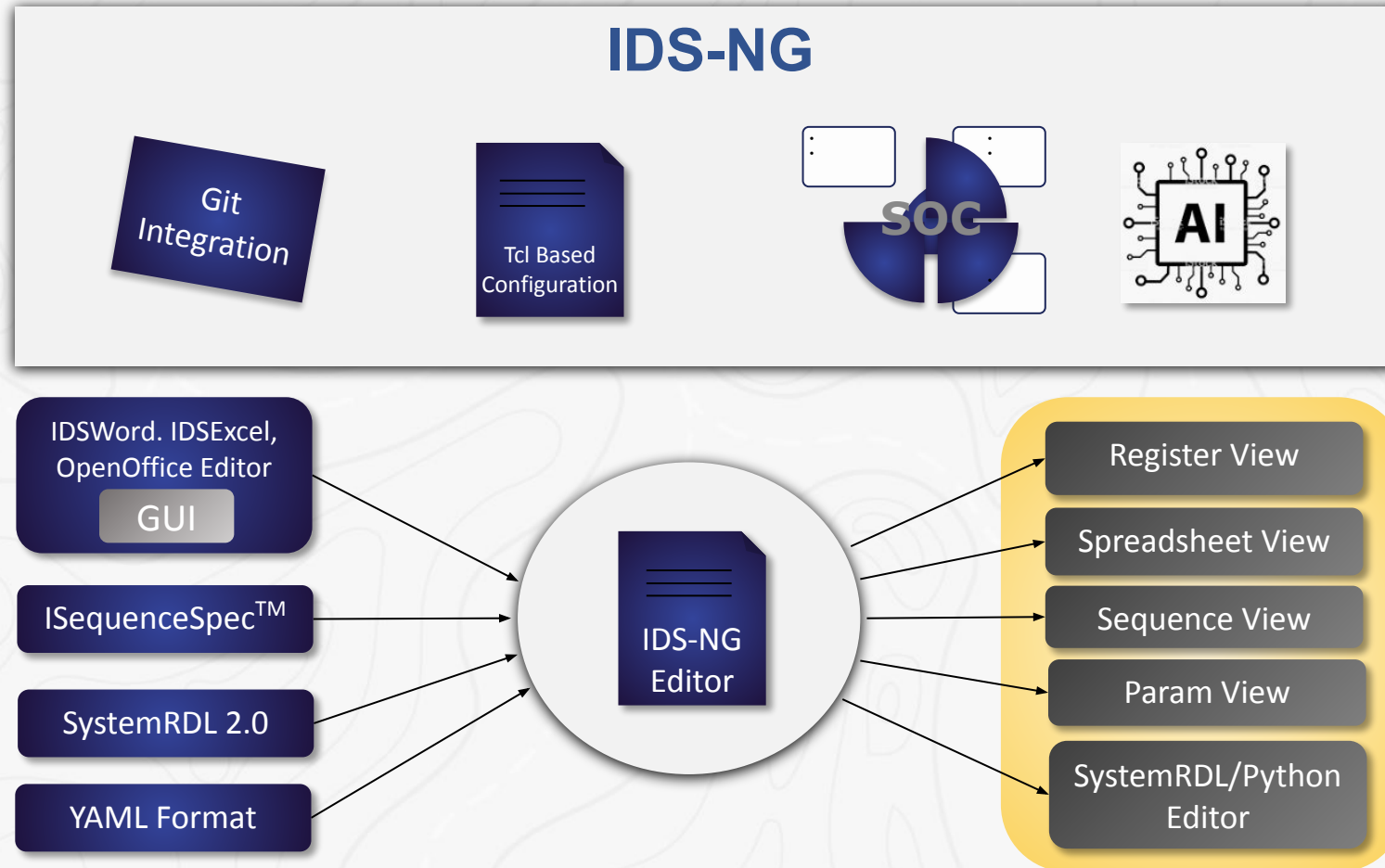
- SOC design companies
 - Increasing demands of design complexity and design performance
 - Combining automation with flexibility to accommodate changes in sub-systems across applications
 - Driving down the cost of design for a better ROI
 - Shrinking market windows
 - Boosting productivity of design teams to meet shorter market windows

An Ideal Solution

- Ease of generation
- Generated code should not be encrypted
- Should provide appropriate error messages
- Ability to reuse IPs
 - Customizing the designs
 - Configuring the designs
- Easy mechanism for generating IP blocks
- Ability to handle different bus protocols
- Handling metastability of multi clock domain designs
- Design must be functionally safe and secure

Creation of correct-by-construction, reusable designs, faster

Cross Platform IDE: IDS-NG



Capabilities of IDS-NG

- Git integration
- Creates design specifications
- Errors linking to specs are highlighted
- Hints are provided while writing sequences, checkers, etc.
- Navigators such as hierarchy view and sequence view show all automated sequences that can be used
- Single GUI for capturing all information related to the IP/SoC
- Property panel to give values to possible properties which can be applied to that component

Features Available for Designing

- Bus protocols
 - The RTL generated can support different bus protocols for high performance data transfer among the IPs
 - The protocols differentiate features such as pipelining, burst, and split transfers
 - The desired SoC bus can be selected using the Configure button

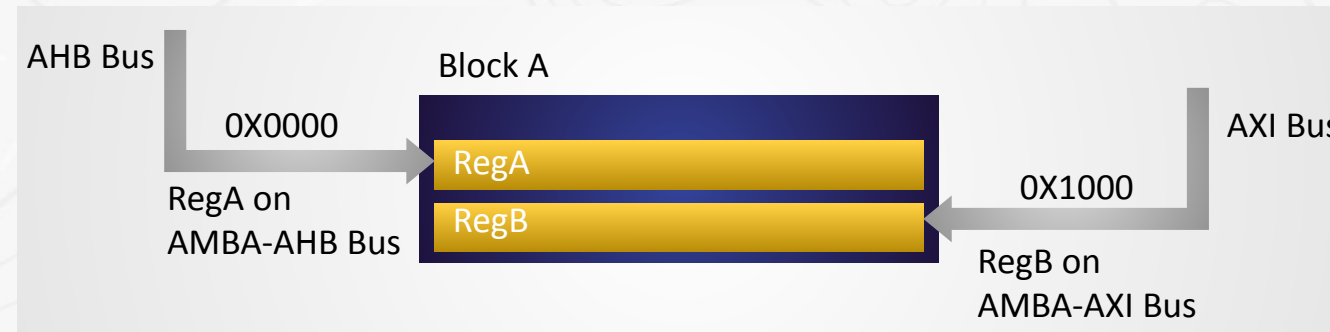
Bus	RTL
AMBA-AHB	✓
AMBA3-AHB-Lite	✓
AMBA-APB	✓
AVALON	✓
AMBA-AXI	✓
AMBA-AXI4FULL	✓
SPI	✓
I2C	✓
TILELINK	✓
WISHBONE	✓
OCP	✓

Bus

<input checked="" type="radio"/> AMBA-AHB	<input type="radio"/> AVALON	<input type="radio"/> PROPRIETARY	<input type="radio"/> AMBA-APB
<input type="radio"/> AMBA3-AHB-lite	<input type="radio"/> AMBA-AXI	<input type="radio"/> AMBA-AXI4FULL	<input type="radio"/> OCP
<input type="radio"/> WISHBONE	<input type="radio"/> SPI-beta	<input type="radio"/> I2C-beta	<input type="radio"/> TILELINK

Features Available for Designing - Contd..

- Multiple bus domains
 - User can select bus domains from the configuration settings in which the block or chip resides
 - The chip/block/register can be configured to reside in multiple bus domains

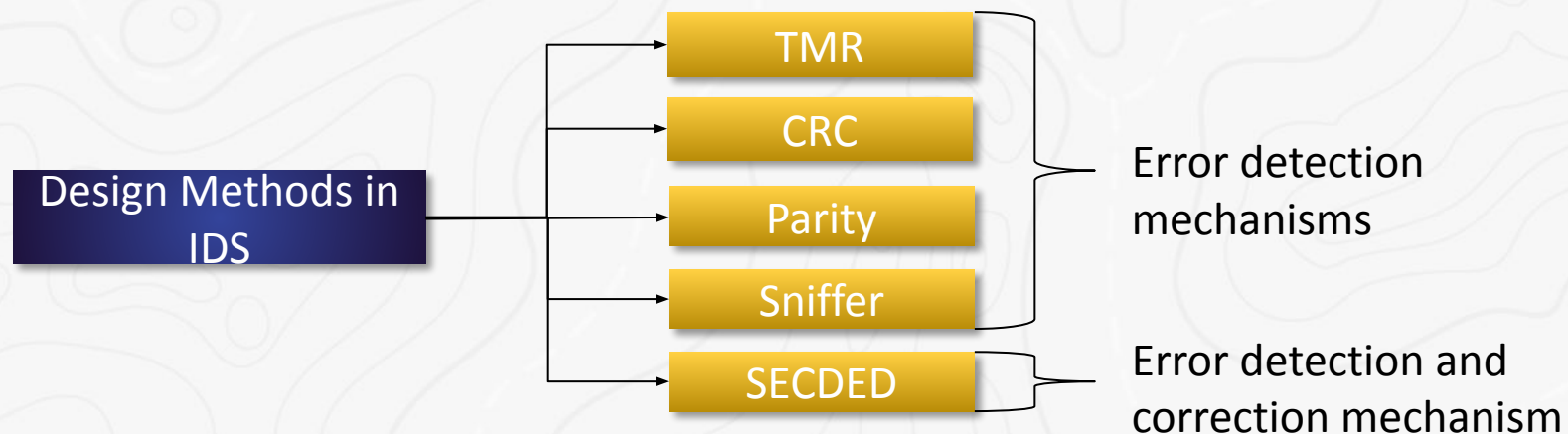


- Various domains (generally 2 or more) are described in a bus domain template (see below) typically at the top of the document

bus domain name	address unit	description	bus
AMBA	8	This is AMBA-AHB bus domain	AMBA-AHB
AXI	32	This is AMBA-AXI bus domain	AMBA-AXI

Features Available for Designing - Contd..

- Functional safety
 - System is functioning correctly or not
 - Human errors, environmental stress, cosmic rays, and hardware failures should be handled properly if they occur in a design
 - Designs should be addressed with mechanisms to detect and correct the failures



Features Available for Designing - Contd..

- Functional security
 - Security is one of the most important aspects of SoCs/IPs in the Internet-of-Things (IoT) era
 - Securing the SoCs/IPs is critical for providing authentication, confidentiality, integrity, non-reproduction, and access control to the system
 - Chip designs should be secured and protected, and should not be affected by any misconfiguration at the designer's end

Design methods in IDS

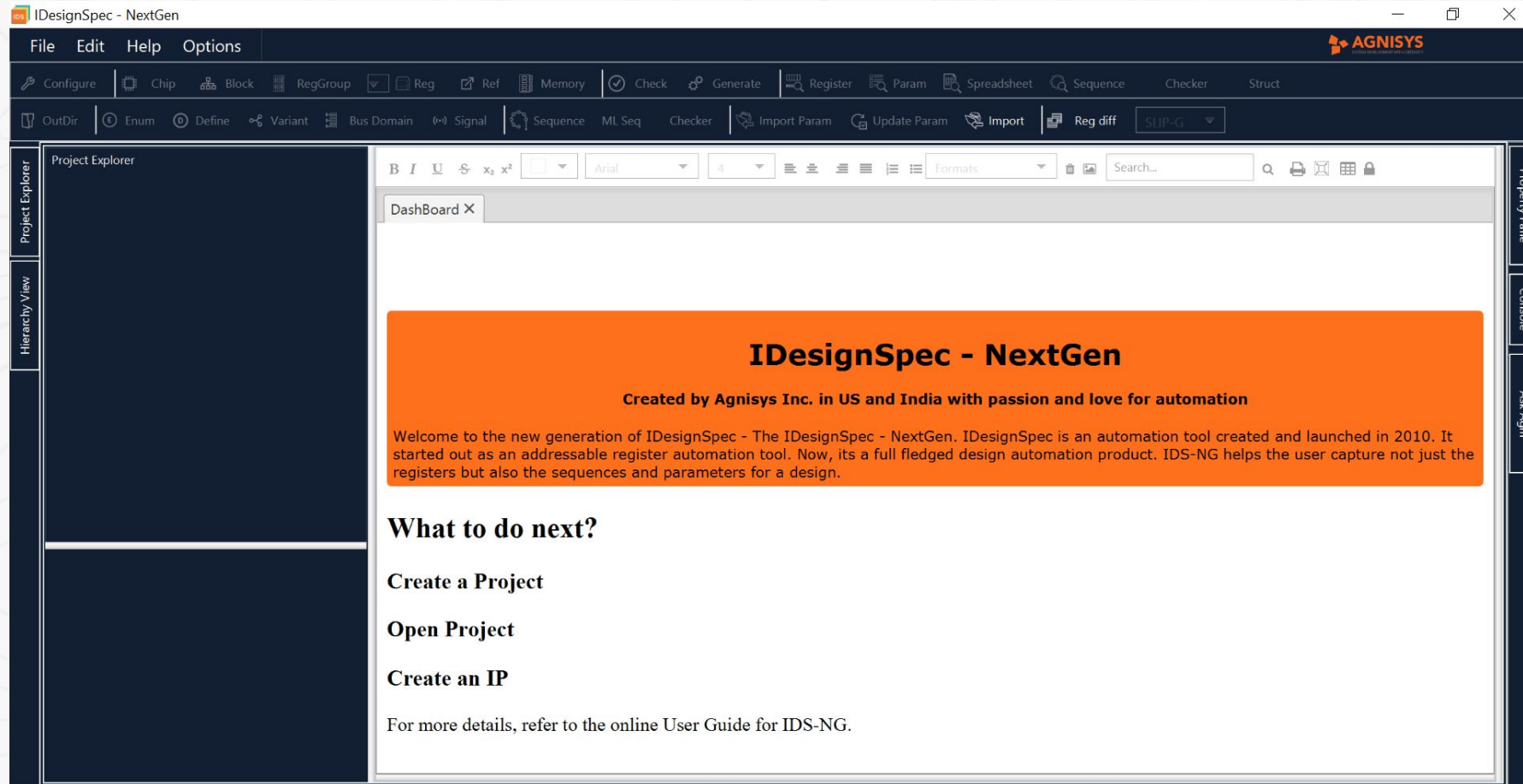
Lock
Register

Protection in
sw interfaces

Advanced
Encryption
Standard

Designing IPs in IDS-NG

- IDS-NG GUI



Designing IPs in IDS-NG - Contd..

- Register specification

Register view:

IDS Register Specification

reg_name		32	address default		
Properties..					
Description..					
bits	name	s/w	h/w	default	description
31:0	pkt32	rw	ro		a 32 bit packet field.

Spreadsheet view:

IDS Register Specification Open IDS Template

	A	B	C	D	E	F	G	H	I	J
1	chip	block	register	width	field	sw access	hw access	field default	bits	description
2			reg_name	32						
3					pkt32	rw	ro		31:0	a 32 bit packet field.
4										

Define user registers' settings

C Header,
Misra-C,
C Tests,

....
SystemRDL,
Register Sequences

...

