Molding Functional Coverage and Reporting for Highly Configurable IP

Jeremy Ridgeway

Broadcom Limited, Corp.

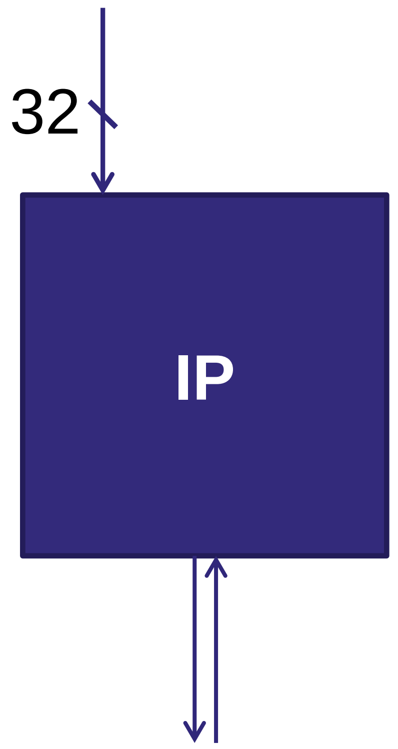
September 29, 2016

SNUG Austin

Agenda

- Background and Problem
- Our Approach
  - Coverpoints are not in a vacuum
  - Mode-of-operation
  - DUT Configuration
- "The Demo" or "What this looks like without actually running a demo"

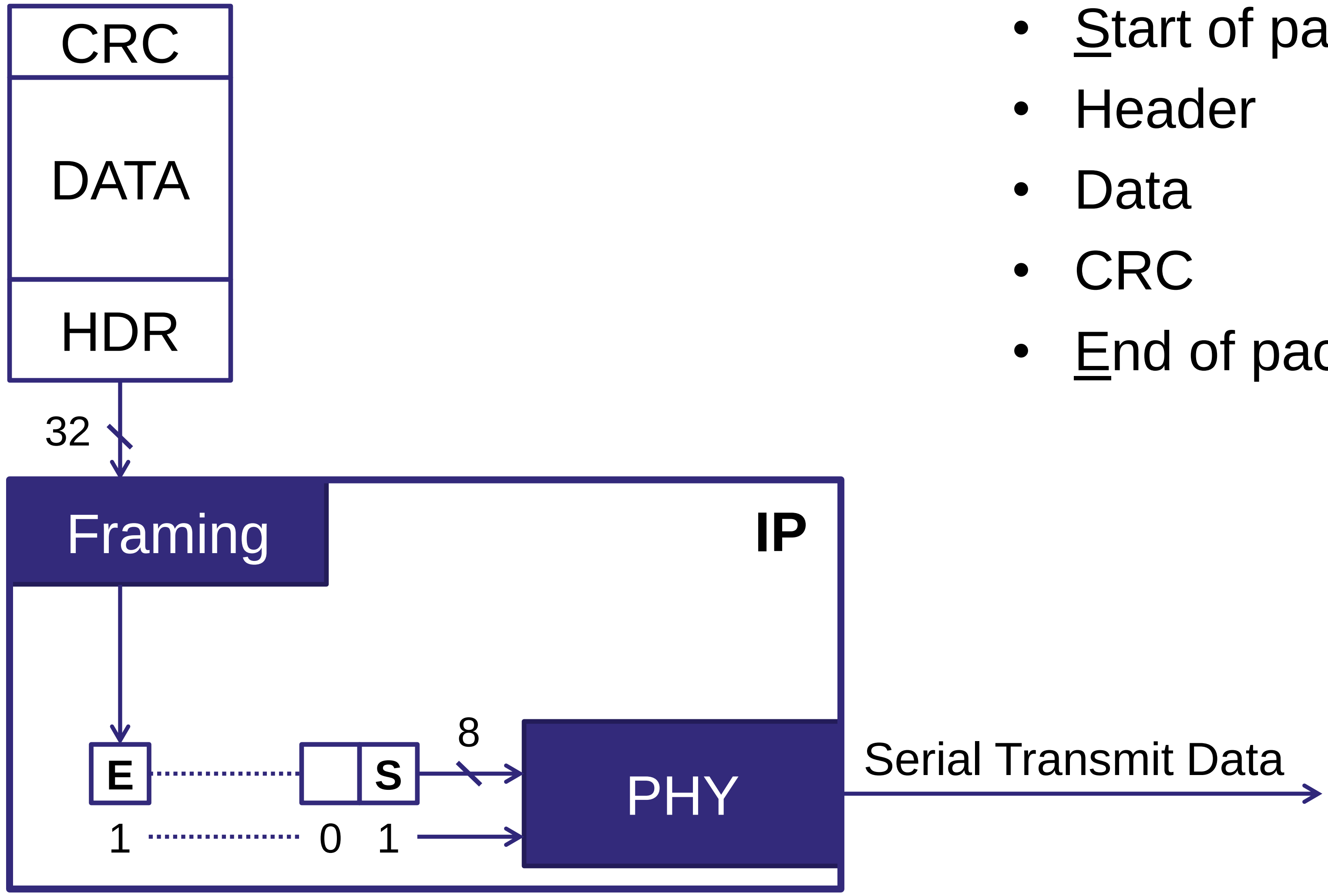
Some Serial Controller IP



- Data path is 32 bits
- Single lane transmit/receive
- Generation 1 speed

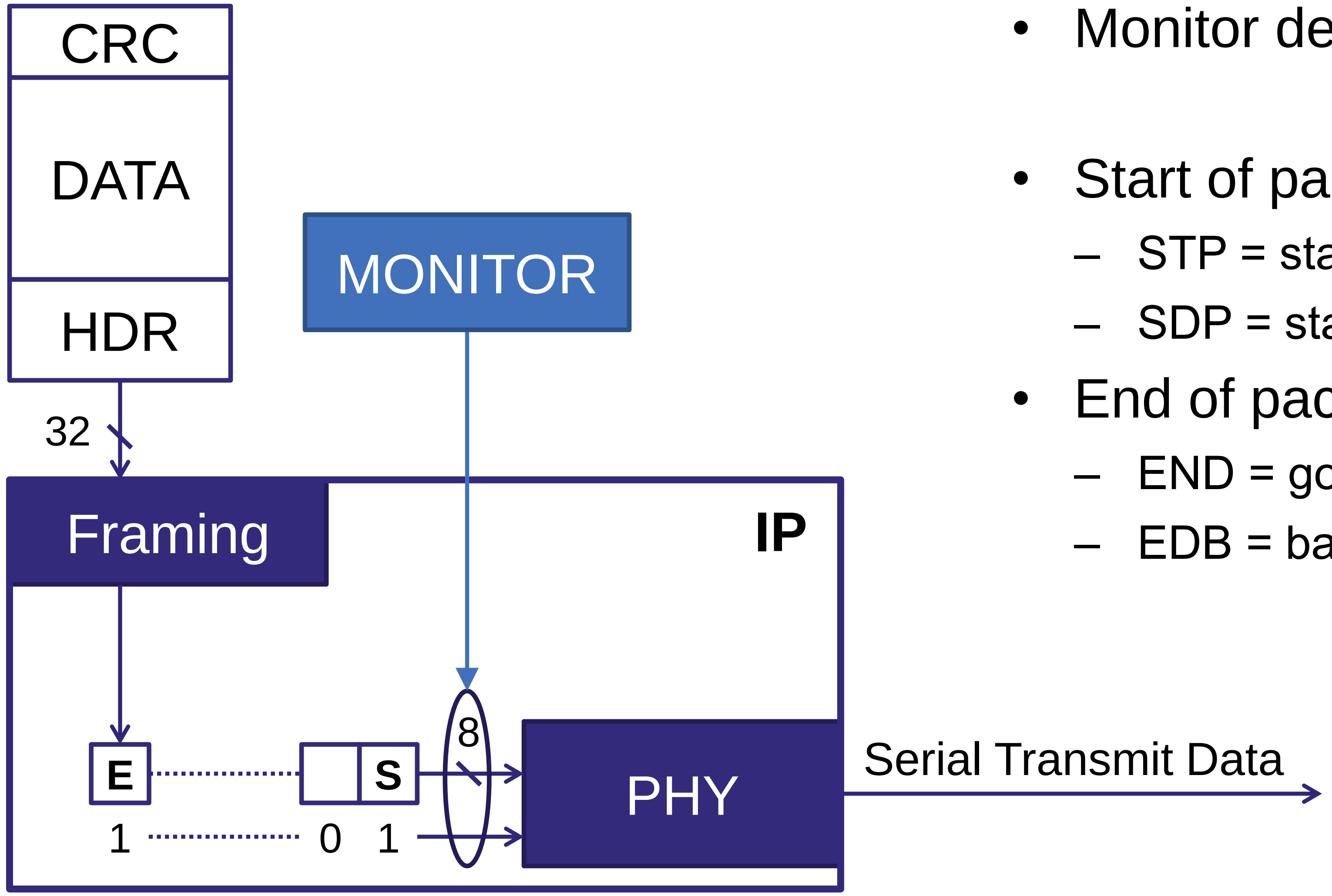
Using PCI-Express as the Example

Packet Data



- Start of packet delimiter
- Header
- Data
- CRC
- End of packet delimiter

Packet Data



- Monitor delimiters
- Start of packet depends on origin
  - STP = start transport packet (8'hFB)
  - SDP = start data link packet (8'h5C)
- End of packet depends on data
  - END = good packet (8'hFD)
  - EDB = bad/nullified packet (8'hFE)

Monitor

```
class MonitorA;
  logic [7:0] data;        // 8-bit data path
  logic ctrl;              // 1-bit control path
  covergroup tx_dp_cg;
    coverpoint data;
    coverpoint ctrl;
  endgroup
endclass
```

- No correlation
  - data == 8'hFE when ctrl == 0
  - False positive for EDB coverage

```
class MonitorB;
  logic [7:0] data;        // 8-bit data path
  logic ctrl;              // 1-bit control path
  covergroup tx_dp_cg;
    coverpoint data;
    coverpoint ctrl;
    cross data, ctrl;      // correlation!
  endgroup
endclass
```

- Over-correlated
  - data == 8'hFE && ctrl == 1
  - EDB covered!
  - False negative for invalid control character

Obviously, not the way to do functional coverage

Monitor

GoldiLocks

```
class MonitorGL;
   logic [7:0] data;          // 8-bit data path
   logic ctrl;                // 1-bit control path
   covergroup tx_dp_cg;
      coverpoint data {
         bins data_0[ ] = {
            8'hFB, 8'hC5C, 8'hFD, 8'hFE };
      }
      coverpoint ctrl {
         bins ctrl_0 = { 1'b1 };
      }
      c_0: cross data, ctrl {
         delim_cross =
            binsof(data.data_0) &&
            binsof(ctrl_0);
      }
   endgroup
endclass
```

- Properly correlated covergroup
- When cross c_0 is covered, the whole group is covered

Only care about delimiter encodings

Only when characters are marked as control

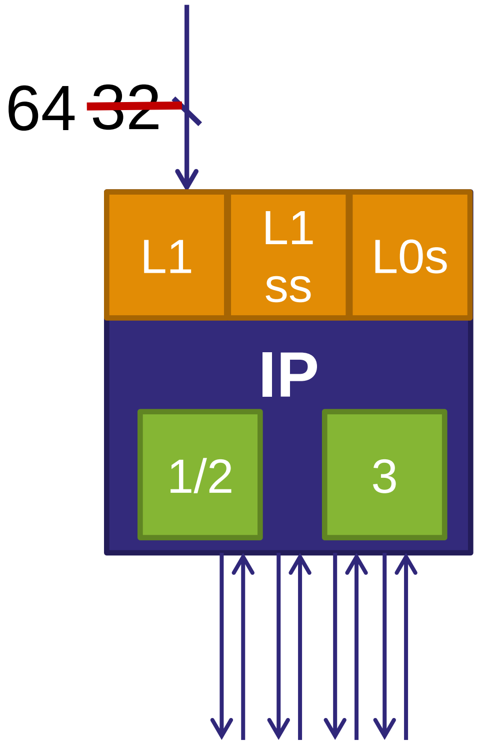
Sample only packet delimiter control characters

General

PCIe Controller IP

= $$ for company

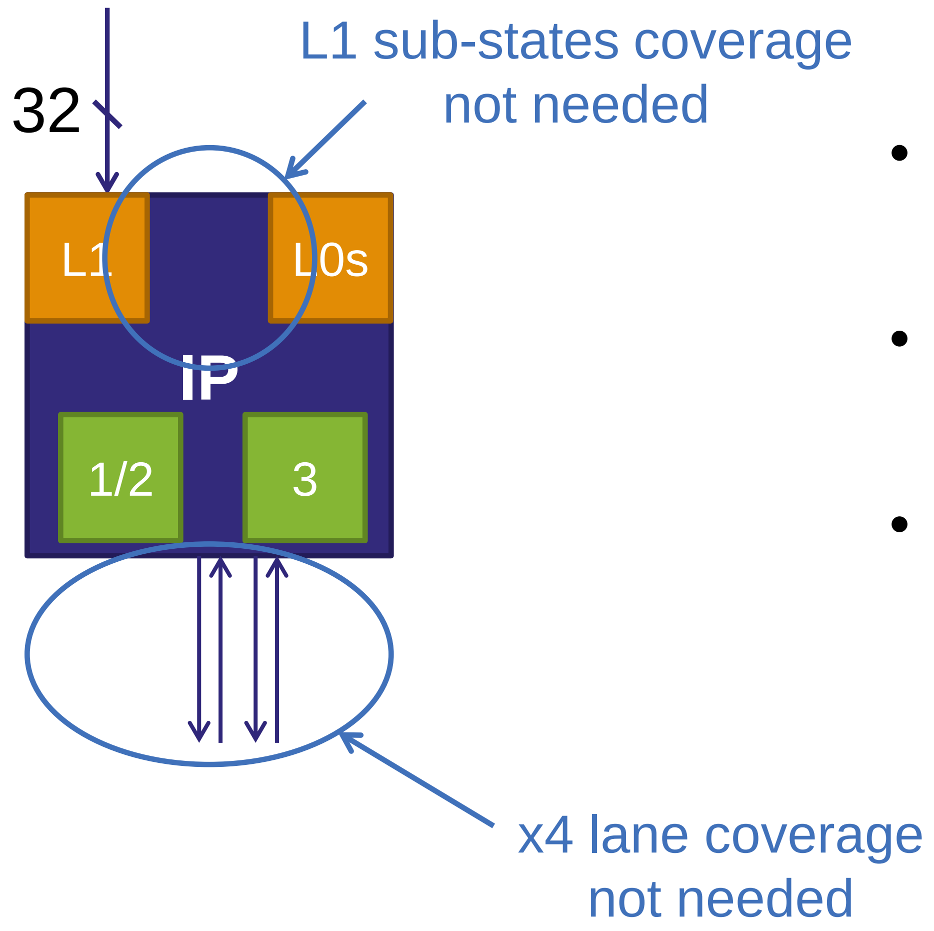
= Job for me



- Data path is ~~32~~ 64 bits
- ~~Single~~ Four lane transmit/receive ~~Two~~
- Generation ~~1~~ ~~2~~ 3 speed
- Low power support ✓
  - L0s ✓
  - L1 ✓
  - L1 Sub-states ✓

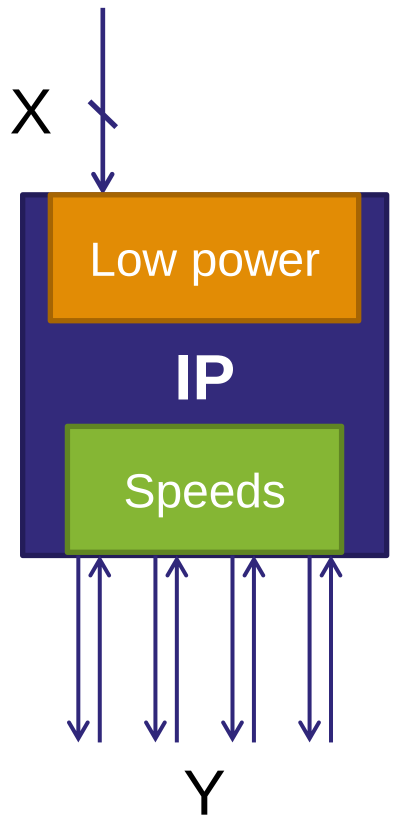
PCIe Controller IP

Gen3, 2-Lane, Low Power L0s and L1 support


L1 sub-states coverage not needed
32
L1
L0s
IP
1/2
3
x4 lane coverage not needed

- Data path is 32 bits
- Two lane transmit/receive
- Generation 3 speed
- Low power support
  - L0s
  - L1