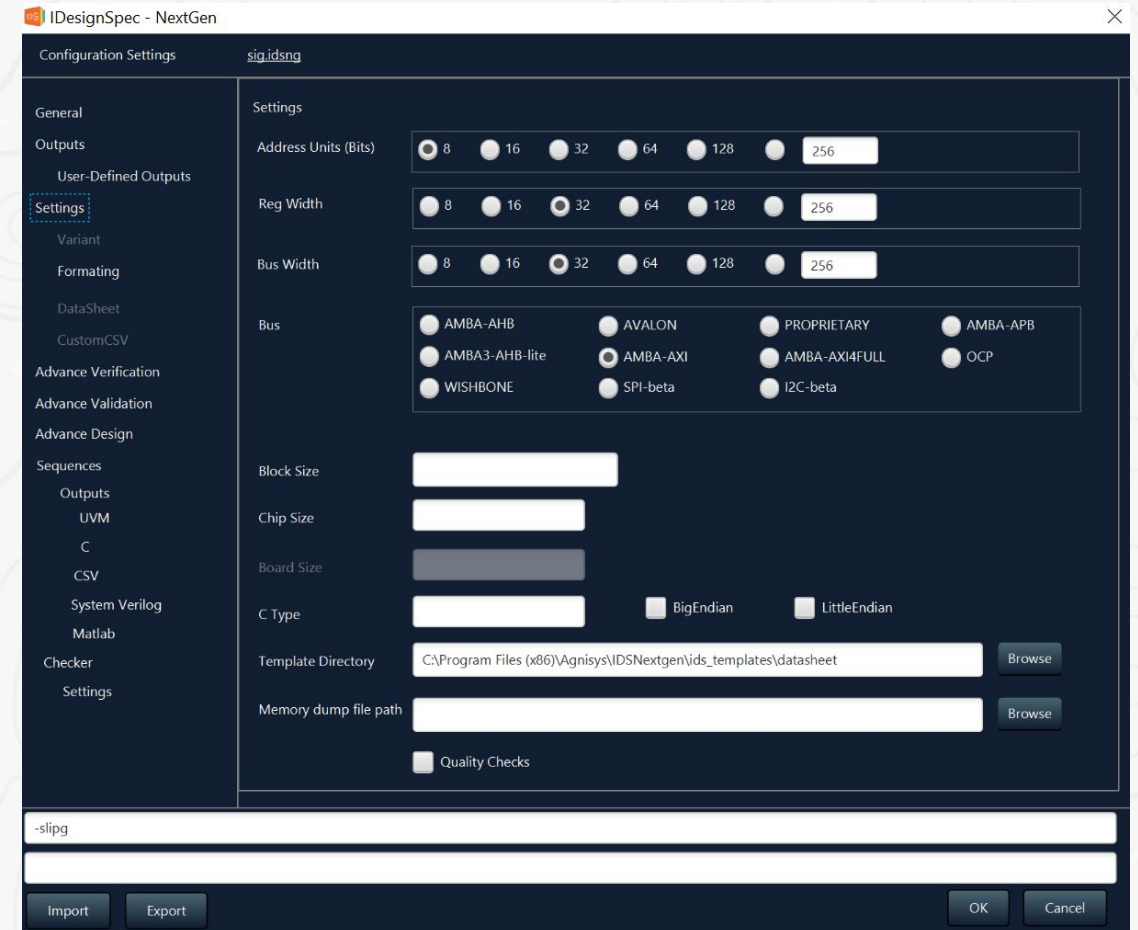
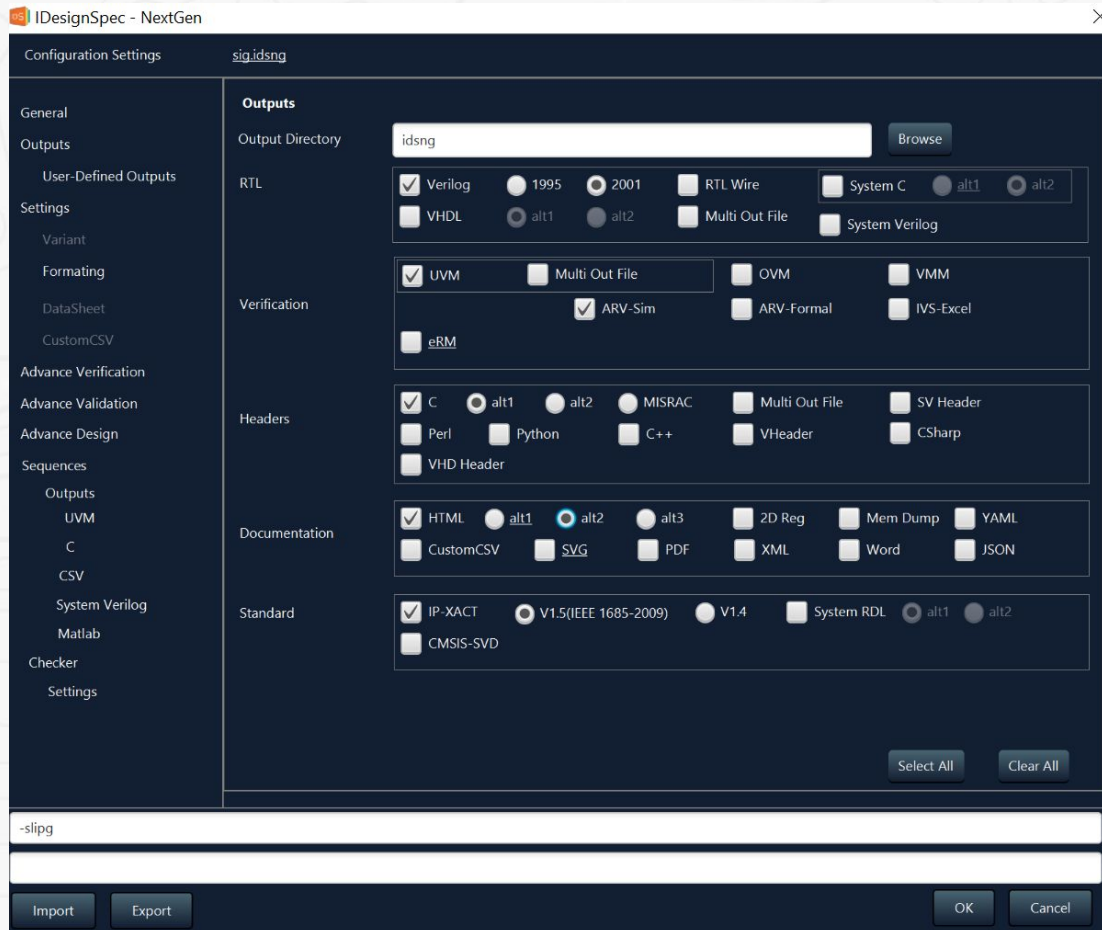
Designing IPs in IDS-NG - Contd..

- Sample specification

1	ethernet_ip				address 0x0
Properties..					
Description..					
1.1	tx_pkt			32	address 0x0 default 0x00000000
Description..					
bits	name	s/w	h/w	default	description
1:0	parity	rw	rw	0	
30:2	pkt	rw	rw	0	
31	start	rw	rw	0	
1.2	rx_pkt			32	address 0x4 default 0x00000000
Properties..					
Description..					
bits	name	s/w	h/w	default	description
1:0	parity	rw	rw	0	
30:2	pkt	rw	rw	0	
31	start	rw	rw	0	

Designing IPs in IDS-NG - Contd..

- Register configuration



Designing IPs in IDS-NG - Contd..

- Generated sample codes

```
module sig_ids#(
    parameter bus_width = 32,
    parameter addr_width = 2,
    parameter block_size = 'h4,
    parameter [addr_width-1 : 0] block_offset = {(addr_width){1'b0}}
)
(
    output reg_name_enb,
    input [32-1 : 0] reg_name_fld_in,
    input reg_name_fld_in_enb,
    output [31 : 0] reg_name_fld_r,
    input aclk,
    input aresetn,
    input [addr_width-1 : 0] awaddr,
    input awvalid,
    output awready,
    input [2 : 0] awprot,
    input [bus_width-1 : 0] wdata,
    input wvalid,
    output wready,
    input [bus_width/8 - 1 : 0] wstrb,
    output [1 : 0] bresp,
    input bready,
    output bvalid,
    input [addr_width-1 : 0] araddr,
    input arvalid,
    output arready,
    input [2 : 0] arprot,
    output [bus_width-1 : 0] rdata,
    output rvalid,
    input rready,
    output [1 : 0] rresp
);
.
.
axi_widget # (.addr_width(addr_width), .bus_width(bus_width) )axi (
.
.
);
.
.
assign wr_slave_select = ((slvwaddr[addr_width - 1 : 0] >= block_offset) && (slvwaddr[addr_width - 1 : 0]
<= block_offset + block_size -1)) ? 1'b1 : 1'b0;
assign rd_slave_select = ((slvraddr[addr_width - 1 : 0] >= block_offset) && (slvraddr[addr_width - 1 : 0]
<= block_offset + block_size -1)) ? 1'b1 : 1'b0;
endmodule
```

RTL

```
/*-----*/
Class      : ethernet_ip_block
DESCRIPTION:-
/*-----*/
`ifndef CLASS_ethernet_ip_block
`define CLASS_ethernet_ip_block
class ethernet_ip_block extends uvm_reg_block;
    `uvm_object_utils(ethernet_ip_block)

    rand ethernet_ip_tx_pkt tx_pkt;
    rand ethernet_ip_rx_pkt rx_pkt;

    // Function : new
    function new(string name = "ethernet_ip_block");
        super.new(name, UVM_NO_COVERAGE);
    endfunction

    // Function : build
    virtual function void build();
        //define default map and add reg/regfiles
        default_map= create_map("default_map", 'h0, 4, UVM_BIG_ENDIAN, 1);

        //TX_PKT
        tx_pkt = ethernet_ip_tx_pkt::type_id::create("tx_pkt");
        tx_pkt.configure(this, null, "tx_pkt");
        tx_pkt.build();
        default_map.add_reg( tx_pkt, 'h0, "RW");

        //RX_PKT
        rx_pkt = ethernet_ip_rx_pkt::type_id::create("rx_pkt");
        rx_pkt.configure(this, null, "rx_pkt");
        rx_pkt.build();
        default_map.add_reg( rx_pkt, 'h4, "RW");

        lock_model();
    endfunction
endclass
`endif
```

UVM

Designing IPs in IDS-NG - Contd..

- Generated sample codes

```

#ifndef _ETHERNET_IP_REGS_H_
#define _ETHERNET_IP_REGS_H_
typedef union {
    struct {
        hwint start : 1;          /* 31 SW=rw HW=rw 0x0 */
        hwint pkt : 29;          /* 30:2 SW=rw HW=rw 0x0 */
        hwint parity : 2;        /* 1:0 SW=rw HW=rw 0x0 */
    } bf;
    hwint dw;
} ethernet_ip_tx_pkt;
typedef union {
    struct {
        hwint start : 1;          /* 31 SW=rw HW=rw 0x0 */
        hwint pkt : 29;          /* 30:2 SW=rw HW=rw 0x0 */
        hwint parity : 2;        /* 1:0 SW=rw HW=rw 0x0 */
    } bf;
    hwint dw;
} ethernet_ip_rx_pkt;
typedef struct {
    ethernet_ip_tx_pkt tx_pkt;
    ethernet_ip_rx_pkt rx_pkt;
} ethernet_ip_s;
#define ethernet_ip_s_SIZE 0x8
#define ethernet_ip_tx_pkt_SIZE 0x4
#define ethernet_ip_rx_pkt_SIZE 0x4
#define ethernet_ip_s_OFFSET 0x0
#define ethernet_ip_tx_pkt_OFFSET 0x0
#define ethernet_ip_rx_pkt_OFFSET 0x4
#define ethernet_ip_s_ADDRESS 0x0
#define ethernet_ip_tx_pkt_ADDRESS 0x0
#define ethernet_ip_rx_pkt_ADDRESS 0x4
#define ETHERNET_IP_TX_PKT_START_OFFSET 31
#define ETHERNET_IP_TX_PKT_START_MASK 0x80000000
#define ETHERNET_IP_TX_PKT_START_INV_MASK 0x7FFFFFFF
#define ETHERNET_IP_TX_PKT_START_VALUE_MASK 0x40000000
#define ETHERNET_IP_TX_PKT_START_INV_VALUE_MASK 0xBFFFFFFF
#define ETHERNET_IP_TX_PKT_START_SIZE 1
#define ETHERNET_IP_TX_PKT_START_DEFAULT 0
.
.
#endif /* _ETHERNET_IP_REGS_H_ */
/* end */

```

CHeader

Block : ethernet_ip

Table of Content			
S.No.	Names	Default	Address
1.1	reg : tx_pkt	0x00000000	0x0
1.2	reg : rx_pkt	0x00000000	0x4

1 : Block : ethernet_ip

0x0

Description:

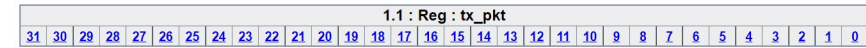
Reg :

tx_pkt
rx_pkt

1.1 : Reg : tx_pkt

0x0

Description:



[31] start

[1:0] parity

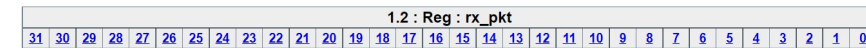
[30:2] pkt

Bits	Field name	sw	hw	default	Description
1:0	parity	rw	rw	0x0	
30:2	pkt	rw	rw	0x0	
31	start	rw	rw	0x0	

1.2 : Reg : rx_pkt

0x4

Description:



[31] start

[1:0] parity

[30:2] pkt

Bits	Field name	sw	hw	default	Description
1:0	parity	rw	rw	0x0	
30:2	pkt	rw	rw	0x0	
31	start	rw	rw	0x0	

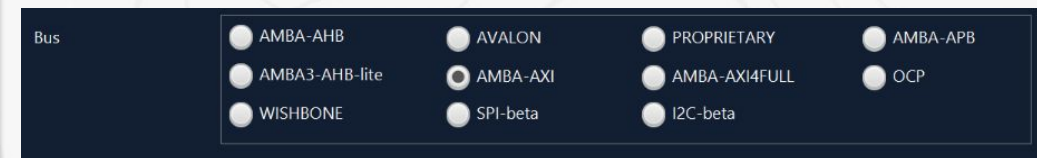
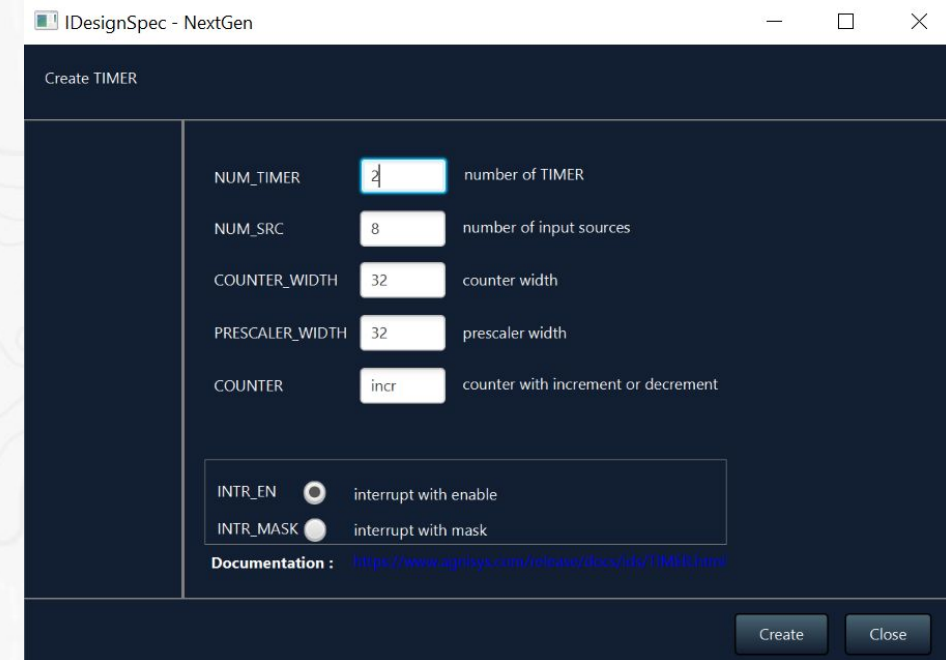
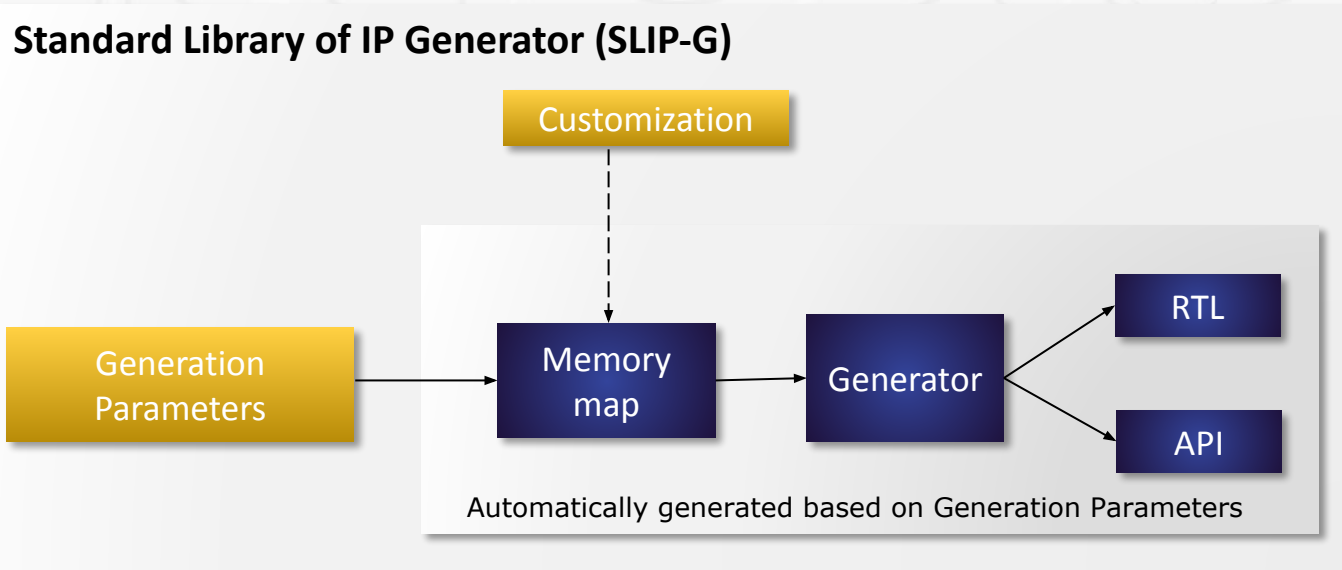
(top of block)

Agnisys, Inc. *****

HTML

Auto Generating Standard IPs

- IDS-NG can also be used to automatically generate standard IPs (fully verified and validated) and their APIs, also provides add-in functionality of configurability and customizability



Auto Generating Standard IPs - Contd..

- Register specification - Automatically generated by setting generation parameters

define name	value	description	private
SNUM_TIMER	2		1
SNUM_SRC	8		1
SCOUNTER_WIDTH	32		1
SPRESCALER_WIDTH	32		1

name	port type	description
status_reset[1:0]	input	
result_reset[1:0]	input	
counter_reset[1:0]	input	
counter_start[1:0]	input	

bits	name	s/w	h/w	default	description
0	overflow	r/w1c	wo	0	Interrupt status for overflow {intr.status=overflow;}
1	run_intr	r/w1c	wo	0	Interrupt status for running mode {intr.status=running;}
2	period_intr	r/w1c	wo	0	Interrupt status for periodic mode {intr.status=periodic;}
3	cfg_ch	r/w1c	wo	0	Interrupt status for configuration change {intr.status=cfg;}

bits	name	s/w	h/w	default	description
0	en	rw	ro	0	1:- Enables the timer block or calculations 0:- Disables the timer block
2:1	mode	rw	ro	0	00: running mode 01: Periodic mode with source enable 00; periodic mode without source 11: reserved
5:3	event_sel	rw	ro	0	Legal values for running mode 000: high level 001: low level 010: between two high edges 011: between two low edges Legal values for the Periodic Mode 100: Posedges 101: negedges 110: both edges
8:6	Src_sel	rw	ro	0	To select the source on which measurement will be performed. These act as a counter enable signals for a given timer. If no source width is defined, then the timer will count the clock edges depending upon the prescaling value.

bits	name	s/w	h/w	default	description
31:0	F1	rw	ro	0	Provides the threshold value

bits	name	s/w	h/w	default	description
0	Over_intr_en	rw	na	0	Interrupt enable for overflow {intr.enable=overflow;}
1	Run_intr_en	rw	na	0	Interrupt enable for running mode {intr.enable=running;}
2	Period_intr_en	rw	na	0	Interrupt enable for periodic mode {intr.enable=periodic;}
3	cfg	rw	na	0	Interrupt enable for configuration change {intr.enable=cfg;}

bits	name	s/w	h/w	default	description
31:0	data	ro	wo	0	Stores the counting information depending upon the selected modes.

bits	name	s/w	h/w	default	description
31:0	F1	rw	ro	0	Defines the prescaling values

bits	name	s/w	h/w	default	description
31:0	F1	rw	ro	0	{counter=incr;resetsignal=counter_reset%d:0:sync:high;h.counter_start:1:sync:high;}

Auto Generating Standard IPs - Contd..