General PCIe Controller IP



- Data path is X bits

Config – Fixed value

- Y number of lanes transmit/receive

Config – Supported at compile-time

Mode – Enabled at runtime

- Generation Z speed

Config – Supported at compile-time

Mode – Enabled at runtime

- Low power options include/exclude

Config – Supported at compile-time

Mode – Enabled at runtime

Monitor

```
class MonitorGL;
    logic [7:0] data;           // 8-bit data path
    logic ctrl;                 // 1-bit control path
    covergroup tx_dp_cg;
        coverpoint data {
            bins data_0[ ] = {
                8'hFB, 8'h5C, 8'hFD, 8'hFE };
        }
        coverpoint ctrl {
            bins ctrl_0 = { 1'b1 };
        }
        c_0: cross data, ctrl {
            delim_cross =
                binsof(data.data_0) &&
                binsof(ctrl_0);
        }
    endgroup
endclass
```

- Are all delimiters covered when:
  - GEN1, GEN2, GEN3 speed?
  - x1 support: x1 enabled?
  - x2 support: x1 enabled, x2 enabled?
  - x4 support: x1, x2, x4 enabled?
  - Lanes reversed?
  - Lane polarity reversed?
  - After transition out of L0s?
  - After transition out of L1?
  - After transition out of an L1 sub-state?

Monitor

Customer 2:
Did you even test this mode?

class
lo
logi
covergroup tx_dp_cg;

t ctrl
_0 =

data
oss =
f(data
f(ctrl_

e

ers covered when:

GEN1, GEN2, GEN3 speed?

x1 support; x1 enabled?

The coverage scenario questions are applicable to ALL covergroups

– Lanes reversed?
– Lane polarity reversed?
– After transition out of L0s?
– After transition out of L1?

Shape group based on config

Provide cross context via mode of operation

Customer 1:
I see an error when … !

Agenda

Background

Our Approach

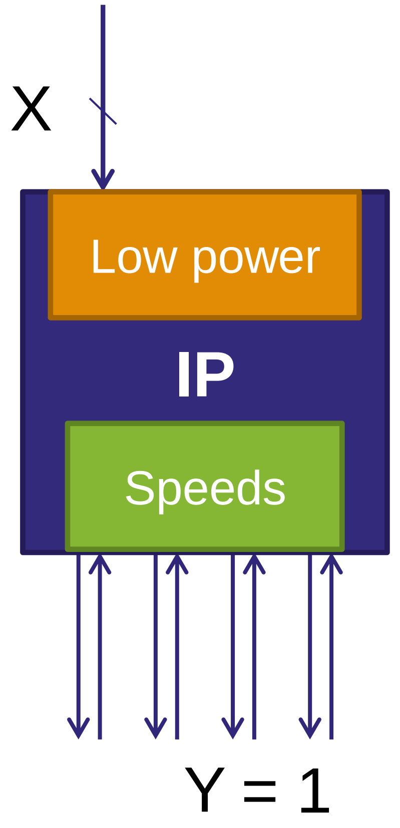
- Coverpoints are not in a vacuum
- Mode-of-operation
- DUT Configuration

"The Demo" or "What this looks like without actually running a demo"

Customer 2: Did you even test this mode?

- Mode-of-operation
  - Selected once or a few times during simulation
  - Often only at the beginning of simulation
  - Affects coverage context
- Configuration
  - Selected via `defines throughout the RTL
  - Affects structure of covergroup

cross Gen3 && x1 && …
~~**cross** Gen3 && x2 && …~~
~~**cross** Gen3 && x4 && …~~



Goals

- Write covergroups once
- Automatically shape all covergroups by configuration
- Automatically propagate mode coverpoints to covergroups
- Avoid waivers at (nearly) all costs

When cross c_0 is covered,
the whole group is covered

Table Format

Coverpoints	Data	Ctrl
c_0	8'hFB, 8'h5C 8'hFD, 8'hFE	1'b1

```
cross data, ctrl {
  binsof(data.data_0) &&
  binsof(ctrl.ctrl_0);
}
```

```
coverpoint data {
  bins data_0[ ] = {
    8'hFB, 8'h5C, 8'hFD, 8'hFE };
}
```

```
coverpoint ctrl { bins ctrl_0 = { 1'b1 }; }
```

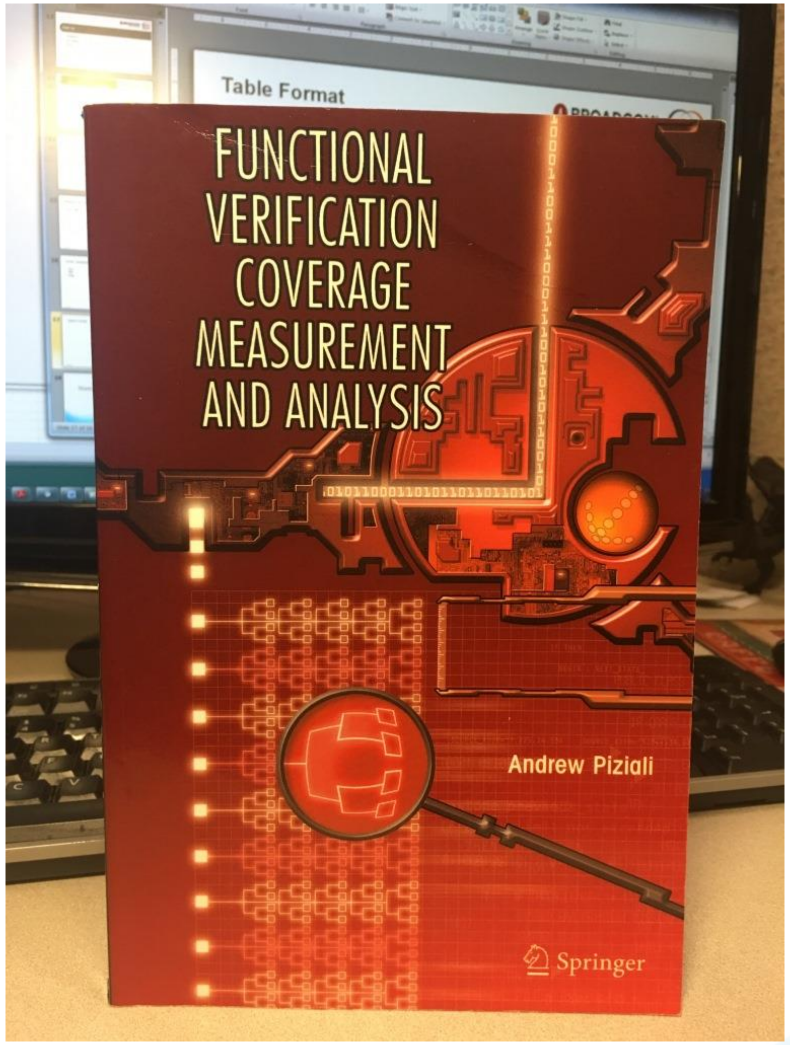
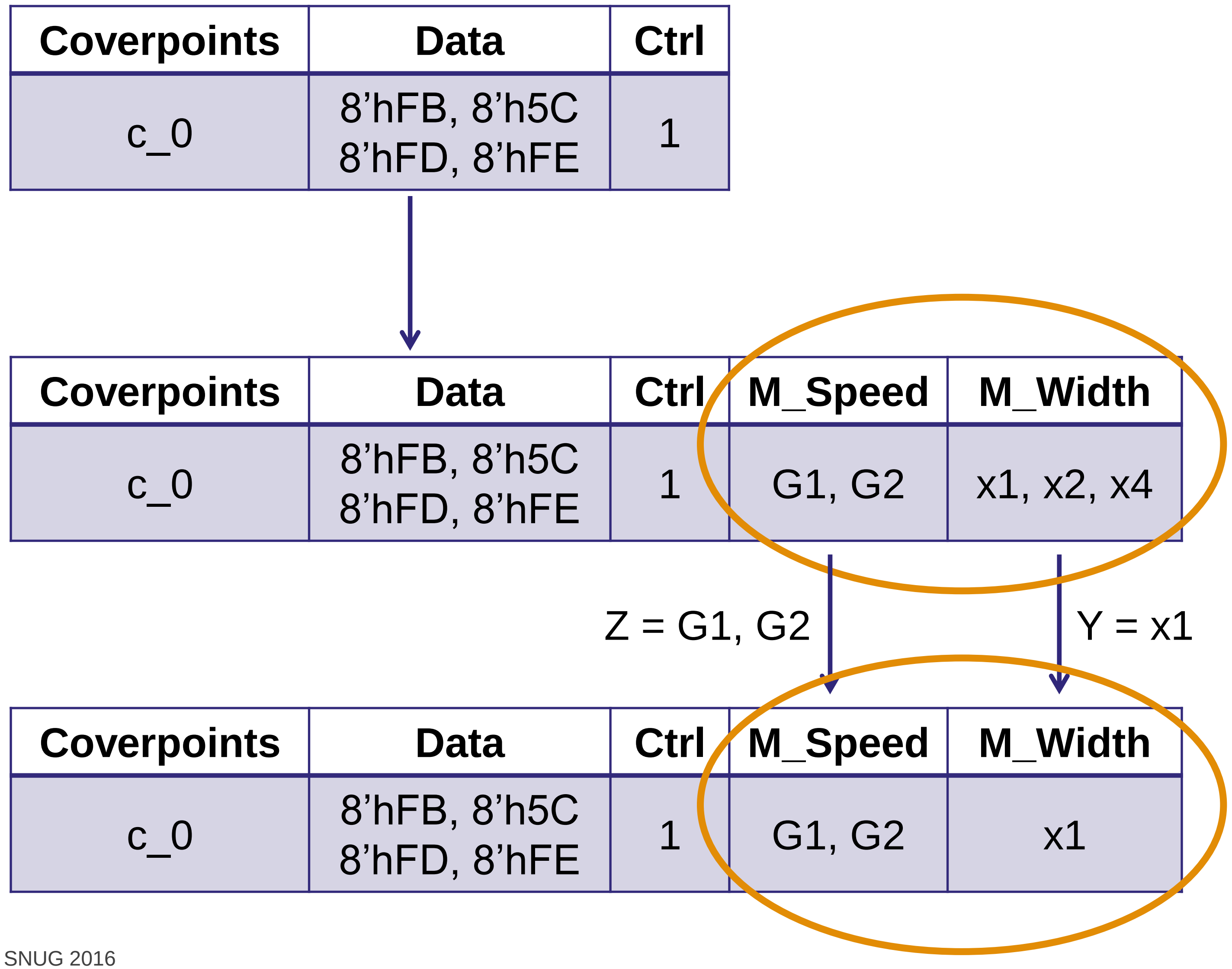


Table Format



Coverpoints	Data	Ctrl
c_0	8’hFB, 8’h5C 8’hFD, 8’hFE	1

Coverpoints	Data	Ctrl	M_Speed	M_Width
c_0	8’hFB, 8’h5C 8’hFD, 8’hFE	1	G1, G2	x1, x2, x4

Coverpoints	Data	Ctrl	M_Speed	M_Width
c_0	8’hFB, 8’h5C 8’hFD, 8’hFE	1	G1, G2	x1

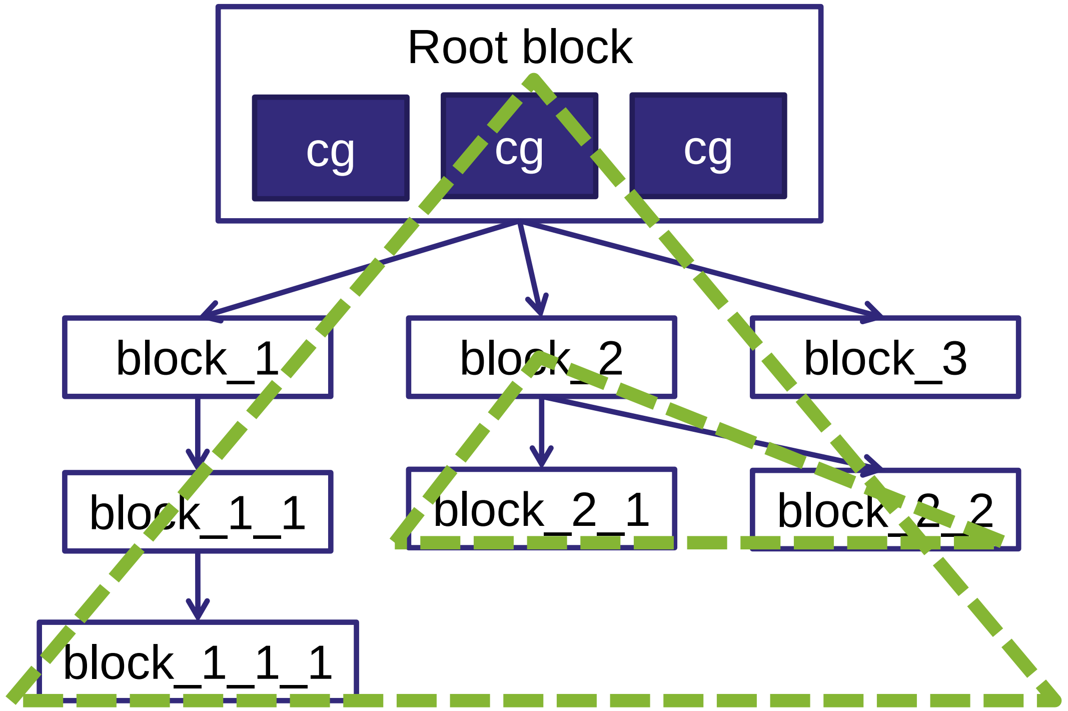
- Write covergroup once
- Automatic mode propagation
- Shape based on configuration

How to setup mode propagation?

How filter based on configuration?

Hierarchical Abstract Model

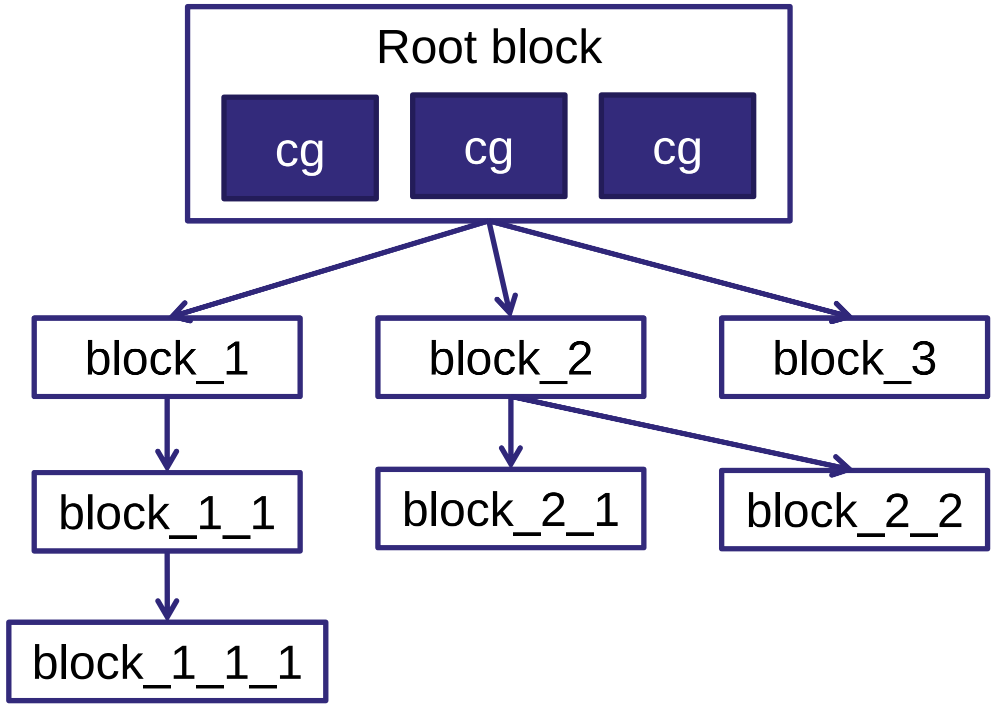
- Cover block
  - Collection of related cover groups and cover blocks
- Block model
  - Sub-tree from any block
- Cover model
  - Sub-tree from the root block