- Generated sample codes

```
module timer_top #(
    parameter bus_width = 32,
    parameter addr_width = 6,
    parameter timer_offset = 'h0,
    parameter timer_count = 2
)
(
    input clk,
    input reset,
    input [7:0]src,
    output [timer_count-1:0]irq_tmr,
    .
);
    reg [timer_count-1:0] cfg_f;
    wire [timer_count-1:0] irq_wire;
    timer_ids #(.bus_width(bus_width), .addr_width(addr_width), .block_offset(timer_offset))
    regmap(
        .status_reset(control_en_r),
        .result_reset(control_en_r)
    );
    generate
        genvar tmr_cnt;
        for(tmr_cnt=0; tmr_cnt < timer_count; tmr_cnt = tmr_cnt + 1) begin : timer_cnt
            timer_core #(.bus_width(bus_width), .addr_width(addr_width)) core(
                .clk(clk),
                .src(src),
                .reset(reset),
                .timer_enb(control_en_r[tmr_cnt]),
                .pre_clk(clk_en[tmr_cnt])
            );
            assign irq_tmr[tmr_cnt] = irq_wire[tmr_cnt];
        end
    endgenerate
endmodule
```

RTL

```
#ifndef IDS_TIMER
#define IDS_TIMER
#include "sig.h"
#endif

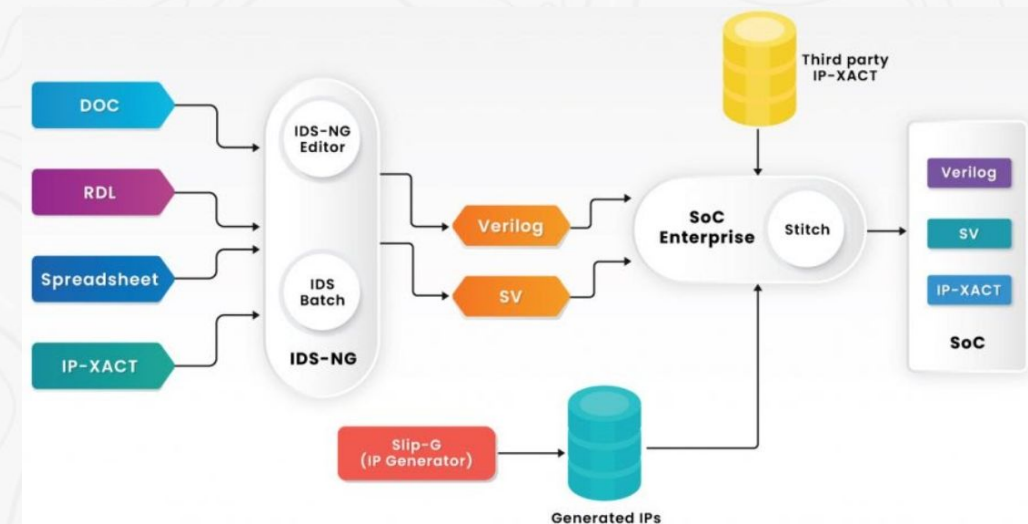
#include "timer_reset_mode_iss.h"

int timer_reset_mode( int timer_sel) {
    unsigned int consolidated_temp_value = 0;
    static const int reset_val = 0 ;
    int dim_wr;
    dim_wr = (timer_control_OFFSET + (timer_control_PER_INSTANCE_SIZE * (
    timer_sel))) + (timer_s_OFFSET);
    REG_WRITE(dim_wr,reset_val);
    dim_wr = 0;
    dim_wr = (timer_period_OFFSET + (timer_period_PER_INSTANCE_SIZE * (timer_sel
    ))) + (timer_s_OFFSET);
    REG_WRITE(dim_wr,reset_val);
    dim_wr = 0;
    dim_wr = (timer_prescaler_OFFSET + (timer_prescaler_PER_INSTANCE_SIZE * (
    timer_sel))) + (timer_s_OFFSET);
    REG_WRITE(dim_wr,reset_val);
    dim_wr = 0;
    dim_wr = (timer_counter_OFFSET + (timer_counter_PER_INSTANCE_SIZE * (
    timer_sel))) + (timer_s_OFFSET);
    REG_WRITE(dim_wr,reset_val);
    dim_wr = 0;
    return 0;
}
```

C sequence

Smart IP Integration and SoC Assembly

- A flexible and customizable environment for design assembly
- Create, package, integrate, and reuse IPs and SoC/FPGA
- Generic, standards compliant (IP-XACT - now IEEE 1685-2014)
- Automatically generates integration logic components and subsystems
- Automatically creates appropriate files for design, verification, and software teams
- Creates schematics for design analysis
- Runs design rule checks to ensure IP and SoC quality



Assembly in GUI Mode

Integration using console:

soce

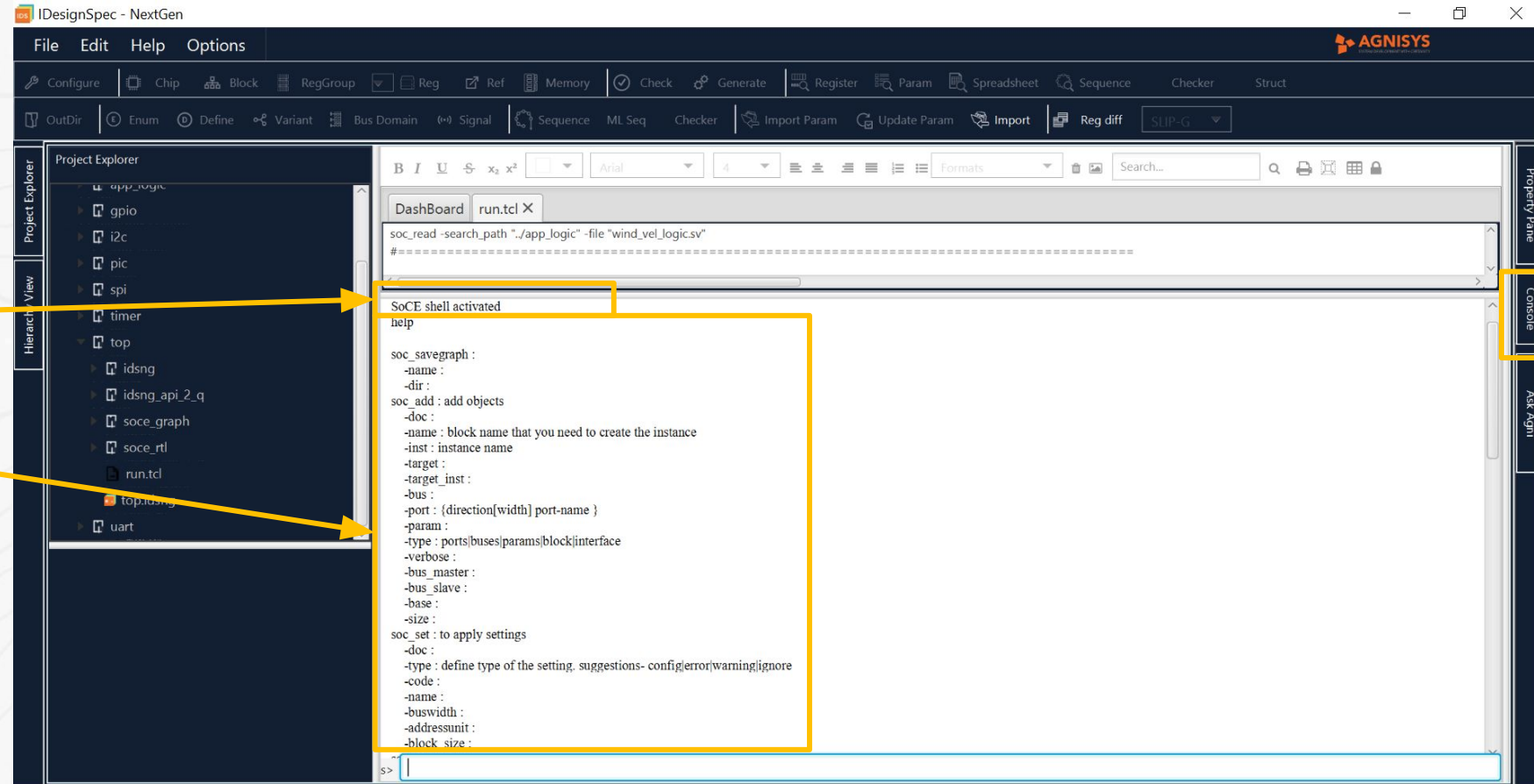
help

graph

showgraph

clear

cleanWorkspace



APIs Available for Assembly

- Following mentioned APIs can be either used in command mode or in a tcl script which can be further used in GUI mode

Configuration APIs

soc_set

Input APIs

soc_read

Creation APIs

soc_create
soc_add

Connection APIs

soc_connect
soc_promote

Graphical APIs

soc_graph
soc_savegraph

Generation APIs

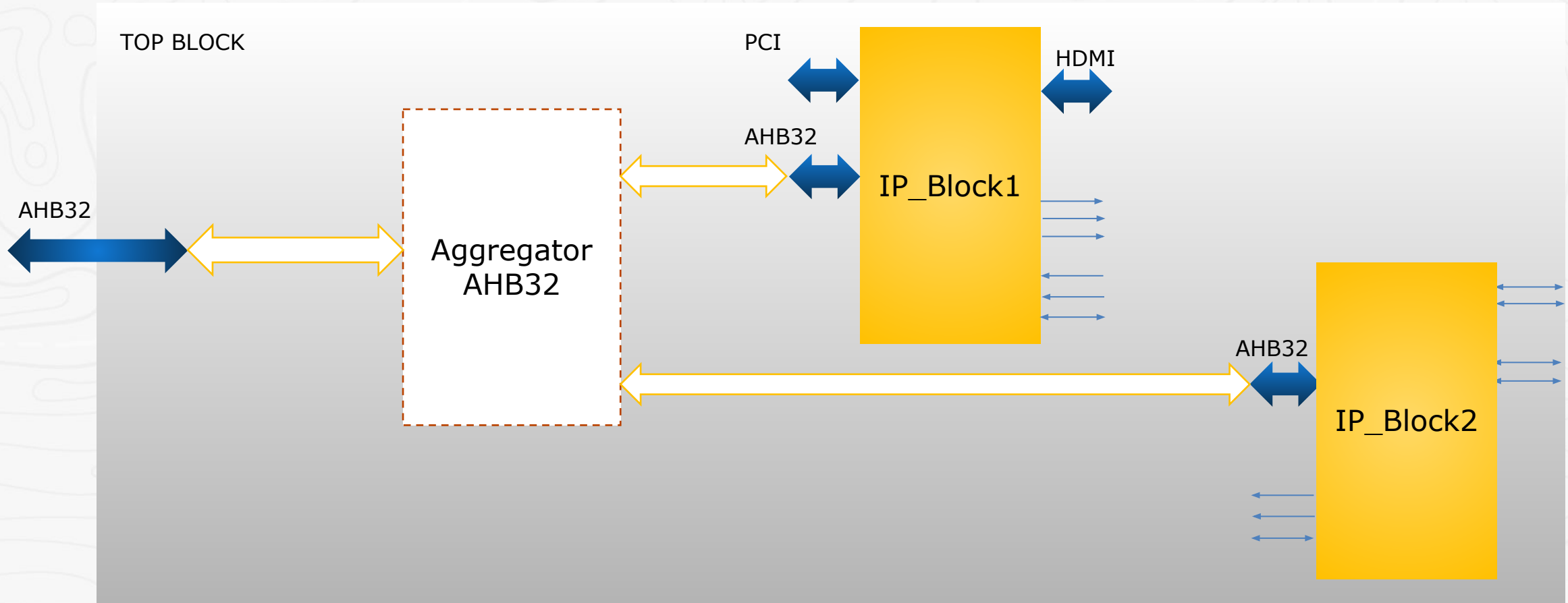
soc_generate

Features Available for SOC-Level Design

- SoC assembly using IDS-NG supports architectural level chip assembly
 - Ability to create and edit a design through script/command line interface
 - Automatically adds instances in the design, makes connections, restructures, etc
 - View the resulting schematics for analysis
 - Runs checks and generate different output collaterals for design, verification and software teams altogether
- Automatically generate major subsystems of an SoC design with flexibility to customize and/or configure for accommodating changes

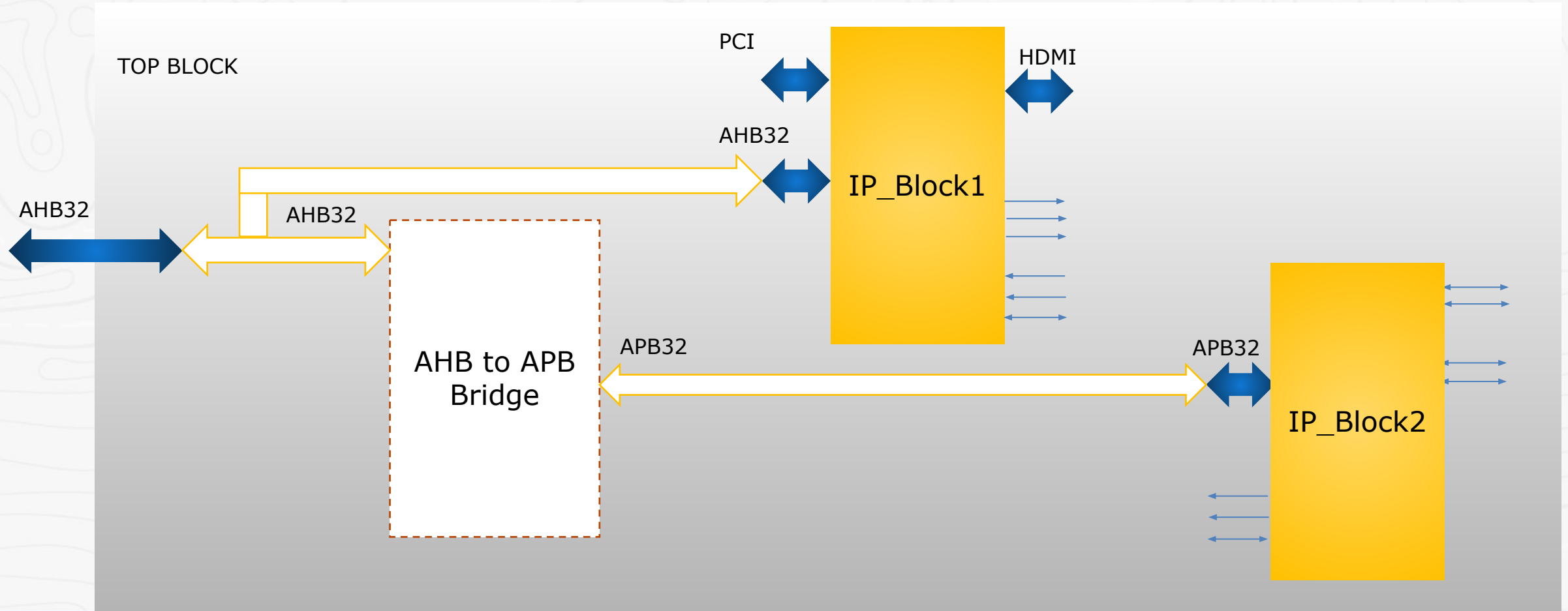
Features Available for SOC-Level Design - Contd..

- Auto generation of bus aggregators



Features Available for SOC-Level Design - Contd..

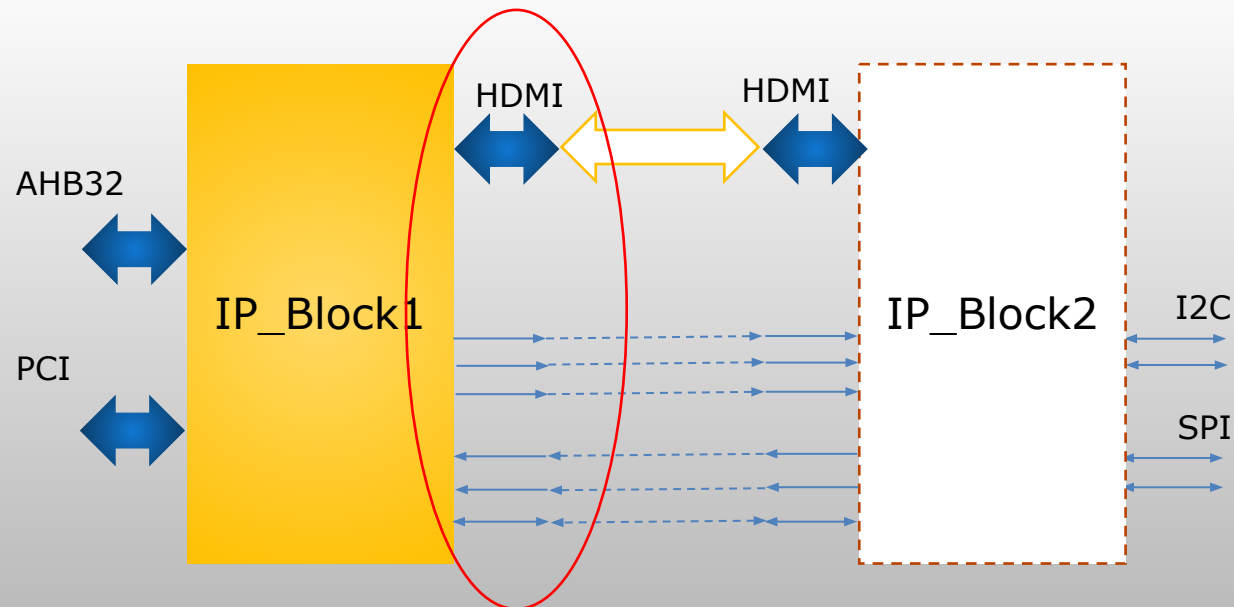
- Auto generation of bus bridges



Features Available for SOC-Level Design - Contd..