Hierarchical Abstract Model

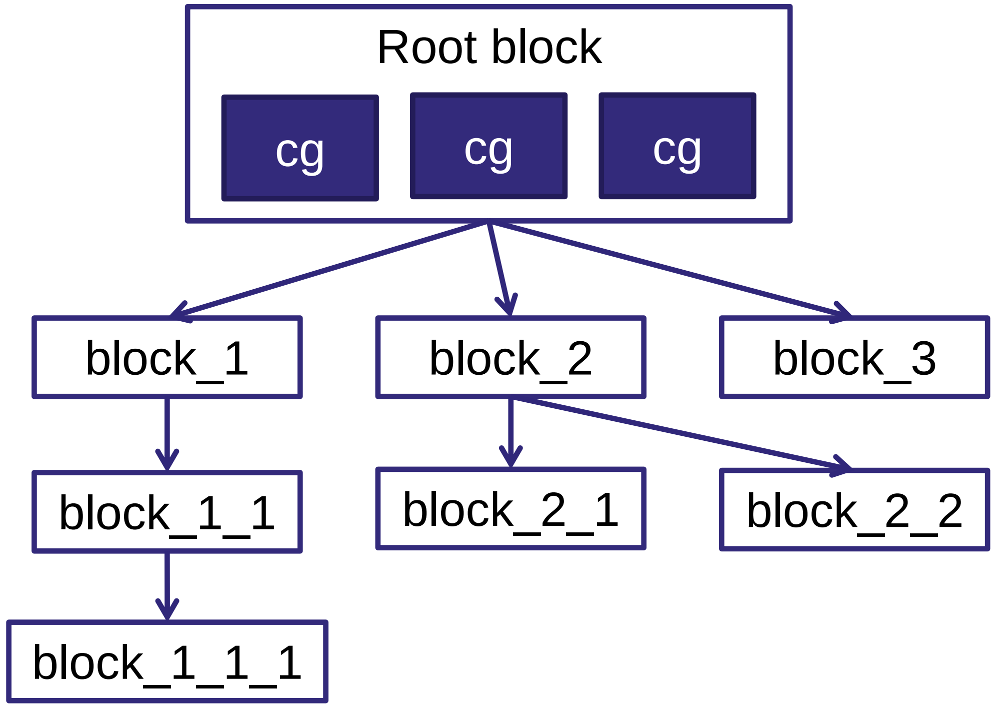
- Cover block
  - Collection of related cover groups and cover blocks
- Block model
  - Sub-tree from any block
- Cover model
  - Sub-tree from the root block

- Cover block – Excel Spreadsheet
- Cover model – Directory structure

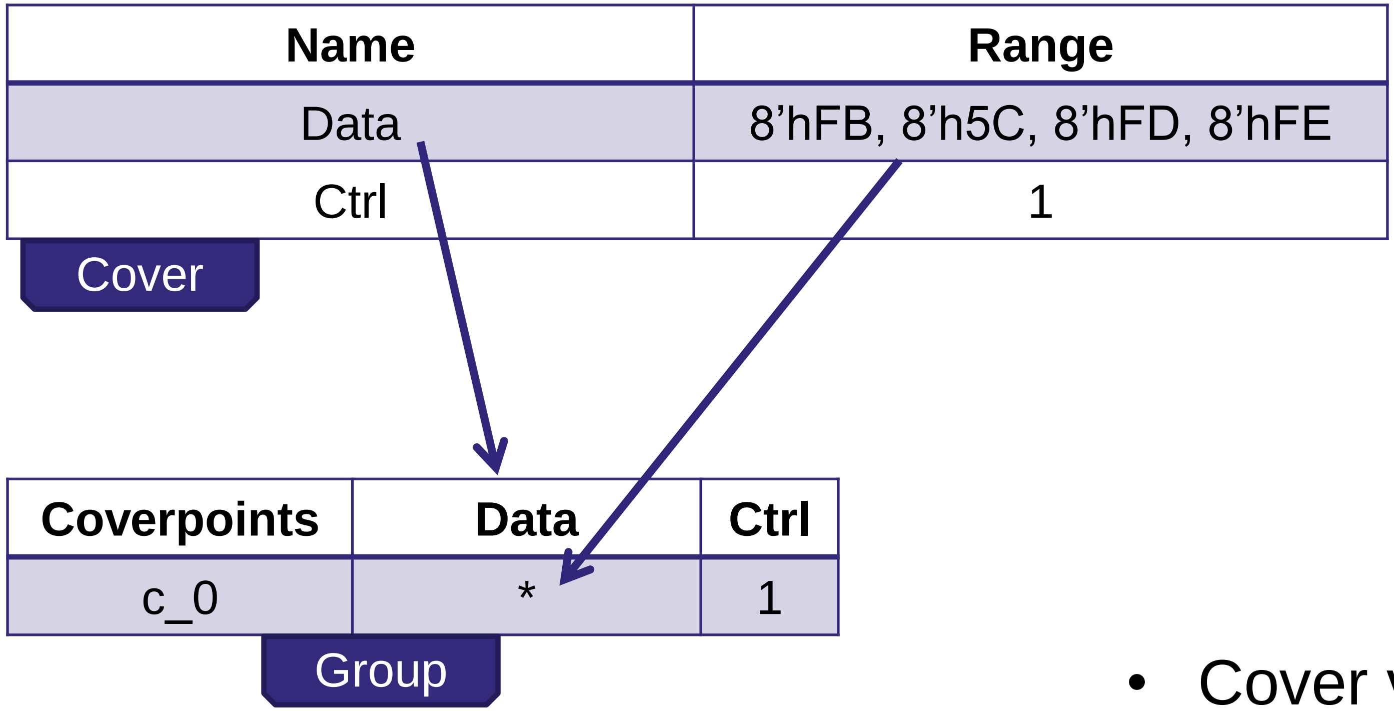


Cover Variables

- Cover
  - encourage reuse
- Mode
  - special case cover variable automatically propagated to all covergroups in the current coverblock and child coverblocks
- Config
  - special case cover variable used to filter mode variable values AND cover group scenarios



Cover variables in block_1



Name	Range
Data	8’hFB, 8’h5C, 8’hFD, 8’hFE
Ctrl	1

Cover

Coverpoints	Data	Ctrl
c_0	*	1

Group

root/block_1/Cover.xlsx

- Cover variables may be re-used:
  - In current spreadsheet
  - In any child block’s spreadsheet

Config and Mode variables in Root

Name	Range
C_Speed	G1, G2, G3
C_Width	x1, x2, x4

Config

Name	Range
M_Speed	G1, G2, G3
M_Width	x1, x2, x4

Mode

root/Cover.xlsx

- Mode variables are propagated to:
  - Current spreadsheet cover groups
  - All child blocks' spreadsheet cover groups

Specific configuration

Supported speeds: G1, G2; Support width: x1

Name	Range
C_Speed	G1, G2, G3
C_Width	x1, x2, x4

Config

Name	Range
M_Speed	G1, G2, G3
M_Width	x1, x2, x4

Mode

root/Cover.xlsx

- Config variables filter:
  - All mode variables
  - Any cover group scenario that specifies it

Specific configuration

Supported speeds: G1, G2; Support width: x1

Name	Range
C_Speed	**G1, G2**
C_Width	**x1**

Config

Name	Range
M_Speed	G1, G2, G3
M_Width	x1, x2, x4

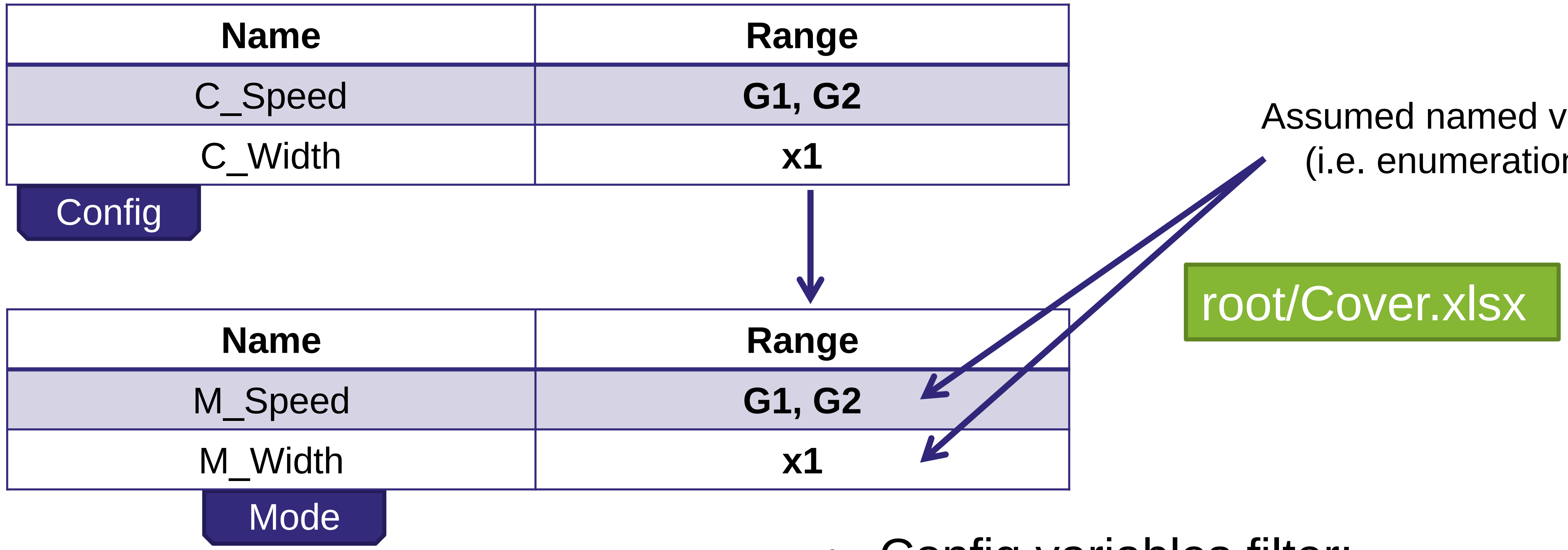
Mode

root/Cover.xlsx

- Config variables filter:
  - All mode variables
  - Any cover group scenario that specifies it

Specific configuration

Supported speeds: G1, G2; Support width: x1



Name	Range
C_Speed	**G1, G2**
C_Width	**x1**

Config

Name	Range
M_Speed	**G1, G2**
M_Width	**x1**

Mode

- Config variables filter:
  - All mode variables
  - Any cover group scenario that specifies it