- Auto generation of mirrored block

TOP BLOCK



Sample Tcl Script

```
DashBoard | Graph | run.tcl | run.tcl X
so_read -search_path "./input" -file "aes_csr.v, aese_secure_csr.v, aesd_secure_csr.v"

so_create -type block -name HFC_INST -top -bus ahb
so_create -type block -name secure_csr_wrapper -bus apb

so_add -type block -target HFC_INST -name aes_csr_ids -inst aes_csr_ids_inst -base 0x0 -size 0x10000
so_add -type block -target HFC_INST -name secure_csr_wrapper -inst secure_csr_wrapper_inst

so_add -type block -target_inst secure_csr_wrapper_inst -name aese_secure_csr_ids -inst aese_secure_csr_ids_inst -base 0x10000 -size 0x10000
so_add -type block -target_inst secure_csr_wrapper_inst -name aesd_secure_csr_ids -inst aesd_secure_csr_ids_inst -base 0x20000 -size 0x10000

so_add -type block -target HFC_INST -name secure_csr_wrapper -inst secure_csr_wrapper_inst -base 0x30000 -size 0x40000

so_connect -source_inst aesd_secure_csr_ids_inst -dest_inst secure_csr_wrapper_inst -bus apb
so_connect -source_inst aese_secure_csr_ids_inst -dest_inst secure_csr_wrapper_inst -bus apb

so_connect -dest HFC_INST -source_inst secure_csr_wrapper_inst -bus apb

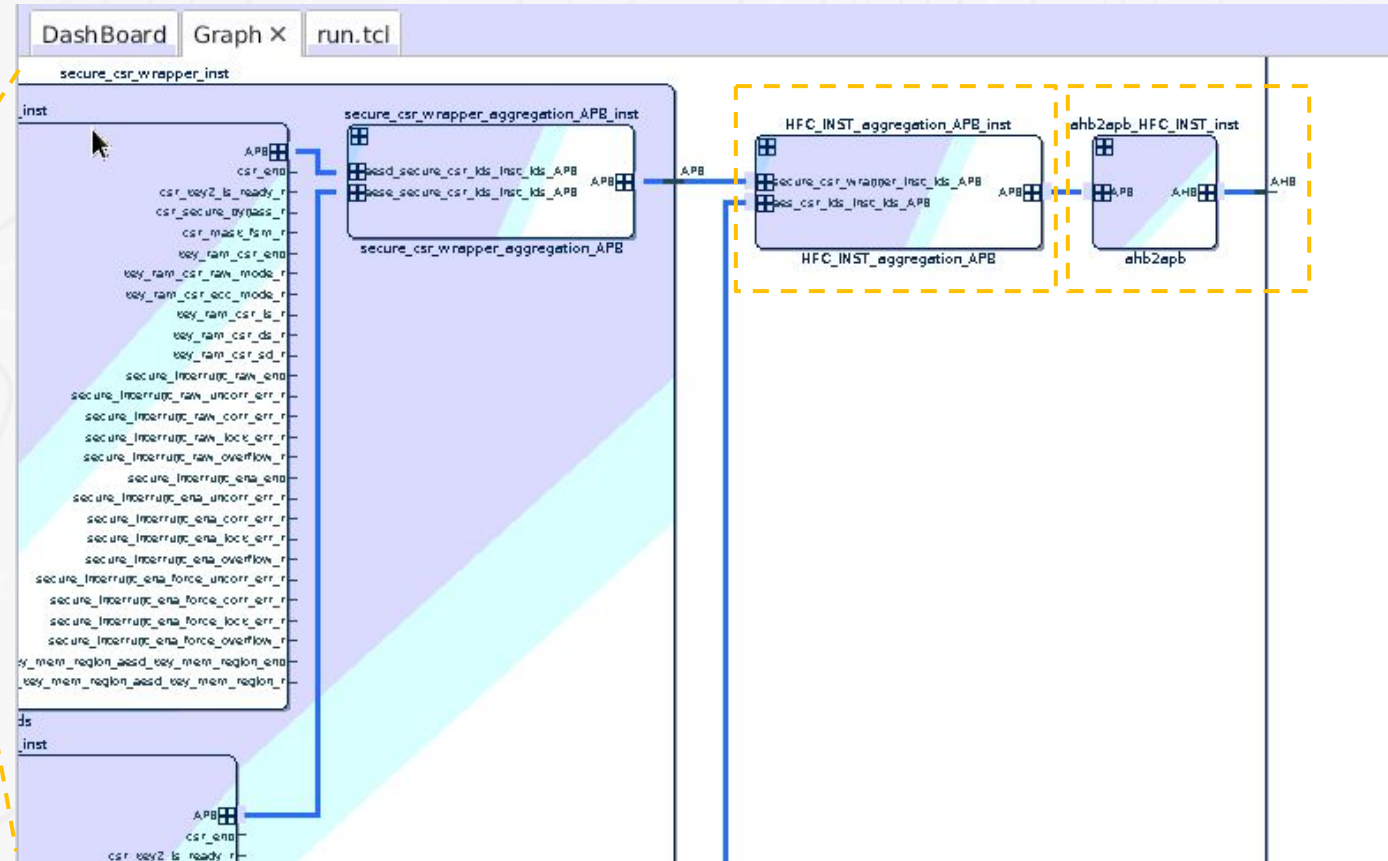
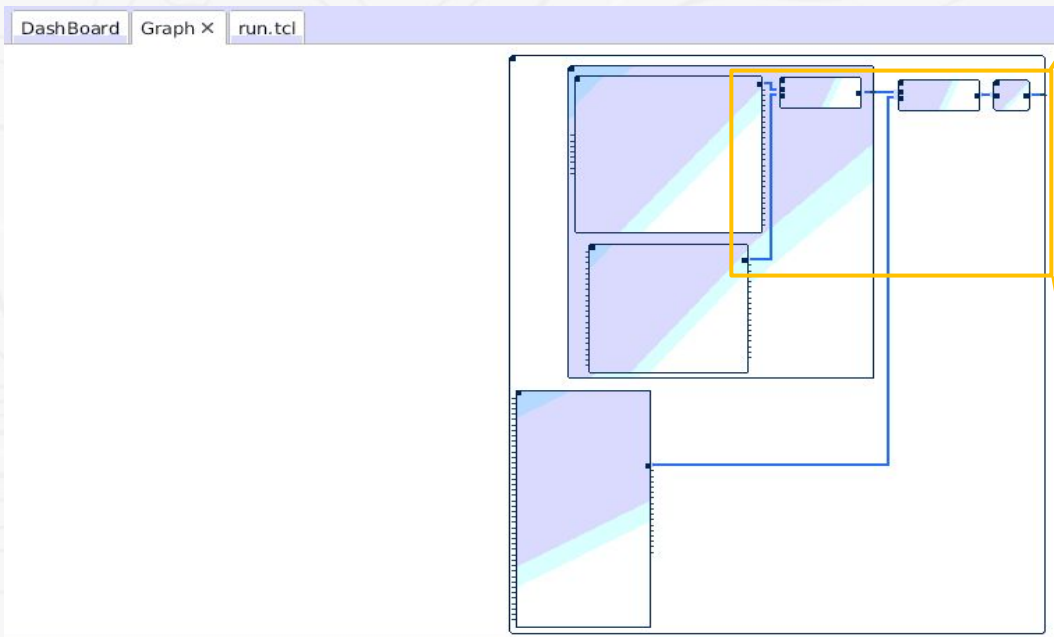
so_connect -dest HFC_INST -source_inst aes_csr_ids_inst -bus apb

so_savegraph -name "graph.nlv" -dir "ids"

so_generate -out {v,sv} -dir "ids"

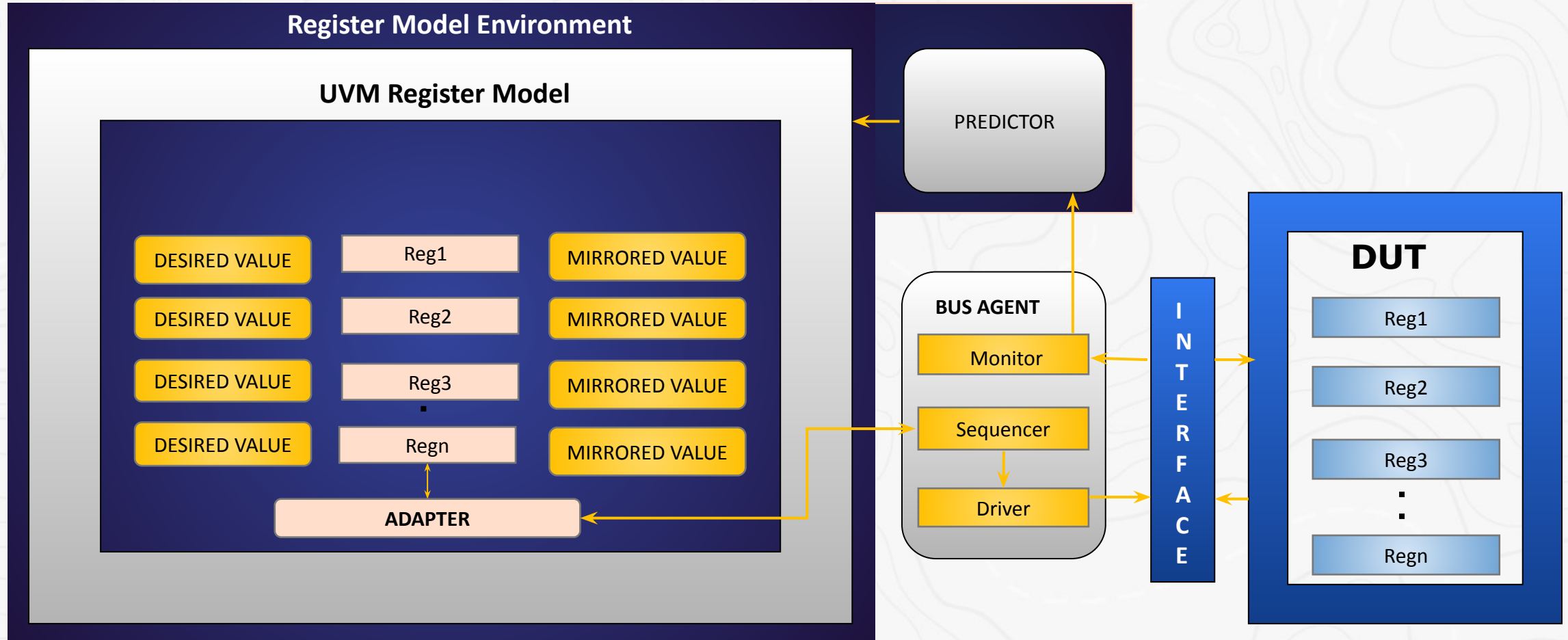
so_graph HFC_INST
```

Corresponding Schematic



Functional Verification

- Automatically generates UVM-RAL model and environment around it



Features Available for Register Verification

- Generates the complete UVM testbench: bus agents, monitors, drivers, adaptors, predictors, sequencers and sequences, as well as the Makefile and Verification Plan
- The UVM testbench is fully connected to the UVM Regmodel and DUT, providing you with a push-button verification
- Generates 100% functional coverage out of the box with register-focused coverage reports
- Generate sequences for special registers including lock, shadow, alias and interrupts registers
- Ability to import IP-XACT, SystemRDL, RALF, Word, Excel, CSV, or XML/YAML
- AMBA-AHB, AMBA-AHB3LITE, AMBA-APB, AMBA-AXI4LITE, AMBA-AXI4FULL, and Wishbone

Sample Sequence

- For lock register various sequence classes are generated, one such uvm sequence is as shown below

1	lock_ip			address 0x0
Properties..				
Description..				

1.1	primary_reg			address 0x0 default 0x00000000	
Description..					
bits	name	s/w	h/w	default	description
31:0	fld	rw	rw	0	

1.2	lock_reg			address 0x4 default 0x00000000	
lock=primary_reg;					
Description..					
bits	name	s/w	h/w	default	description
31:0	fld	rw	rw	0	

```
class uvm_pos_lock_pos_seq extends uvm_reg_sequence;
  `uvm_object_utils(uvm_pos_lock_pos_seq)
  uvm_reg rg;
  uvm_reg_field lock;
  .
  .
  .
  task body();
    uvm_reg_map maps[$];
    uvm_reg_map def_map;
    uvm_reg_map hw_map;
    .
    .
    .
    rg.get_maps(maps);
    foreach(maps[j])
      begin
        case(maps[j].get_name())
          default : def_map = maps[j];
        endcase
      end
    rg.read(stat,data, .path(UVM_FRONTDOOR),.map(def_map),.parent(this));
    `uvm_info("pos_lock_pos_test", {"writing/reading register '", rg.get_full_name(),""
    while unlocked through map '", def_map.get_full_name(), "' ..."}, UVM_LOW)
    lock.poke(stat,'b0);
    rwseq = uvm_reg_swRW_seq::type_id::create("rwseq");
    rwseq.rg = rg;
    rwseq.start(this.get_sequencer());
    `uvm_info("pos_lock_pos_test", {"reading register '", rg.get_full_name(),"" while
    locked through map '", def_map.get_full_name(), "' ..."}, UVM_LOW)
    lock.poke(stat,2**bitwidth -1);
    roseq = uvm_reg_swRO_seq::type_id::create("roseq");
    roseq.rg = rg;
    roseq.start(this.get_sequencer());
  endtask
endclass
```

Formal Verification

- Automatic Generation of
 - System Verilog properties and assertions to check the register access policies etc
 - Top-level file to bind the DUT RTL as well as third-party design IP with the assertions
 - Makefile or TCL command scripts
 - A verification plan with ability to back-annotate these formal results so that engineers can analyze the results
 - Comprehensive C/C++ tests for Firmware/Validation

Types of Check Generated

- Protocol compliance checking of Standard buses
 - AMBA-AHB
 - AMBA3AHBLITE
 - AMBA-APB
 - PROPRIETARY
 - AMBA-AXI
- Checks for repeat on address map & registers
- ARV Formal supports the following special registers
 - Lock Register
 - Shadow Register
- Reset checks for all fields
 - Whether reset value specified in specification, also holds true for DUT

Sample Assertion

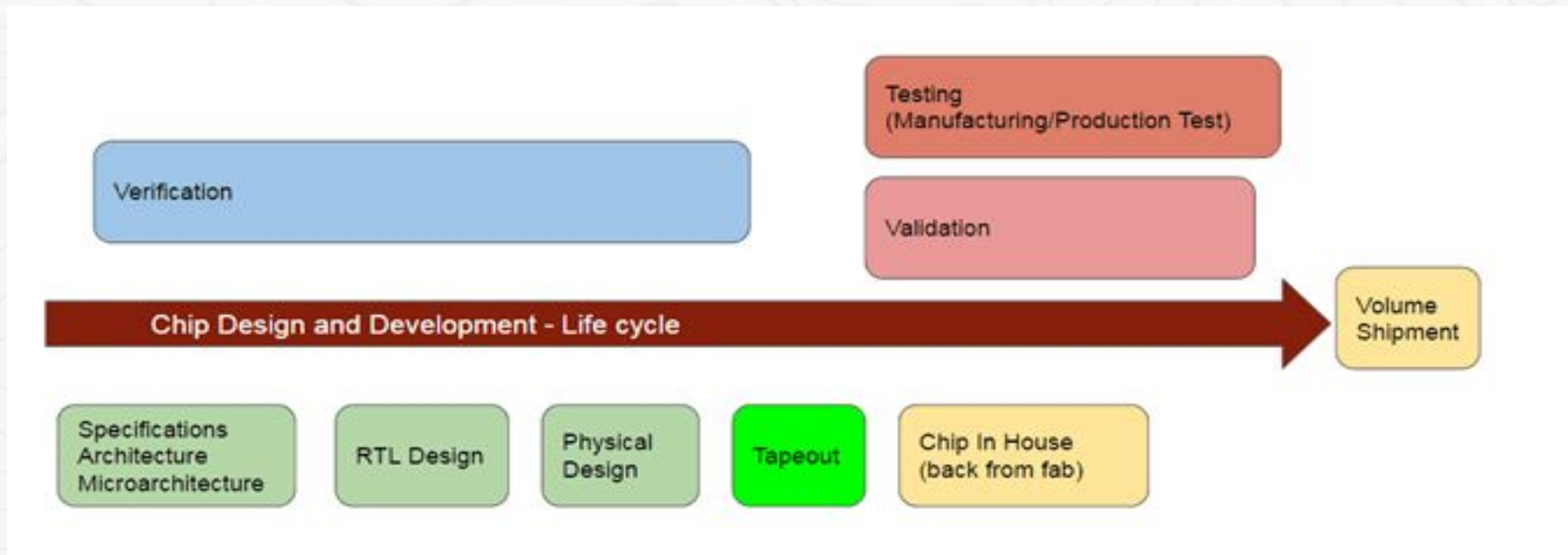
- Assertions for verifying special register, like lock register, is as shown below

1	lock_ip			address 0x0	
Properties..					
Description..					
1.1	primary_reg		32	address 0x0 default 0x00000000	
Description..					
bits	name	s/w	h/w	default	description
31:0	fld	rw	rw	0	
1.2	lock_reg		32	address 0x4 default 0x00000000	
lock=primary_reg;					
Description..					
bits	name	s/w	h/w	default	description
31:0	fld	rw	rw	0	

```
//----- FIELD ACCESS - primary_reg_fld - is rw-----  
  
property primary_reg_fld_write_check;  
  @(posedge pclk )disable iff (!presetn | primary_reg_fld_in_enb)  
  apb_write( 'h0,4'b1111)  
  |->  
  ($past(pslverr,1) == 0) && ( primary_reg_fld_q == $past(pwdata[31:0],1) );  
endproperty  
  
property primary_reg_fld_hw_write_check;  
  @(posedge pclk) disable iff (!presetn)  
  (primary_reg_fld_in_enb)  
  |->  
  ( primary_reg_fld_q == $past(primary_reg_fld_in,1) );  
endproperty  
  
property primary_reg_fld_hw_read_check;  
  @(posedge pclk) disable iff (!presetn)  
  (primary_reg_fld_q == primary_reg_fld_r );  
endproperty  
  
//----- FIELD ACCESS - lock_reg_fld - is rw-----  
  
property lock_reg_fld_read_check;  
  @(posedge pclk )disable iff (!presetn | lock_reg_wr_valid)  
  apb_read( 'h4,4'b1111)  
  |->  
  (pslverr == 0) && ( lock_reg_fld_q == prdata[31:0] );  
endproperty  
  
//----- FIELD ACCESS - lock_reg_fld - is rw-----  
  
property lock_reg_fld_lock_write_check;  
  @(posedge pclk )disable iff (!presetn | lock_reg_fld_in_enb)  
  apb_write( 'h4,4'b1111) ##0 !(primary_reg_rd_data)  
  |->  
  ($past(pslverr,1) == 0) && ( lock_reg_fld_q == $past(pwdata[31:0],1) );  
endproperty
```

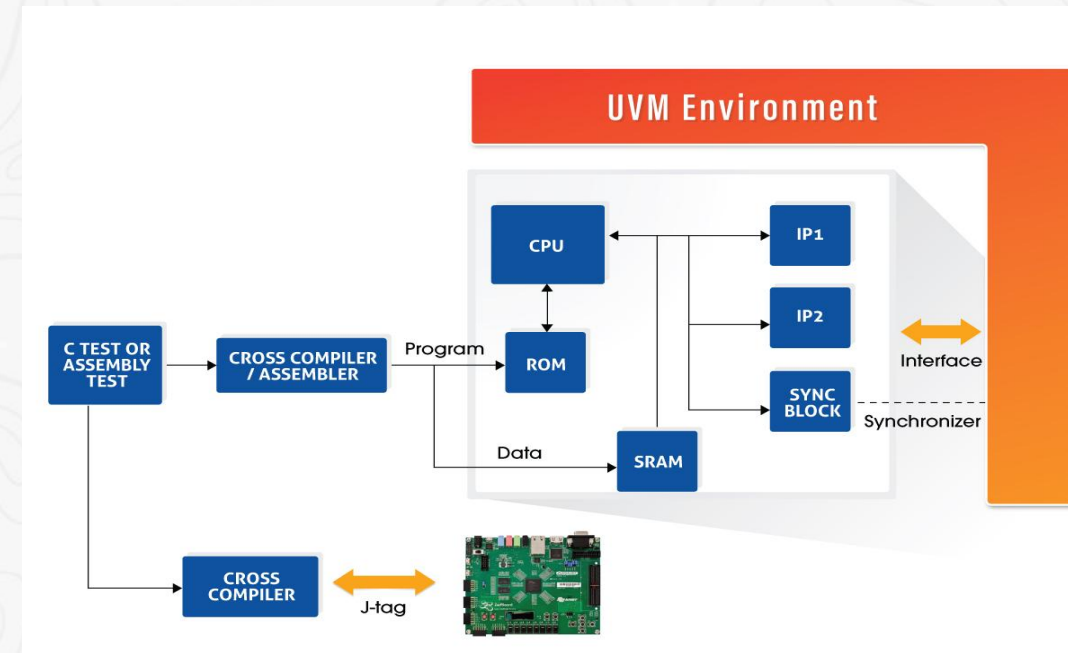
System Verification and Validation

- To test/verify a design against a given specification before the actual tape-out to ensure functional correctness
- Involves validating the chip in a system level environment with real software running on the hardware



Automatic Test Generation

- Process of sequence generation include
 - Positive and negative tests for different access types
 - Positive and negative tests to check functionality of special register including shadow, alias and interrupt registers
- Test sequences can be customised to generate 100% functional coverage out of the box with register-focused coverage reports



C Tests

- C-tests can be used to test IPs in a CPU simulation environment and can be run on CPU that is connected to IPs
- The tool environment generates and uses standard C-tests for a captured IP/registermap
- These C based standard tests includes
 - Random value test
 - Walking one tests
 - Write/read 0s and 1s tests
 - Special software register access test
 - Special register (like lock, shadow, alias, etc) tests are generated
 - Customized tests w.r.t supported properties

Sample C Test

- C Test are combination of read/write API's to configure the register and validate its firmware macros

```
int writeOnes()
{
    #ifndef block_name_Reg1_skip
    // BLOCK_NAME_REG1 //
    unsignedWord32 BLOCK_NAME_REG1_writeData = 0xffffffff & (block_name_Reg1_WRITEMASK) ;
    unsignedWord32 BLOCK_NAME_REG1_readData = BLOCK_NAME_REG1_writeData & block_name_Reg1_READMASK ;
    REG32_WRITE(baseAddress + block_name_Reg1_ADDRESS, BLOCK_NAME_REG1_writeData) ;
    READ32_CHK(baseAddress + block_name_Reg1_ADDRESS, BLOCK_NAME_REG1_readData) ;

    #endif

    return 0;
}

int writeZero()
{
    #ifndef block_name_Reg1_skip
    // BLOCK_NAME_REG1 //
    unsignedWord32 BLOCK_NAME_REG1_writeData = 0x00000000 & (block_name_Reg1_WRITEMASK) ;
    unsignedWord32 BLOCK_NAME_REG1_readData = BLOCK_NAME_REG1_writeData & block_name_Reg1_READMASK ;
    REG32_WRITE(baseAddress + block_name_Reg1_ADDRESS, BLOCK_NAME_REG1_writeData) ;
    READ32_CHK(baseAddress + block_name_Reg1_ADDRESS, BLOCK_NAME_REG1_readData) ;

    #endif

    return 0;
}

int writeWalkingOnes()
{
    #ifndef block_name_Reg1_skip
    // BLOCK_NAME_REG1 //
    unsignedWord32 BLOCK_NAME_REG1_writeData = 0x55555555 & (block_name_Reg1_WRITEMASK) ;
    unsignedWord32 BLOCK_NAME_REG1_readData = BLOCK_NAME_REG1_writeData & block_name_Reg1_READMASK ;
    REG32_WRITE(baseAddress + block_name_Reg1_ADDRESS, BLOCK_NAME_REG1_writeData) ;
    READ32_CHK(baseAddress + block_name_Reg1_ADDRESS, BLOCK_NAME_REG1_readData) ;

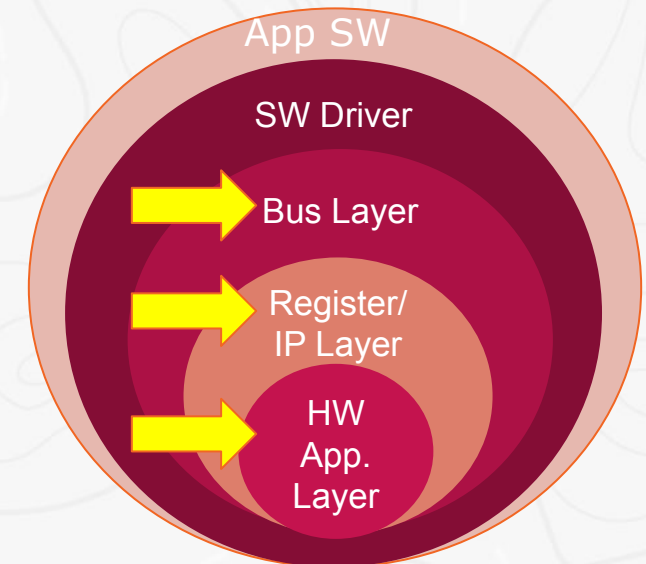
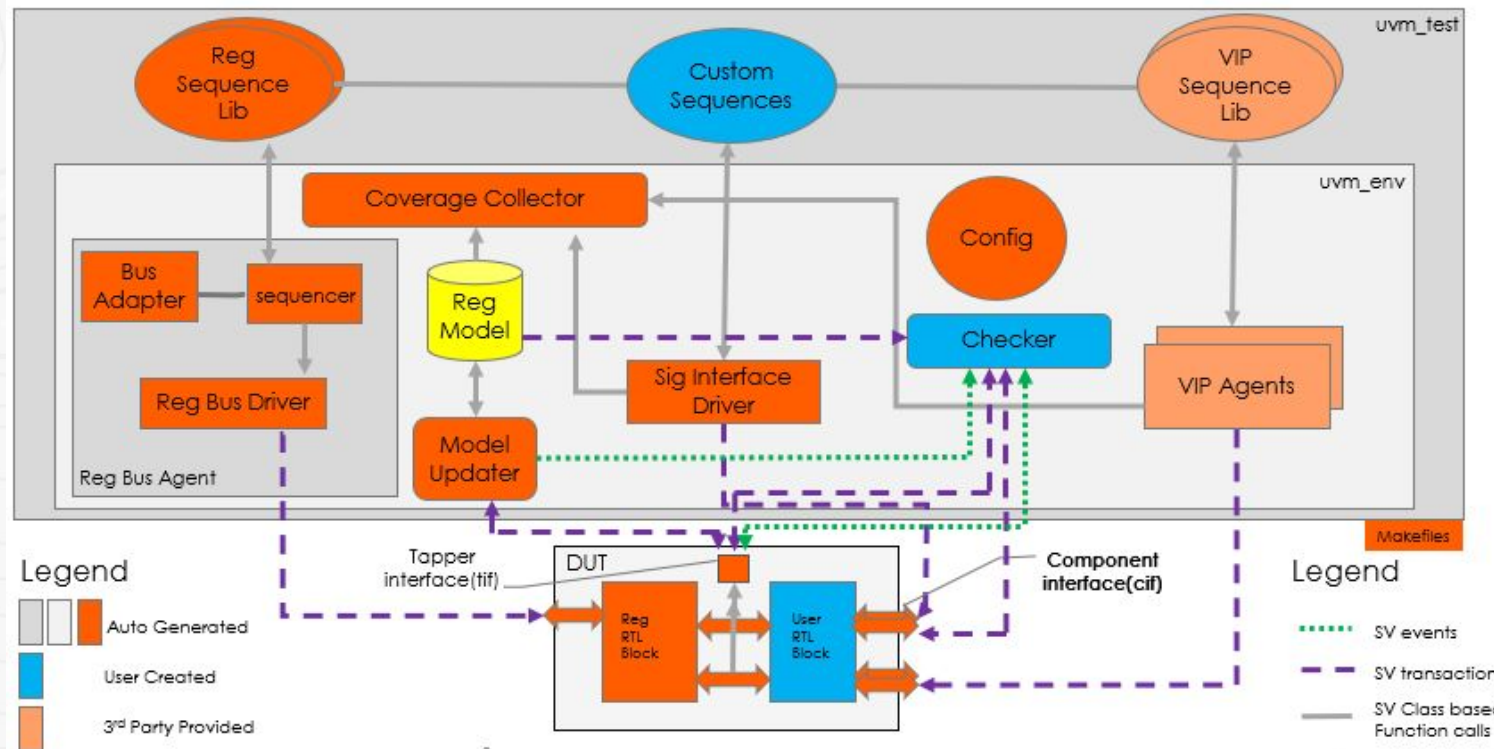
    #endif

    return 0;
}
```

These are the customizable read/write API's

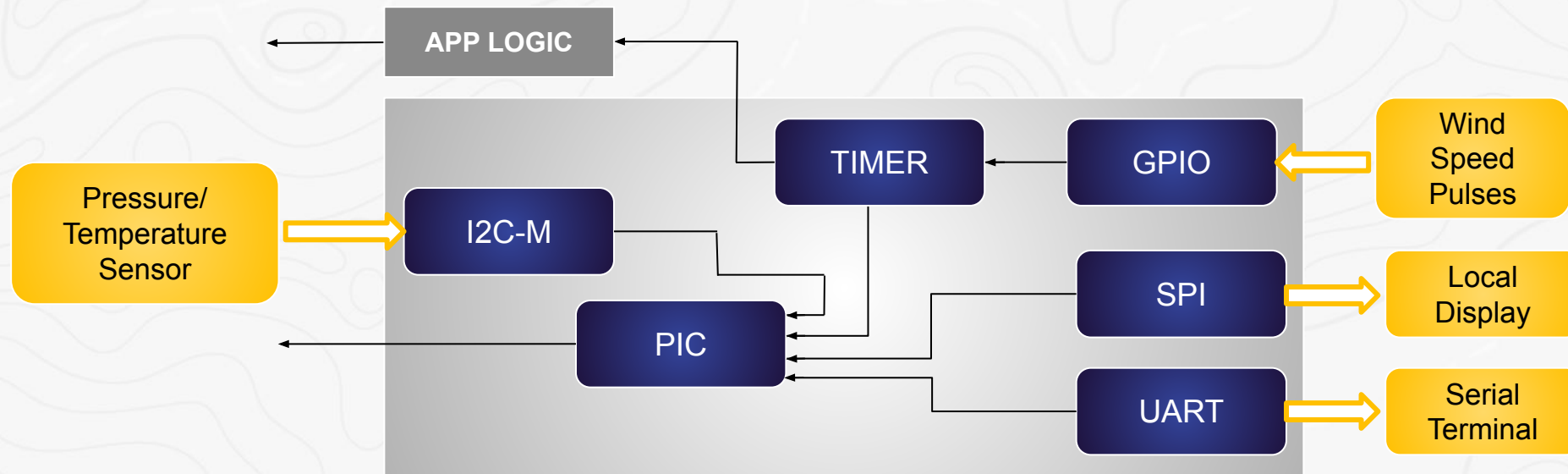
Automatic Verification

- UVM testbench generated automatically
- Tests from Register Map generated automatically
- Run custom sequences



Project - Weather Station

- A weather station measures atmospheric parameters and shares this data over a communication interface
- It typically has the following sensors
 - Atmospheric pressure – BMP280
 - Wind velocity – Tachometer pulses



Functional Description

- Wind speed Sensor
 - The total time for a round will be $t_h \times m \times n$, where t_h is cycle time for timer, n is total number of pulses in a round and m is recorded timer count between two pulses
 - Wind velocity = $(W \times 1000 \times 60) / (t_h \times m \times n)$
- Atmospheric Pressure
 - In the current design, the I2C interface is considered for the sensor and is programmed in Pressure mode



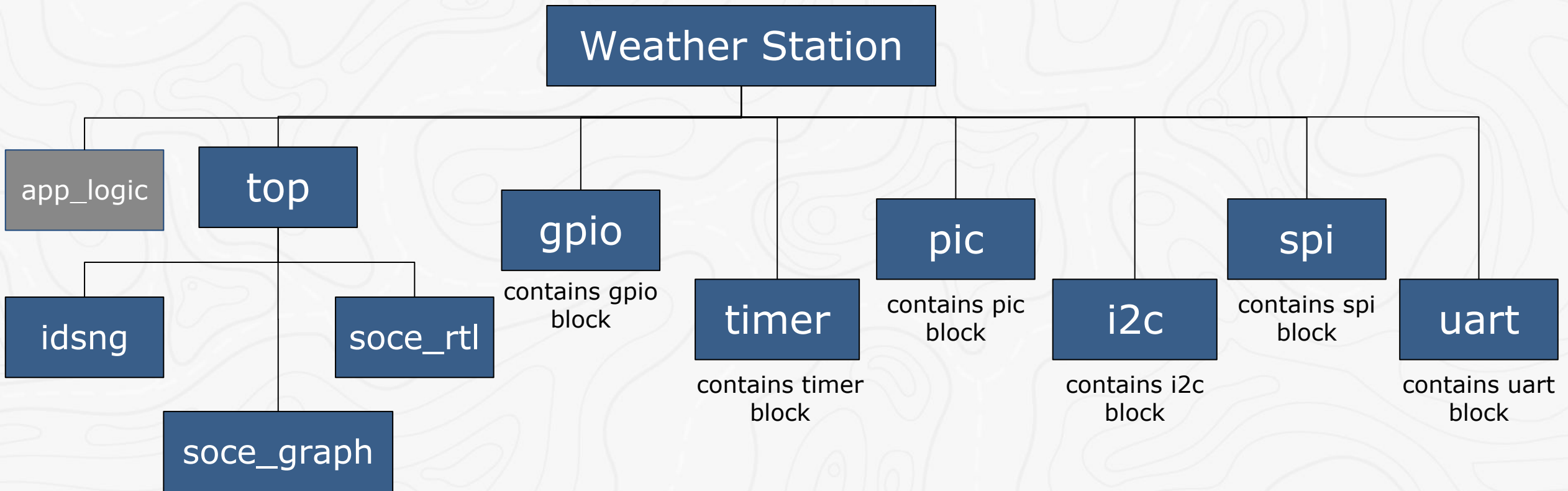
Functional Description - Contd..