Cover variables in block_1

Name	Range
C_Speed	G3
C_Width	x1

Config

root/Cover.xlsx

Coverpoints	Data	Ctrl	C_Speed
~~c_0~~	~~*~~	~~1~~	~~G1, G2~~

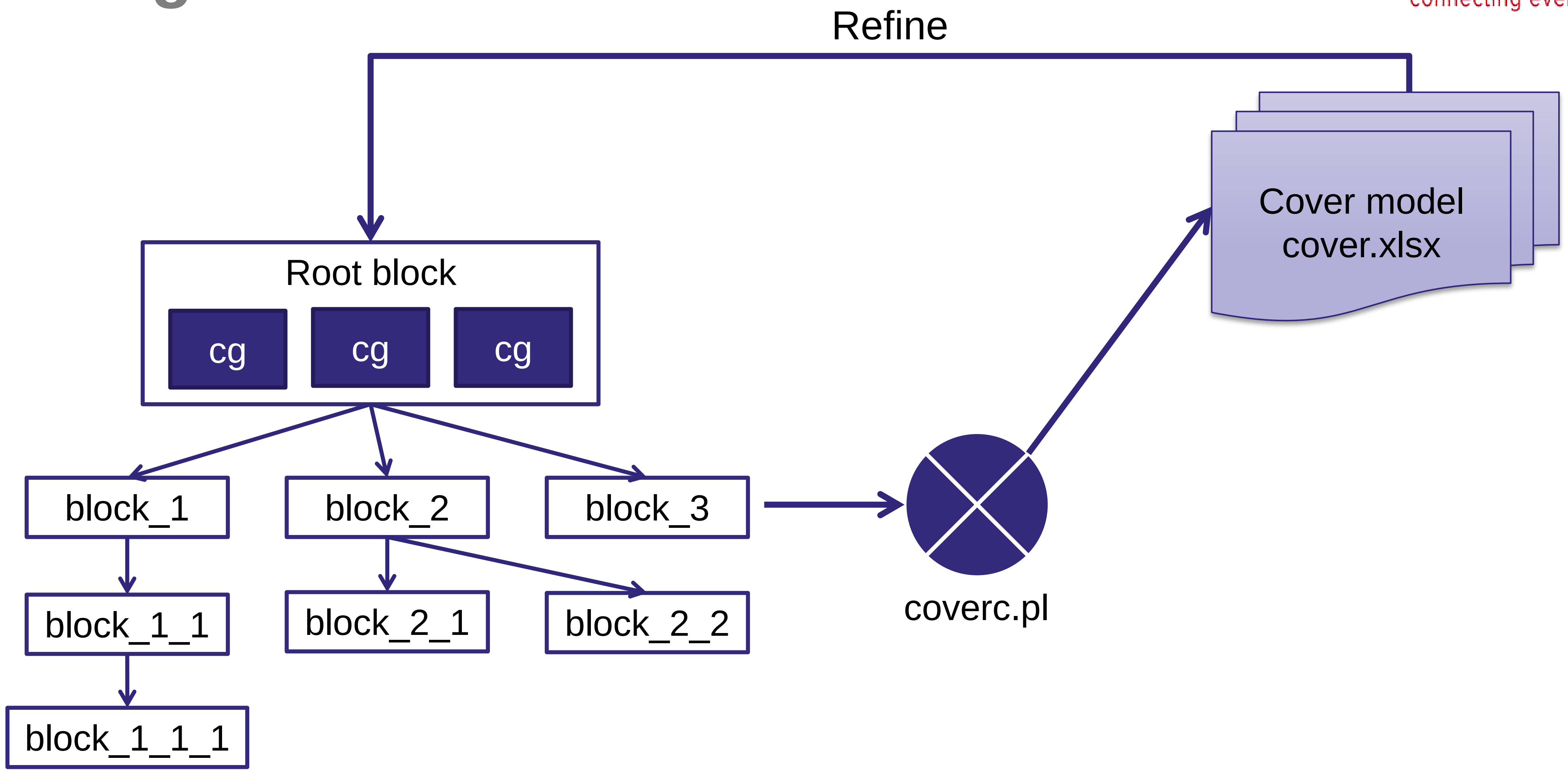
Group

Not applicable
Removed

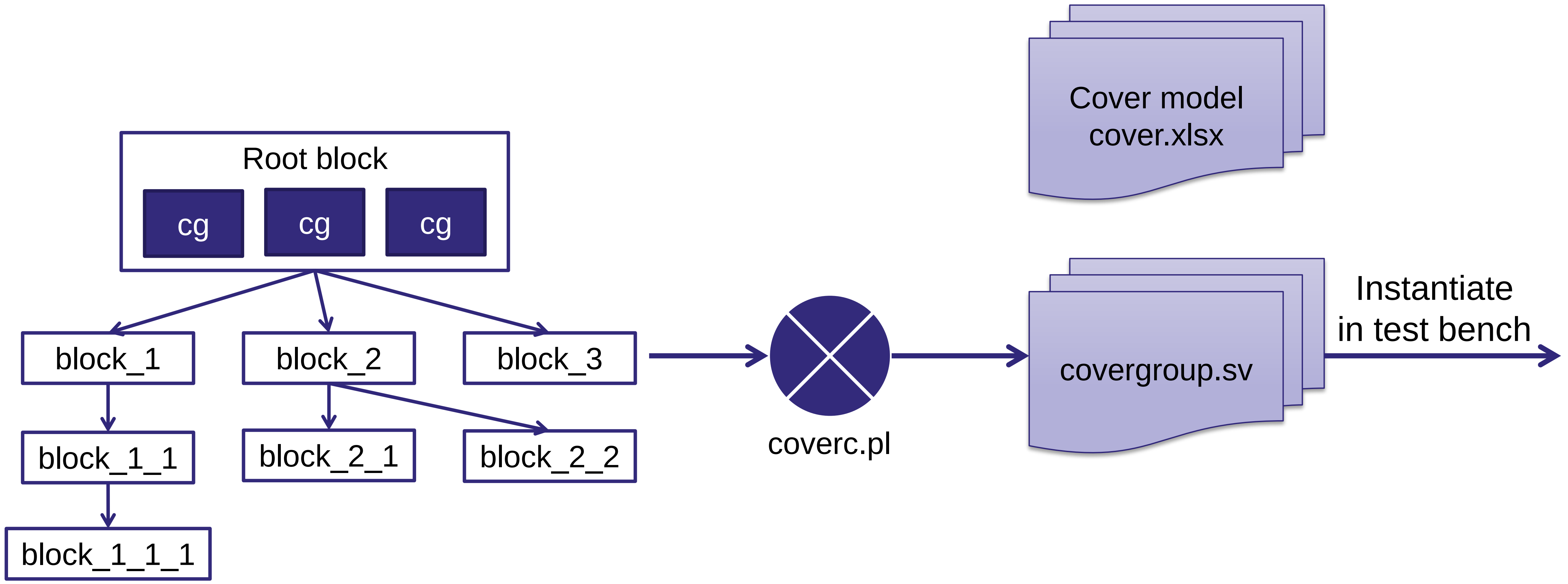
root/block_1/Cover.xlsx

- Config variables filter:
  - Any cover group scenario that specifies it

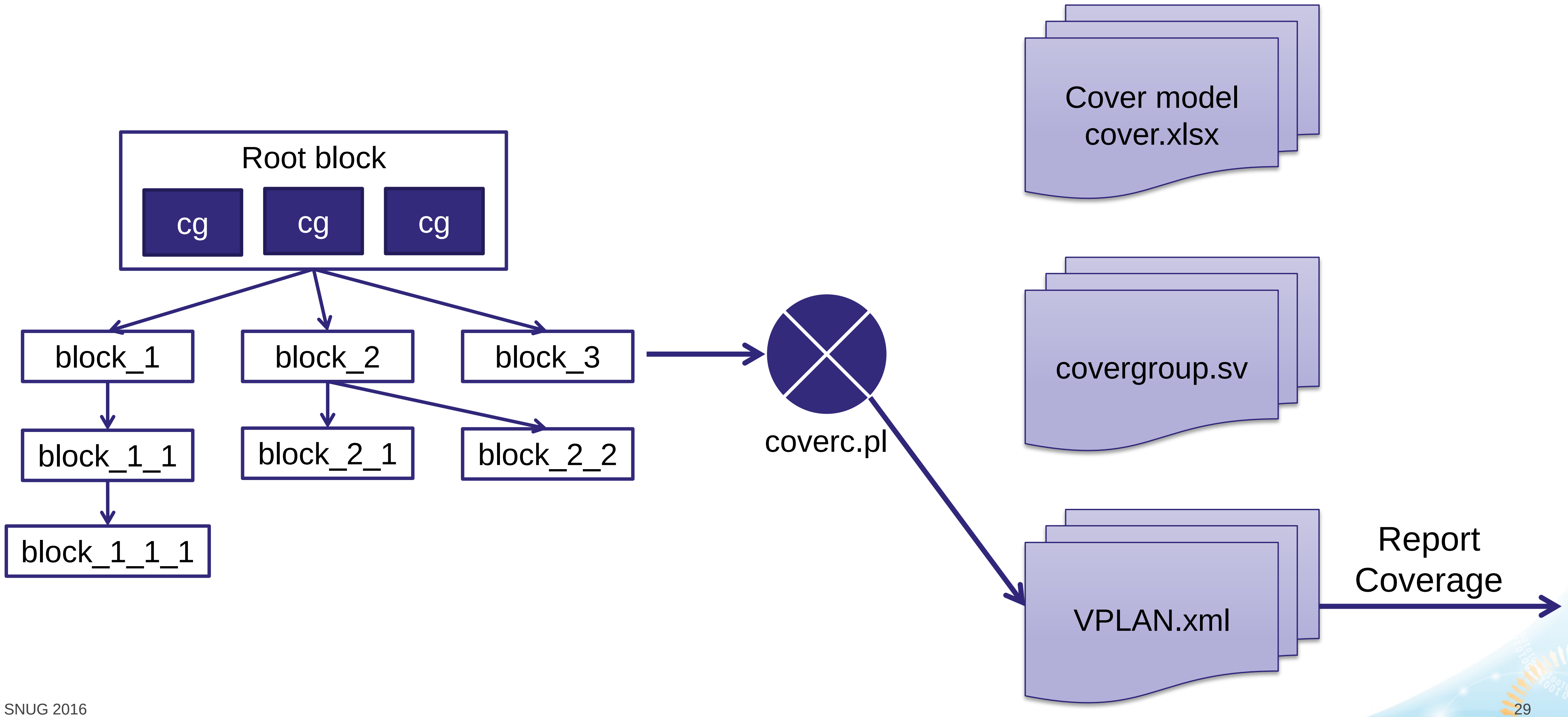
Usage model


Refine
Cover model
cover.xlsx
Root block
cg
cg
cg
block_1
block_2
block_3
block_1_1
block_2_1
block_2_2
block_1_1_1
coverc.pl

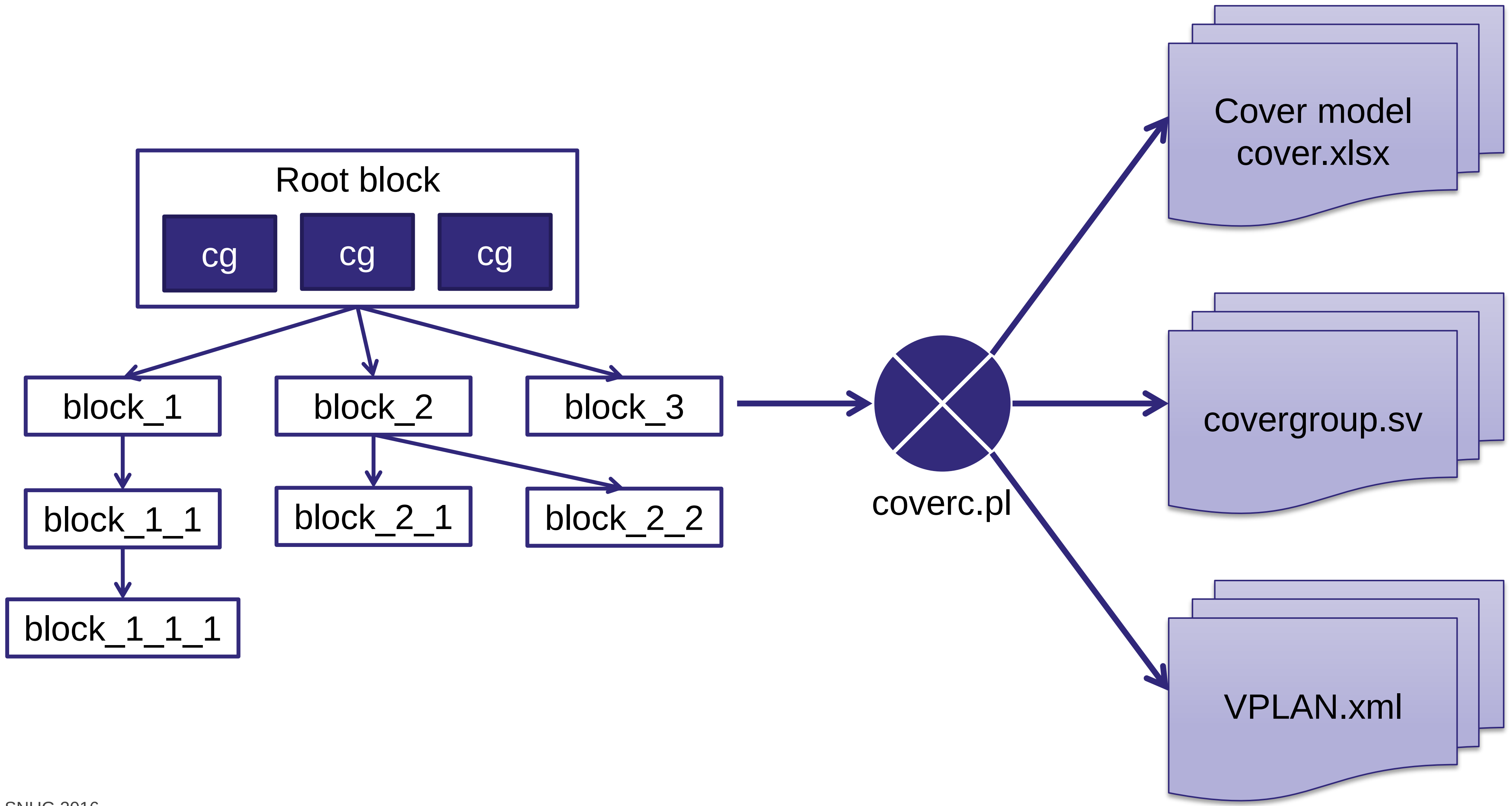
Usage model


Root block
cg
cg
cg
block_1
block_2
block_3
block_1_1
block_2_1
block_2_2
block_1_1_1
coverc.pl
Cover model
cover.xlsx
covergroup.sv
Instantiate
in test bench