- SPI Display
 - To send the commands and data to the local display
- UART Terminal
 - The weather station data is shared over a UART interface with the external world
 - Following is the ASCII format

#<Pressure in kg/sqcm>,<Wind velocity>



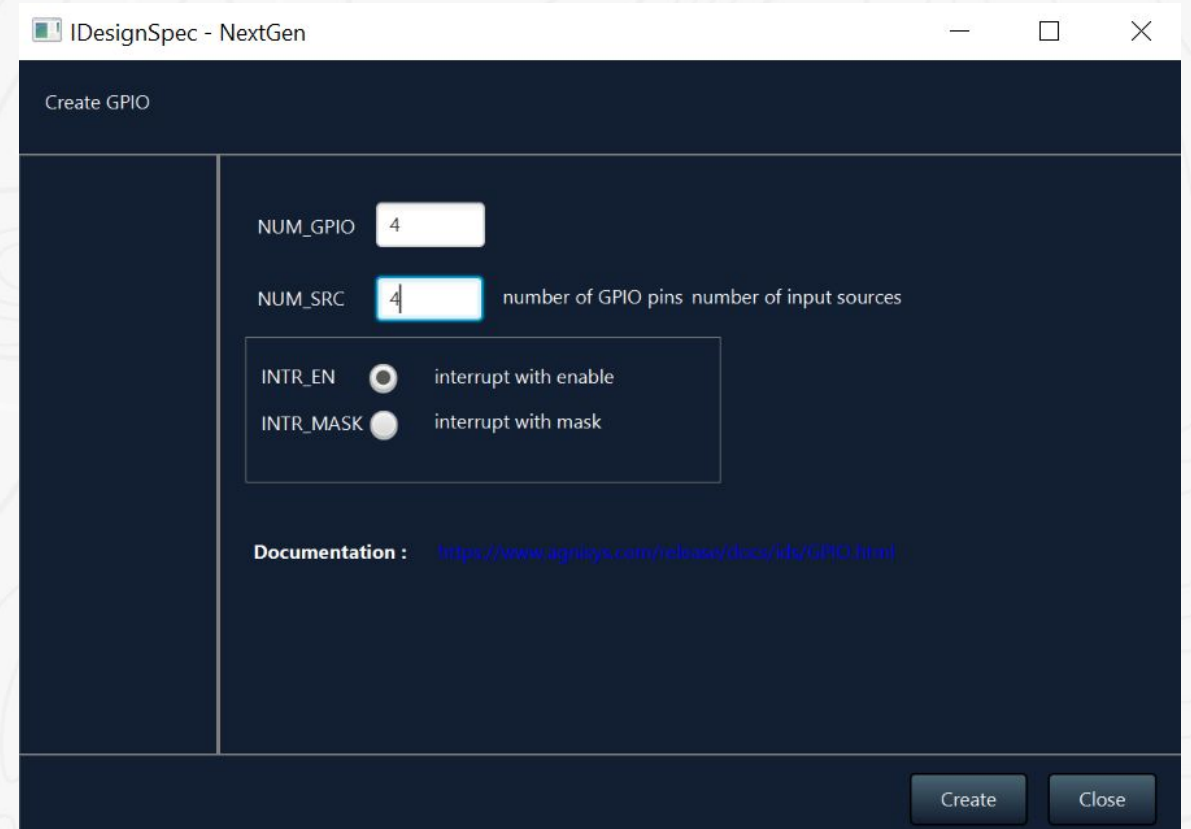
Directory Structure



- Contains top idsng file (containing reference to each ip)
- Contains rtl files in “idsng” directory (approx 12,730 lines)
- Contains tcl run file
- Contains graph and wrapper’s directory (containing wrapper top file, approx 240 lines) generated using soc-e

Details of Standard IPs

- GPIO
 - Bus interface: APB
 - Number of gpio pins: 4
 - Number of input sources: 4
 - Interrupt generation with enable
 - Pins are configured as input
 - Edge detection mode: can be used either in posedge or negedge



Details of Standard IPs - Contd..

- **TIMER**
 - Customised with registers to store result value, pulse count and wind velocity
 - Bus interface: APB
 - Number of TIMER: 1
 - Number of input sources: 4
 - Width of counter: 32-bit
 - Width of prescaler register: 32-bit
 - Incrementing counter mode
 - Running mode to count between two pulses
 - Interrupt generation with enable

The screenshot shows a configuration window titled "Create TIMER" from the IDesignSpec - NextGen tool. The window contains several parameters for configuring a timer IP:

- NUM_TIMER:** 1 (number of TIMER)
- NUM_SRC:** 4 (number of input sources)
- COUNTER_WIDTH:** 32 (counter width)
- PRESCALER_WIDTH:** 32 (prescaler width)
- COUNTER:** incr (counter with increment or decrement)

Below these parameters, there are two radio button options for interrupt generation:

- INTR_EN:** interrupt with enable
- INTR_MASK:** interrupt with mask

A "Documentation" link is provided at the bottom: <https://www.agnisys.com/release/docs/ids/TIMER.html>

At the bottom right of the window, there are two buttons: "Create" and "Close".

Details of Standard IPs - Contd..

- PIC
 - Bus interface: APB
 - Number of interrupt sources: 7
 - Software interface: 1
 - Vectored addressing
 - Interrupt clear functionality: interrupt clear bits are packed
 - Handling pending interrupt requests: usage of register read/write
 - Single source output and priority detection

IDesignSpec - NextGen

Create PIC

NUM_INTR_SRCS	<input type="text" value="7"/>	Number of interrupt sources
SOFT_INTR	<input type="text" value="1"/>	Number of software interface.
INTR_EN_CLR_PACK	<input type="text" value="1"/>	Software interrupt generation.
VECTORED	<input type="text" value="1"/>	Select for vectored interrupt.
NON_VECTORED	<input type="text" value="0"/>	Select for non_vectored interrupt.
REG_ACK	<input type="text" value="1"/>	Use register read/write for halting the other pending interrupt
PORT_ACK	<input type="text" value="0"/>	Use port for halting the other pending interrupt

INTR_EN interrupt with enable

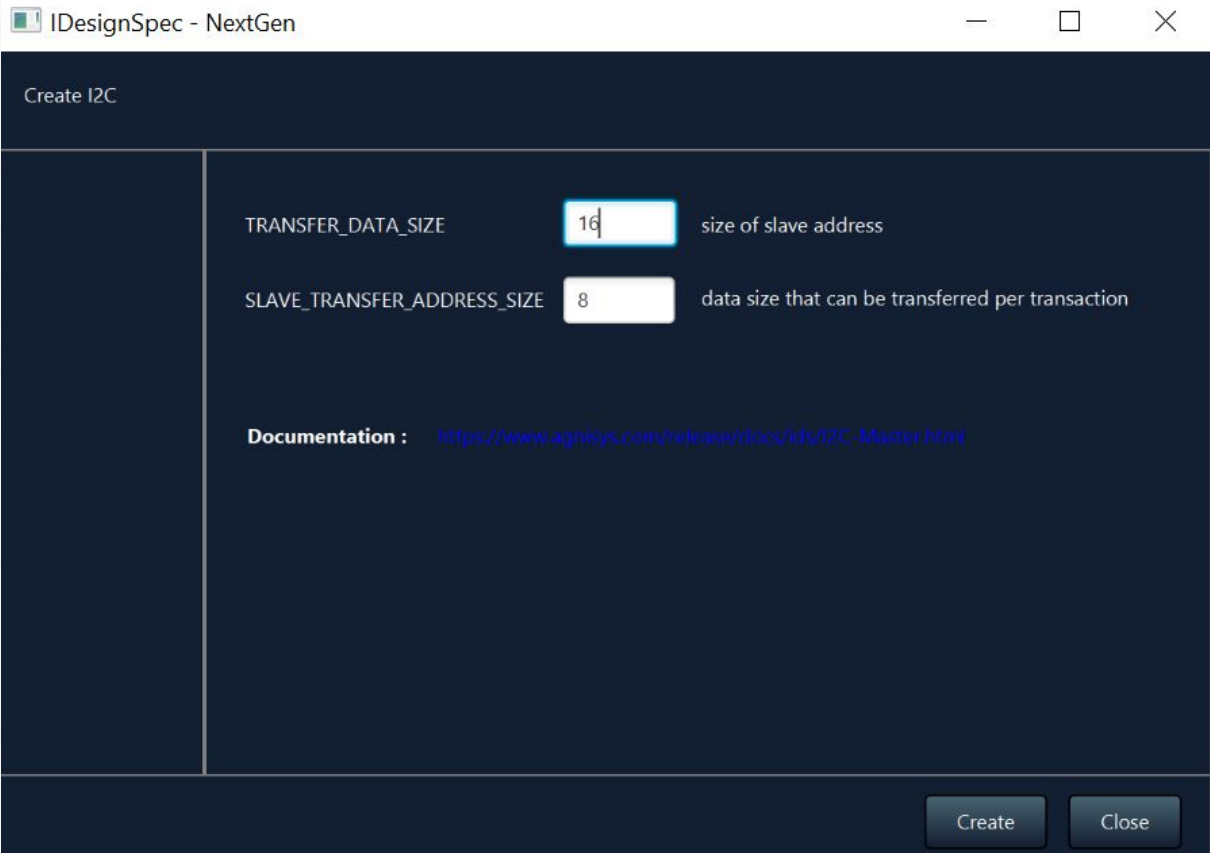
INTR_MASK interrupt with mask

Documentation : <https://www.agrways.com/release/docs/ide/PIC.html>

Create Close

Details of Standard IPs - Contd..

- I2C
 - Bus Interface: APB
 - data that can be transferred per transaction: 16
 - Size of slave address: 8
 - By default, interrupt generation is done via enable
 - I2C IP acts in receiving mode



IDesignSpec - NextGen

Create I2C

TRANSFER_DATA_SIZE size of slave address

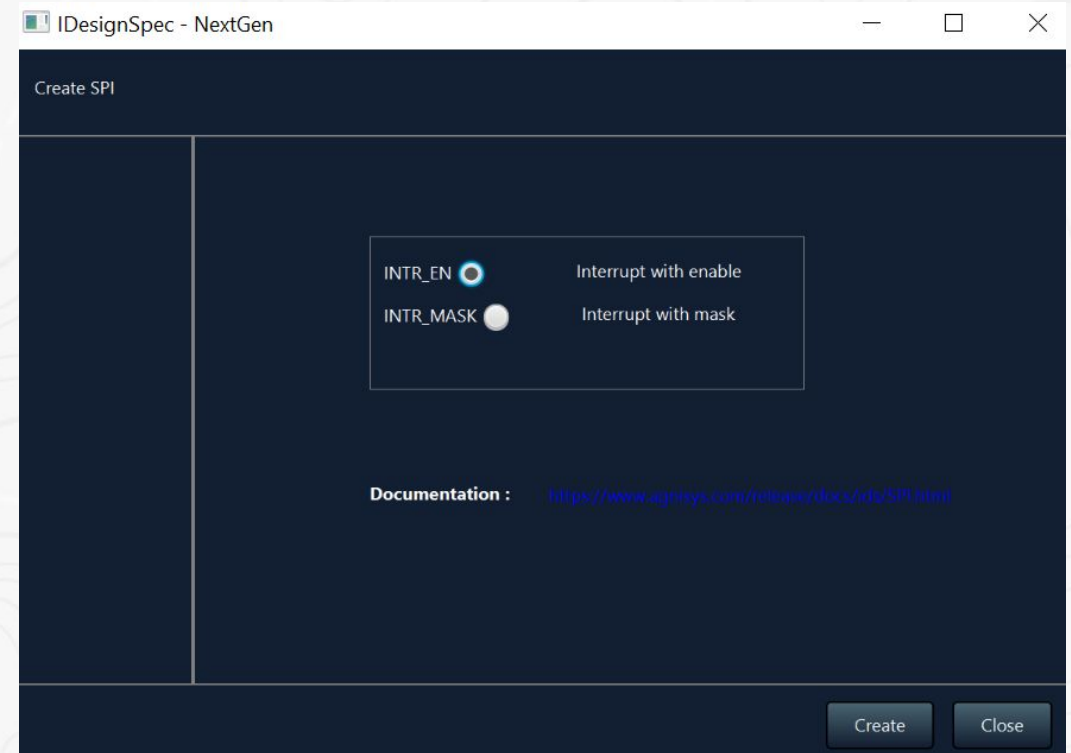
SLAVE_TRANSFER_ADDRESS_SIZE data size that can be transferred per transaction

Documentation : https://www.agnisys.com/release/docs/ids/I2C_Master.html

Create Close






Details of Standard IPs - Contd..






- SPI
 - Bus interface: APB
 - Interrupt generation with enable
 - SPI is configured in transmitting mode
- UART
 - Bus Interface: APB
 - 8 bits are transferred per character
 - Even parity is used
 - 1-stop bit detection and generation
 - UART is configured in transmitting mode



Top Containing Standard IPs

- Contains references to the standard IPs
- RTL generated of this top is used as input while assembling designs at SOC level

1	top				address 0x0
Properties..					
Description..					
1.1	gpio				address 0x0
name	gpio	path	../gpio/gpio.idsng		
Properties..					
Description..					
1.2	timer_inst1				address 0x24
name	timer	path	../timer/timer.idsng		
Properties..					
Description..					
1.3	timer_inst2				address 0x4C
name	timer	path	../timer/timer.idsng		
Properties..					
Description..					
1.4	timer_inst3				address 0x74
name	timer	path	../timer/timer.idsng		
Properties..					
Description..					

1.5	timer_inst4				address 0x9C
name	timer	path	../timer/timer.idsng		
Properties..					
Description..					
1.6	pic				address 0xC4
name	pic	path	../pic/pic.idsng		
Properties..					
Description..					
1.7	i2c				address 0x18C
name	i2cm	path	../i2c/i2c.idsng		
Properties..					
Description..					
1.8	spi				address 0x1AC
name	spimaster_csr	path	../spi/spi.idsng		
Properties..					
Description..					
1.9	uart				address 0x1D8
name	uart_regmap	path	../uart/uart.idsng		
Properties..					
Description..					

RTL of Top Specification

```
module top_ids #(
  // PARAMETERS
  parameter chip_offset = 'h0,
  parameter gpio_ids_offset = chip_offset + 'h0,
  parameter timer_inst1_ids_offset = chip_offset + 'h24,
  parameter timer_inst2_ids_offset = chip_offset + 'h4C,
  parameter timer_inst3_ids_offset = chip_offset + 'h74,
  parameter timer_inst4_ids_offset = chip_offset + 'h9C,
  parameter pic_ids_offset = chip_offset + 'hc4,
  parameter i2c_ids_offset = chip_offset + 'h18C,
  parameter spi_ids_offset = chip_offset + 'h1AC,
  parameter uart_ids_offset = chip_offset + 'h1D8,
  parameter bus_width = 32,
  parameter addr_width = 9,
  .
) (
  input [3 : 0] gpio_gpio_in0,
  output [3 : 0] gpio_gpio_out0,
  input gpio_clk,
  input gpio_reset,
  input [1 : 0] gpio_ext_src,
  output gpio_pin0_irq,
  output gpio_pin1_irq,
  output gpio_pin2_irq,
  output gpio_pin3_irq,
  output [4 -1:0] gpio_pin_cfg_out_en,
  input timer_inst1_clk,
  input timer_inst1_reset,
  input [1-1 : 0] timer_inst1_src,
  output timer_inst1_ids_irq,
  output [31 : 0] timer_inst1_idspulse_count_fld_r,
  output [31 : 0] timer_inst1_idsresult_val_fld_r,
  input [32- 1 : 0] timer_inst1_idswind_velocity_val_in,
  output [31 : 0] timer_inst1_idswind_velocity_val_r,
```

GPIO ports

TIMER instance
1 ports

```
  input timer_inst2_clk,
  input timer_inst2_reset,
  input [1-1 : 0] timer_inst2_src,
  .
  input [6 : 0] pic_intr_src,
  input pic_clk,
  input pic_reset,
  output pic_ids_irq,
  input i2c_sda_in,
  output i2c_sda_out,
  output i2c_scl_out,
  output i2c_sda_oen,
  output i2c_ids_irq,
  output spi_spi_clk,
  output [1:0] spi_spi_mode,
  output logic spi_MOSI,
  input logic spi_MISO,
  output logic spi_SS_n,
  output spi_ids_irq,
  input uart_clk,
  input uart_reset_l,
  output uart_ids_irq,
  output uart_tx_port,
  output [7:0] uart_rx_port,
  //APB signals
  .
  .
);
```

TIMER instance
2-4 ports

PIC ports

I2C ports

SPI ports

UART ports

RTL of Top Specification - Contd..

```
gpio top #(.addr width(addr width), .gpio offset(gpio ids offset)) gpio ip
wire timer_inst1_ids_select;
assign timer_inst1_ids_select = ( ((paddr >= chip_offset + 'h24) && (paddr <=
chip_offset + 'h4B) ) && (psel == 1'b1) && (invalid_address != 1'b1) ) ? 1'b1
: 1'b0;
timer_inst1 top #(.addr width(addr width), .timer_inst1 offset(
timer_inst1_ids offset)) timer ip
wire timer_inst2_ids_select;
assign timer_inst2_ids_select = ( ((paddr >= chip_offset + 'h4C) && (paddr <=
chip_offset + 'h73) ) && (psel == 1'b1) && (invalid_address != 1'b1) ) ? 1'b1
: 1'b0;
timer_inst2 top #(.addr width(addr width), .timer_inst2 offset(
timer_inst2_ids offset)) timer ip2
wire timer_inst3_ids_select;
assign timer_inst3_ids_select = ( ((paddr >= chip_offset + 'h74) && (paddr <=
chip_offset + 'h9B) ) && (psel == 1'b1) && (invalid_address != 1'b1) ) ? 1'b1
: 1'b0;
timer_inst3 top #(.addr width(addr width), .timer_inst3 offset(
timer_inst3_ids offset)) timer ip3
wire timer_inst4_ids_select;
assign timer_inst4_ids_select = ( ((paddr >= chip_offset + 'h9C) && (paddr <=
chip_offset + 'hC3) ) && (psel == 1'b1) && (invalid_address != 1'b1) ) ? 1'b1
: 1'b0;
timer_inst4 top #(.addr width(addr width), .timer_inst4 offset(
timer_inst4_ids offset)) timer ip4
```

```
wire pic_ids_select;
assign pic_ids_select = ( ((paddr >= chip_offset + 'hC4) && (paddr <=
chip_offset + 'h18B) ) && (psel == 1'b1) && (invalid_address != 1'b1) ) ? 1'b1
: 1'b0;
pic top #(.addr width(addr width), .bus width(bus width), .block offset(
pic_ids offset)) pic ip
wire i2c_ids_select;
assign i2c_ids_select = ( ((paddr >= chip_offset + 'h18C) && (paddr <=
chip_offset + 'h1AB) ) && (psel == 1'b1) && (invalid_address != 1'b1) ) ? 1'b1
: 1'b0;
i2c top #(.addr width(addr width), .block offset(i2c_ids offset)) i2cm ip
wire spi_ids_select;
assign spi_ids_select = ( ((paddr >= chip_offset + 'h1AC) && (paddr <=
chip_offset + 'h1D7) ) && (psel == 1'b1) && (invalid_address != 1'b1) ) ? 1'b1
: 1'b0;
spi_wrapper #(.addr width(addr width), .bus width(bus width), .fifo len(8))
spimaster csr ip
wire uart_ids_select;
assign uart_ids_select = ( ((paddr >= chip_offset + 'h1D8) && (paddr <=
chip_offset + 'h1FB) ) && (psel == 1'b1) && (invalid_address != 1'b1) ) ? 1'b1
: 1'b0;
uart top #(.addr width(addr width), .block offset(uart_ids offset))
uart topinst
assign prdata = gpio_ids_prdata | timer_inst1_ids_prdata |
timer_inst2_ids_prdata | timer_inst3_ids_prdata | timer_inst4_ids_prdata |
pic_ids_prdata | i2c_ids_prdata | spi_ids_prdata | uart_ids_prdata;
assign pready = gpio_ids_pready & timer_inst1_ids_pready &
timer_inst2_ids_pready & timer_inst3_ids_pready & timer_inst4_ids_pready &
pic_ids_pready & i2c_ids_pready & spi_ids_pready & uart_ids_pready;
assign pslverr = invalid_address | gpio_ids_pslverr | timer_inst1_ids_pslverr
| timer_inst2_ids_pslverr | timer_inst3_ids_pslverr | timer_inst4_ids_pslverr
| pic_ids_pslverr | i2c_ids_pslverr | spi_ids_pslverr | uart_ids_pslverr;

endmodule
```

TCL Script for Assembling Design at SOC Level

To read IPs or blocks in different formats like IP-XACT, RTL, IDS supported blocks

```
#=====
#Top level tcl
#=====
#Reading files
#=====
#Reading Top, including all the IPs files
soc_read -search_path "idsng" -file "top.sv" -inc_dir "idsng" -include
"apb_widget.v,gpio_edge_detect.v,gpio_sync_ff.v,gpio_detect_sync.v,gpio.v,gpio_top.v, sync_f
f.v,comp_vec.v,edge_detect.v,comp_vec_last.v,pic.v,pic_top.v,clockgen.sv,txn_fifo.sv,spi_rd
_txn.sv,spi_wr_txn.sv,spi.v,spi_core.sv,spi_wrapper.sv,i2cm_byte_transfer.v,i2c_block.v,i2c
.v,i2c_top.v,fifo.v,baud_rate_generator.v,uart_rx.v,uart_tx.v,uart.v,uart_top.v,edge_detect
_src.v,prescaler.v,edge_detect.v,timer_inst1.v,timer_core.v,timer_inst1_top.v,timer_inst2_t
op.v,timer_inst3_top.v,timer_inst4_top.v,"
#Reading App Logic for calculation of wind velocity
soc_read -search_path "../app_logic" -file "wind_vel_logic.sv"
#=====
#
#Creating Top Wrapper
#=====
soc_create -type block -name wrapper_top -port {output [3:0] gpio_out0, output [3:0]
pin_cfg_out_en, output irq_top, output uart_port, output logic spi_port, output i2c_port,
output [31:0] wind_velocity_timer1, output [31:0] wind_velocity_timer2, output [31:0]
wind_velocity_timer3, output [31:0] wind_velocity_timer4} -top -bus ahb
#=====
```

To create the template and the instance like block, interface, etc

TCL Script - Contd..

To add an instance of an IP/block whose template is already present in the memory either by using a read api or a create api

```
#####
#Adding files to the wrapper
#####
#Adding top to the wrapper
soc_add -type block -target wrapper_top -name top_ids -inst top_ids_inst

#Adding Wind Velocity App logic to the TOP wrapper for timer 1
soc_add -type block -target wrapper_top -name wind_vel_logic -inst wind_vel_logic_inst1

#Adding Wind Velocity App logic to the TOP wrapper for timer 2
soc_add -type block -target wrapper_top -name wind_vel_logic -inst wind_vel_logic_inst2

#Adding Wind Velocity App logic to the TOP wrapper for timer 3
soc_add -type block -target wrapper_top -name wind_vel_logic -inst wind_vel_logic_inst3

#Adding Wind Velocity App logic to the TOP wrapper for timer 4
soc_add -type block -target wrapper_top -name wind_vel_logic -inst wind_vel_logic_inst4
#####

#####
#Creating instances at the wrapper file
#####
#Connecting top instance to the wrapper file
soc_connect -dest wrapper_top -source_inst top_ids_inst -bus apb

#Connecting Wind Velocity App Logic to the TOP wrapper for timer 1
soc_connect -dest wrapper_top -source_inst wind_vel_logic_inst1 -bus apb

#Connecting Wind Velocity App Logic to the TOP wrapper for timer 2
soc_connect -dest wrapper_top -source_inst wind_vel_logic_inst2 -bus apb

#Connecting Wind Velocity App Logic to the TOP wrapper for timer 3
soc_connect -dest wrapper_top -source_inst wind_vel_logic_inst3 -bus apb

#Connecting Wind Velocity App Logic to the TOP wrapper for timer 4
soc_connect -dest wrapper_top -source_inst wind_vel_logic_inst4 -bus apb
#####
```

To connect the instances of IPs/blocks together within a container or connecting them with the container itself

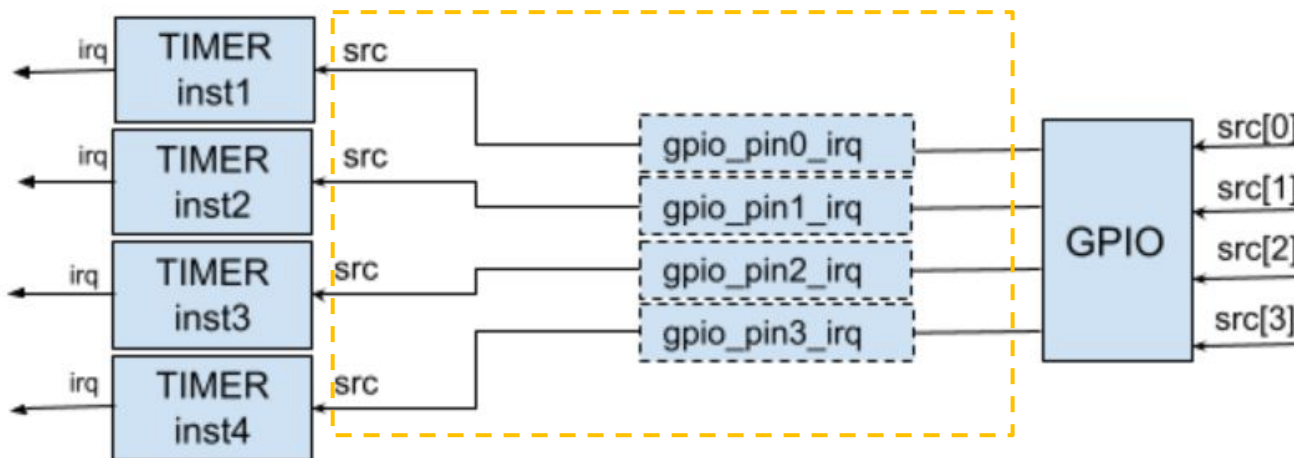
TCL Script - Contd..

```
#=====
#GPIO connections with TIMER
#=====
#Connecting timer_ip "src" port with gpio_ip "gpio_pin0_irq" port
soc_connect -source_inst top_ids_inst.gpio_pin0_irq -dest_inst top_ids_inst.timer_inst1_src

#Connecting timer_ip "src" port with gpio_ip "gpio_pin1_irq" port
soc_connect -source_inst top_ids_inst.gpio_pin1_irq -dest_inst top_ids_inst.timer_inst2_src

#Connecting timer_ip "src" port with gpio_ip "gpio_pin2_irq" port
soc_connect -source_inst top_ids_inst.gpio_pin2_irq -dest_inst top_ids_inst.timer_inst3_src

#Connecting timer_ip "src" port with gpio_ip "gpio_pin3_irq" port
soc_connect -source_inst top_ids_inst.gpio_pin3_irq -dest_inst top_ids_inst.timer_inst4_src
#=====
```



TCL Script - Contd..

```
=====
#Interrupt connections with PIC
#
#Connecting timer_ip "timer_inst1_ids_irq" port with pic_ip "src[0]" port
soc_connect -source_inst top_ids_inst.timer_inst1_ids_irq -dest_inst
top_ids_inst.pic_intr_src[0]

#Connecting timer_ip "timer_inst2_ids_irq" port with pic_ip "src[1]" port
soc_connect -source_inst top_ids_inst.timer_inst2_ids_irq -dest_inst
top_ids_inst.pic_intr_src[1]

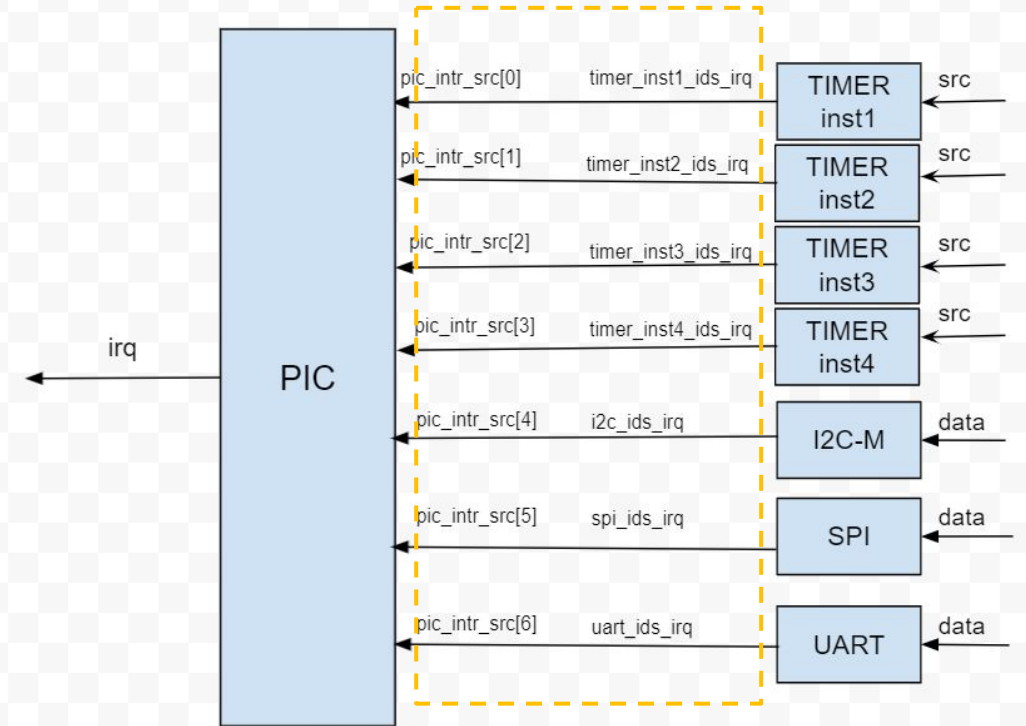
#Connecting timer_ip "timer_inst3_ids_irq" port with pic_ip "src[2]" port
soc_connect -source_inst top_ids_inst.timer_inst3_ids_irq -dest_inst
top_ids_inst.pic_intr_src[2]

#Connecting timer_ip "timer_inst4_ids_irq" port with pic_ip "src[3]" port
soc_connect -source_inst top_ids_inst.timer_inst4_ids_irq -dest_inst
top_ids_inst.pic_intr_src[3]

#Connecting i2cm_ip "i2c_ids_irq" port with pic_ip "src[4]" port
soc_connect -source_inst top_ids_inst.i2c_ids_irq -dest_inst top_ids_inst.pic_intr_src[4]

#Connecting spimaster_csr_ip "spi_ids_irq" port with pic_ip "src[5]" port
soc_connect -source_inst top_ids_inst.spi_ids_irq -dest_inst top_ids_inst.pic_intr_src[5]

#Connecting uart_topinst "uart_ids_irq" port with pic_ip "src[6]" port
soc_connect -source_inst top_ids_inst.uart_ids_irq -dest_inst top_ids_inst.pic_intr_src[6]
=====
```



TCL Script - Contd..

```
#=====
#Connecting IPs "clk" port with wrapper_top "hclk" port
#=====
soc_connect -source_inst top_ids_inst.gpio_clk -dest wrapper_top.hclk
soc_connect -source_inst top_ids_inst.timer_inst1_clk -dest wrapper_top.hclk
soc_connect -source_inst top_ids_inst.timer_inst2_clk -dest wrapper_top.hclk
soc_connect -source_inst top_ids_inst.timer_inst3_clk -dest wrapper_top.hclk
soc_connect -source_inst top_ids_inst.timer_inst4_clk -dest wrapper_top.hclk
soc_connect -source_inst top_ids_inst.pic_clk -dest wrapper_top.hclk
soc_connect -source_inst top_ids_inst.uart_clk -dest wrapper_top.hclk
#=====

#=====
#Connecting IPs "reset" port with wrapper_top "hresetn" port
#=====
soc_connect -source_inst top_ids_inst.gpio_reset -dest wrapper_top.hresetn
soc_connect -source_inst top_ids_inst.timer_inst1_reset -dest wrapper_top.hresetn
soc_connect -source_inst top_ids_inst.timer_inst2_reset -dest wrapper_top.hresetn
soc_connect -source_inst top_ids_inst.timer_inst3_reset -dest wrapper_top.hresetn
soc_connect -source_inst top_ids_inst.timer_inst4_reset -dest wrapper_top.hresetn
soc_connect -source_inst top_ids_inst.pic_reset -dest wrapper_top.hresetn
soc_connect -source_inst top_ids_inst.uart_reset_1 -dest wrapper_top.hresetn
#=====

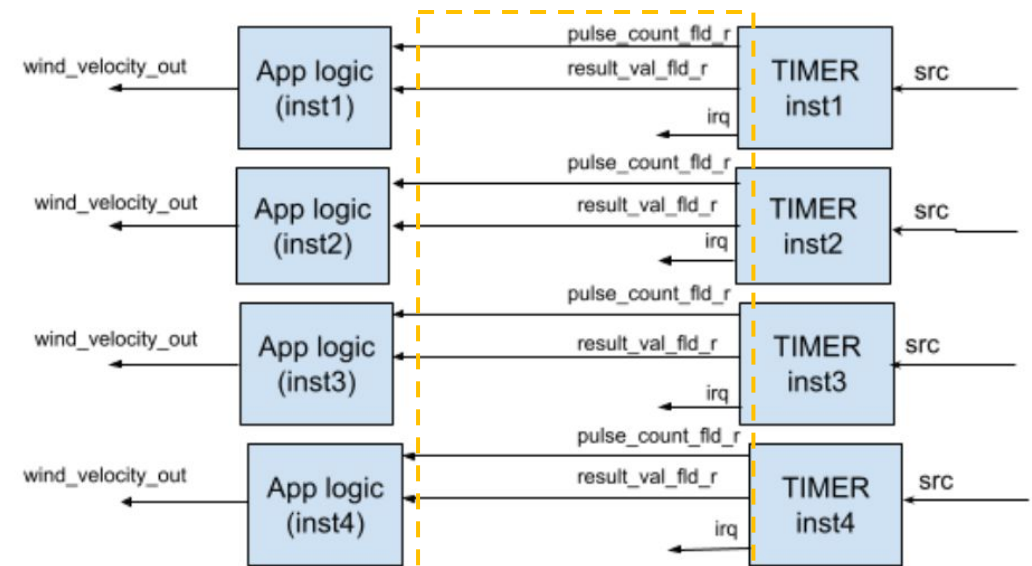
#=====
#Connecting IPs input port with 0
#=====
soc_connect -source_inst top_ids_inst -port {gpio_ext_src,gpio_gpio_in0,spi_MISO} -tie 0
#=====
```

TCL Script - Contd..

```
#=====
#Connecting IP output ports with wrapper_top output ports
#=====
soc_connect -source_inst top_ids_inst.gpio_gpio_out0 -dest wrapper_top.gpio_out0
soc_connect -source_inst top_ids_inst.gpio_pin_cfg_out_en -dest wrapper_top.pin_cfg_out_en
soc_connect -source_inst top_ids_inst.pic_ids_irq -dest wrapper_top.irq_top
soc_connect -source_inst top_ids_inst.spi_MOSI -dest wrapper_top.spi_port
soc_connect -source_inst top_ids_inst.uart_tx_port -dest wrapper_top.uart_port
soc_connect -source_inst top_ids_inst.i2c_sda_out -dest wrapper_top.i2c_port
soc_connect -source_inst top_ids_inst.timer_inst1_idswind_velocity_val_r -dest
wrapper_top.wind_velocity_timer1
soc_connect -source_inst top_ids_inst.timer_inst2_idswind_velocity_val_r -dest
wrapper_top.wind_velocity_timer2
soc_connect -source_inst top_ids_inst.timer_inst3_idswind_velocity_val_r -dest
wrapper_top.wind_velocity_timer3
soc_connect -source_inst top_ids_inst.timer_inst4_idswind_velocity_val_r -dest
wrapper_top.wind_velocity_timer4
#=====
```

TCL Script - Contd..

```
#####  
#Connecting Timers to app logic for wind velocity calculation  
#####  
soc_connect -source_inst top_ids_inst.timer_inst1_idspulse_count_fld_r -dest_inst  
wind_vel_logic_inst1.pulse_count_fld_in  
soc_connect -source_inst top_ids_inst.timer_inst1_idsresult_val_fld_r -dest_inst  
wind_vel_logic_inst1.result_val_fld_in  
soc_connect -source_inst top_ids_inst.timer_inst1_idswind_velocity_val_in -dest_inst  
wind_vel_logic_inst1.wind_velocity_out  
  
soc_connect -source_inst top_ids_inst.timer_inst2_idspulse_count_fld_r -dest_inst  
wind_vel_logic_inst2.pulse_count_fld_in  
soc_connect -source_inst top_ids_inst.timer_inst2_idsresult_val_fld_r -dest_inst  
wind_vel_logic_inst2.result_val_fld_in  
soc_connect -source_inst top_ids_inst.timer_inst2_idswind_velocity_val_in -dest_inst  
wind_vel_logic_inst2.wind_velocity_out  
  
soc_connect -source_inst top_ids_inst.timer_inst3_idspulse_count_fld_r -dest_inst  
wind_vel_logic_inst3.pulse_count_fld_in  
soc_connect -source_inst top_ids_inst.timer_inst3_idsresult_val_fld_r -dest_inst  
wind_vel_logic_inst3.result_val_fld_in  
soc_connect -source_inst top_ids_inst.timer_inst3_idswind_velocity_val_in -dest_inst  
wind_vel_logic_inst3.wind_velocity_out  
  
soc_connect -source_inst top_ids_inst.timer_inst4_idspulse_count_fld_r -dest_inst  
wind_vel_logic_inst4.pulse_count_fld_in  
soc_connect -source_inst top_ids_inst.timer_inst4_idsresult_val_fld_r -dest_inst  
wind_vel_logic_inst4.result_val_fld_in  
soc_connect -source_inst top_ids_inst.timer_inst4_idswind_velocity_val_in -dest_inst  
wind_vel_logic_inst4.wind_velocity_out  
#####
```



Generated Wrapper File

```
`include "../..//app_logic/wind_vel_logic.sv"
`include "../idsng/top.sv"
`include "ahb2apb.v"
module wrapper_top #(
    parameter addr_width = 'h20,
    parameter bus_width = 'h20,
    parameter wrapper_top_offset = 'h0,
    parameter top_ids_offset = 'h0
) (
    output irq_top ,
    output uart_port ,
    output spi_port ,
    input i2c_data ,
    input [3:0] ext_src ,
    input hclk ,
    input hresetn ,
    input hwrite ,
    input [1:0] htrans ,
    input [2:0] hsize ,
    input [2:0] hburst ,
    input [3:0] hprot ,
    input [bus_width-1:0] hwdata ,
    input [addr_width-1:0] haddr ,
    input hsel ,
    output [bus_width-1:0] hrdata ,
    output hready ,
    output [1:0] hresp
);
```

```
wire wire_ahb2apb_wrapper_top_inst_pclk_top_ids_instpclk;
wire wire_ahb2apb_wrapper_top_inst_presetn_top_ids_instpresetn;
. . .
wind_vel_logic #( .timer_freq('h2faf080), .wind_constant('h3a980)) wind_vel_logic_inst4(
    .pulse_count_fld_in(
wire_wind_vel_logic_inst4_pulse_count_fld_in_top_ids_insttimer_inst4_idspulse_count_fld_r),
. . .
);
top_ids #( .NUM_TIMER('h1), .NUM_SRC('h1), .COUNTER_WIDTH('h20), .PRESCALER_WIDTH('h20), .
NUM_INTR_SRCS('h7), .TRANSFER_DATA_SIZE('h10), .SLAVE_TRANSFER_ADDRESS_SIZE('h8), .
chip_offset(top_ids_offset), .gpio_ids_offset(top_ids_offset), .timer_inst1_ids_offset(
top_ids_offset), .timer_inst2_ids_offset(top_ids_offset), .timer_inst3_ids_offset(
top_ids_offset), .timer_inst4_ids_offset(top_ids_offset), .pic_ids_offset(top_ids_offset),
.i2c_ids_offset(top_ids_offset), .spi_ids_offset(top_ids_offset), .uart_ids_offset(
top_ids_offset), .bus_width('h20), .addr_width('h20), .pic_idsintr_cfg_count('h7), .
pic_idsintr_cfg_address_width('h0), .pic_idsstatus_count('h7), .pic_idsstatus_address_width(
'h0), .pic_idsenable_count('h7), .pic_idsenable_address_width('h0), .pic_idspending_count(
'h7), .pic_idspending_address_width('h0), .pic_idspost_count('h7), .
pic_idspost_address_width('h0), .pic_idspriority_reg_count('h7), .
pic_idspriority_reg_address_width('h0), .pic_idsvect_addr_count('h7), .
pic_idsvect_addr_address_width('h0)) top_ids_inst(
    .gpio_gpio_in0(4'h0),
    .gpio_gpio_out0(),
    .gpio_clk(hclk),
    .gpio_reset(hresetn),
    .gpio_ext_src(ext_src),
    .gpio_pin0_irq(wire_top_ids_inst_timer_inst1_src_top_ids_instgpio_pin0_irq),
. . .
);
wind_vel_logic #( .timer_freq('h2faf080), .wind_constant('h3a980)) wind_vel_logic_inst2(
wind_vel_logic #( .timer_freq('h2faf080), .wind_constant('h3a980)) wind_vel_logic_inst3(
wind_vel_logic #( .timer_freq('h2faf080), .wind_constant('h3a980)) wind_vel_logic_inst1(
ahb2apb #( .addr_width('h20), .bus_width('h20)) ahb2apb_wrapper_top_inst(
endmodule
```

Sequence Specification

- Calling standard reset APIs of IPs from top (chip-level)

sequence name	ip	description		
reset_seq	top.idsng			
arguments	value	description		
reset_val	0			
constants	value	description		
variables	value	description		
assign	value	description		
command	step	value	description	refpath
call	top.gpio.gpio_reset()			
call	top.timer_inst1.timer_reset_mode()			
call	top.timer_inst2.timer_reset_mode()			
call	top.timer_inst3.timer_reset_mode()			
call	top.timer_inst4.timer_reset_mode()			
write	top.pic.intr_cfg	reset_val		
write	top.pic.status	reset_val		
write	top.pic.enable	reset_val		
write	top.pic.post	reset_val		
write	top.pic.priority_reg	reset_val		
write	top.pic.vect_addr	reset_val		
write	top.pic.isr_addr	reset_val		
call	top.i2c.i2c_reset()			
call	top.spi.spi_reset()			
call	top.uart.uart_reset()			

Sequence Specification - Contd..

- Creating top sequence
- Calling reset sub-sequence and configuration APIs of IPs

sequence name	ip	description
top_seq	top.idsng	
arguments	value	description
i2c_slave_addr	0x0000	
i2c_reg_addr	0x0000	
i2c_data_write	0	
pressure_val	0	
wind_val	0	
spi_slave_addr	0x0000	
spi_data_write	0	
spi_trflen	0	
constants	value	description
variables	value	description
vel1	0	
vel2	0	
vel3	0	
vel4	0	
pressure_done	0	
wind_vel1_done	0	
wind_vel2_done	0	
wind_vel3_done	0	
wind_vel4_done	0	

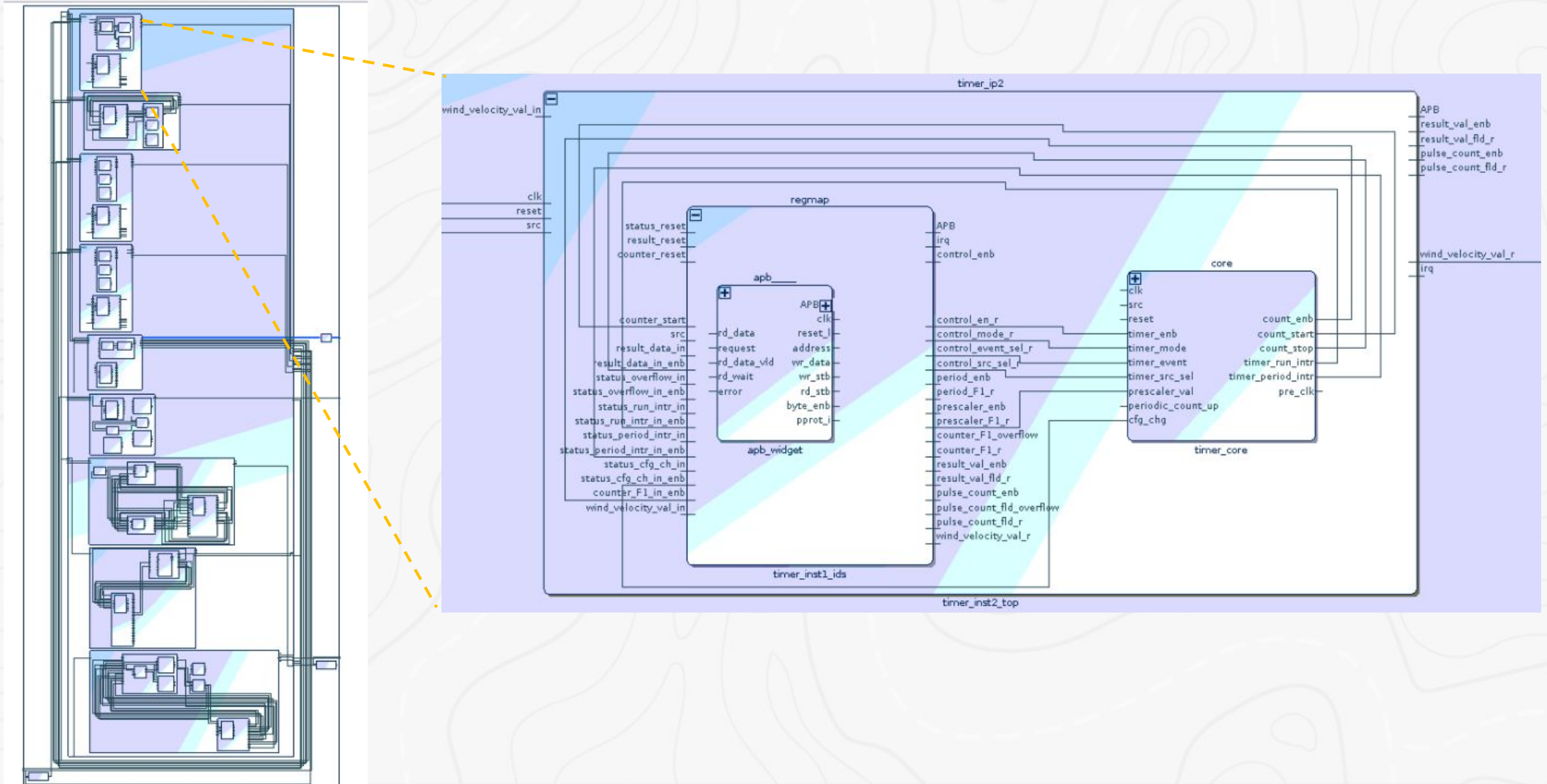
command	step	value	description	refpath
call	reset_seq()			
if(pressure_val){				
call	i2cwrite(i2c_slave_addr, i2c_reg_addr, i2c_data_write, 1)			
pressure_done	1			
}				
if(wind_val){				
for(int i=0; i<5; i++){				
call	top.gpio.gpio_init.out(i,11,1)			
}				
}				
if(top.timer_inst1.control.src_sel){				
call	top.timer_inst1.timer_running_mode(1,0,2)			
vel1	top.timer_inst1.result			
if(vel1!=0){				
wind_vel1_done	1			
}				

if(top.timer_inst3.control.src_sel){				
call	top.timer_inst3.timer_running_mode(1,0,2)			
vel3	top.timer_inst3.result			
if(vel3!=0){				
wind_vel3_done	1			
}				
}				
if(top.timer_inst4.control.src_sel){				
call	top.timer_inst4.timer_running_mode(1,0,2)			
vel4	top.timer_inst4.result			
if(vel4!=0){				
wind_vel4_done	1			
}				
}				
if(pressure_done && wind_vel1_done && wind_vel2_done && wind_vel3_done && wind_vel4_done){				
call	top.spi.spi_write(spi_slave_addr, spi_data_write, spi_trflen)			
}				

List of Files Generated

1. apb_widget.v – 80 lines
2. baud_rate_generator.v – 27 lines
3. clockgen.sv – 76 Lines
4. comp_vec.v – 80 lines
5. comp_vec_last.v – 63 lines
6. edge_detect.v – 26 lines
7. edge_detect_src.v – 30 lines
8. fifo.v – 100 lines
9. gpio.v – 745 lines
10. gpio_detect_sync.v – 62 lines
11. gpio_edge_detect.v – 15 lines
12. gpio_sync_ff.v – 22 lines
13. gpio_top.v – 252 lines
14. i2c.v – 963 lines
15. i2c_block.v – 317 lines
16. i2c_top.v – 218 lines
17. i2cm_byte_transfer.v – 94 lines
18. ids_top_apb_aggregation.v – 459
19. pic.v – 677 lines
20. pic_top.v – 293 lines
21. prescaler.v – 49 lines
22. spi.v – 950 lines
23. spi_core.sv – 522 lines
24. spi_rd_txn.sv – 124 lines
25. spi_wr_txn.sv – 125 lines
26. spi_wrapper.sv – 248 lines
27. sync_ff.v – 24 lines
28. timer_core.v – 214 lines
29. timer_inst1.v – 960 lines
30. timer_inst1_top.v – 175 lines
31. timer_inst2_top.v – 172 lines
32. timer_inst3_top.v – 174 lines
33. timer_inst4_top.v – 174 lines
34. top.sv – 724 lines
35. top.v – 713 lines
36. txn_fifo.sv – 112 lines
37. uart.v – 1138 lines
38. uart_rx.v – 322 lines
39. uart_top.v – 223 lines
40. uart_tx.v – 309 lines
41. ahb2apb.v – 189 lines
42. wrapper_top.sv – 207 lines
43. wrapper_top.v – 215 lines
44. sequence related files - approx 500 lines each UVM and C

Generated Schematic



Generated Top IP-XACT File

```
• wrapper_top_design.xml X
3 <ipxact:design xmlns:ipxact="http://www.spiritconsortium.org/XMLSchema/IPXACT/1685-2014" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4   xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:soce="http://www.agnisys.com/">
5   <ipxact:vendor>Agnisys</ipxact:vendor>
6   <ipxact:library>mixed_signal</ipxact:library>
7   <ipxact:name>wrapper_top_design</ipxact:name>
8   <ipxact:version>1.0</ipxact:version>
9   <ipxact:componentInstances>
10    <ipxact:componentInstance>
11      <ipxact:instanceName>wind_vel_logic_inst4</ipxact:instanceName>
12      <ipxact:componentRef vendor="Agnisys" library="mixed_signal" name="wind_vel_logic" version="1.0"/>
13    </ipxact:componentInstance>
14    <ipxact:componentInstance>
15      <ipxact:instanceName>top_ids_inst</ipxact:instanceName>
16      <ipxact:componentRef vendor="Agnisys" library="mixed_signal" name="top_ids" version="1.0"/>
17    </ipxact:componentInstance>
18    <ipxact:componentInstance>
19      <ipxact:instanceName>wind_vel_logic_inst2</ipxact:instanceName>
20      <ipxact:componentRef vendor="Agnisys" library="mixed_signal" name="wind_vel_logic" version="1.0"/>
21    </ipxact:componentInstance>
22    <ipxact:componentInstance>
23      <ipxact:instanceName>wind_vel_logic_inst3</ipxact:instanceName>
24      <ipxact:componentRef vendor="Agnisys" library="mixed_signal" name="wind_vel_logic" version="1.0"/>
25    </ipxact:componentInstance>
26    <ipxact:componentInstance>
27      <ipxact:instanceName>wind_vel_logic_inst1</ipxact:instanceName>
28      <ipxact:componentRef vendor="Agnisys" library="mixed_signal" name="wind_vel_logic" version="1.0"/>
29    </ipxact:componentInstance>
30    <ipxact:componentInstance>
31      <ipxact:instanceName>ahb2apb_wrapper_top_inst</ipxact:instanceName>
```

Benefits

- Easier handling of complex and large SoC designs through Tcl like scripts and GUI
- On-the-fly generation
 - IPs and subsystems are automatically generated with support for customization and configuration
- Unencrypted code
- Boosting productivity of SoC design teams significantly leading to faster time-to-market for competitive advantage
- Keeps the development costs lower
- Ensures that semiconductor companies meet the stringent time-to-market requirements for competitive advantage
- Reduces SoC design and development cost significantly

Questions