Usage model


Root block
cg
cg
cg
block_1
block_2
block_3
block_1_1
block_2_1
block_2_2
block_1_1_1
coverc.pl
Cover model
cover.xlsx
covergroup.sv
VPLAN.xml
Report
Coverage

Usage model


Root block
cg
cg
cg
block_1
block_2
block_3
block_1_1
block_2_1
block_2_2
block_1_1_1
coverc.pl
Cover model
cover.xlsx
covergroup.sv
VPLAN.xml

Agenda

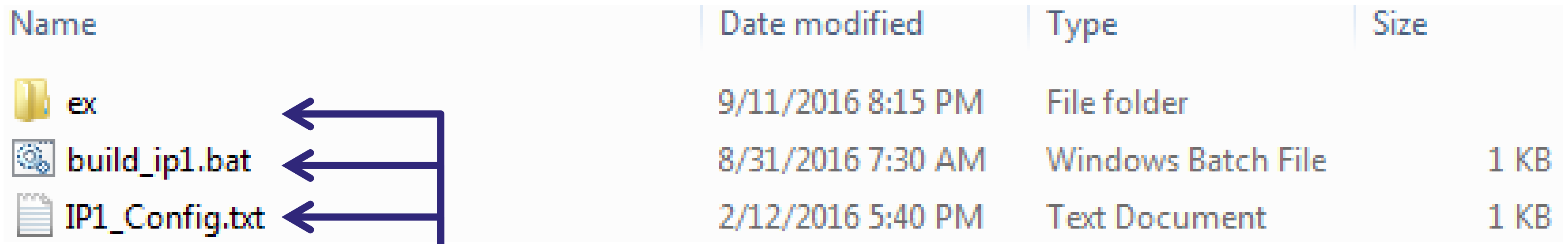
Background

Our Approach

"The Demo" or "What this looks like without actually running a demo"

- Coversheet examples from the paper

Root Cover Model Directory



Name	Date modified	Type	Size
ex	9/11/2016 8:15 PM	File folder	
build_ip1.bat	8/31/2016 7:30 AM	Windows Batch File	1 KB
IP1_Config.txt	2/12/2016 5:40 PM	Text Document	1 KB

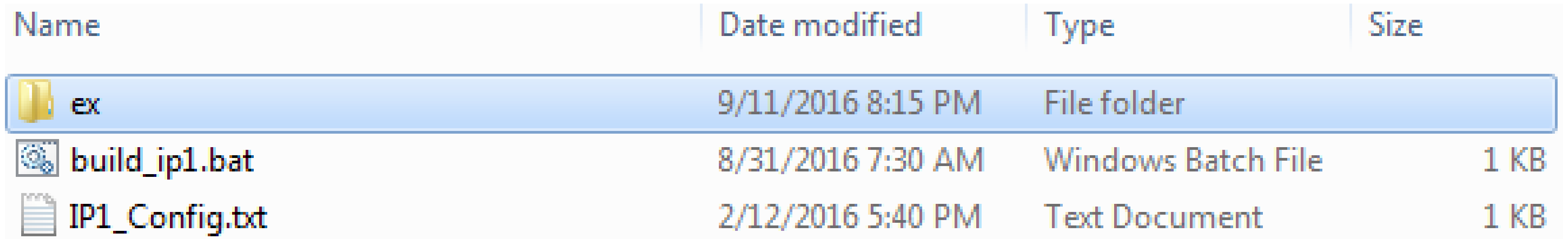
Contains the coversheets from the paper

IP1 static configuration

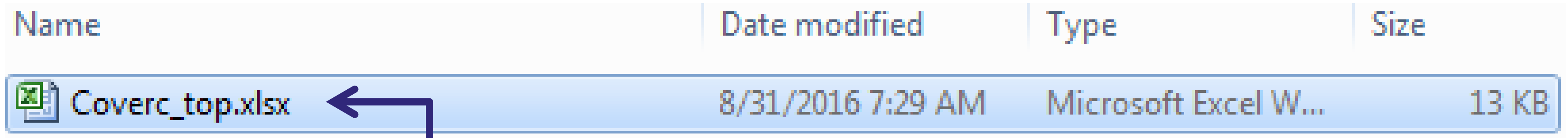
Script to build cover model for IP1

It's PERL, it can be run on Windows, too!

Root Cover Block



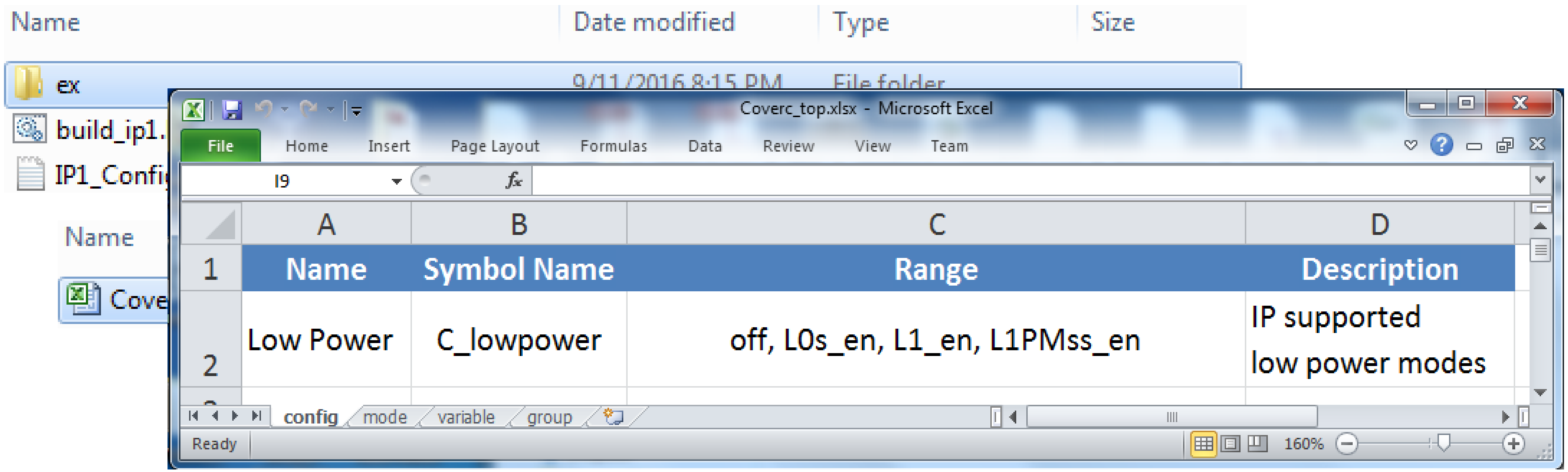
Name	Date modified	Type	Size
ex	9/11/2016 8:15 PM	File folder	
build_ip1.bat	8/31/2016 7:30 AM	Windows Batch File	1 KB
IP1_Config.txt	2/12/2016 5:40 PM	Text Document	1 KB



Name	Date modified	Type	Size
Coverc_top.xlsx	8/31/2016 7:29 AM	Microsoft Excel W...	13 KB

Root block coversheet

Root Block::Config Tab



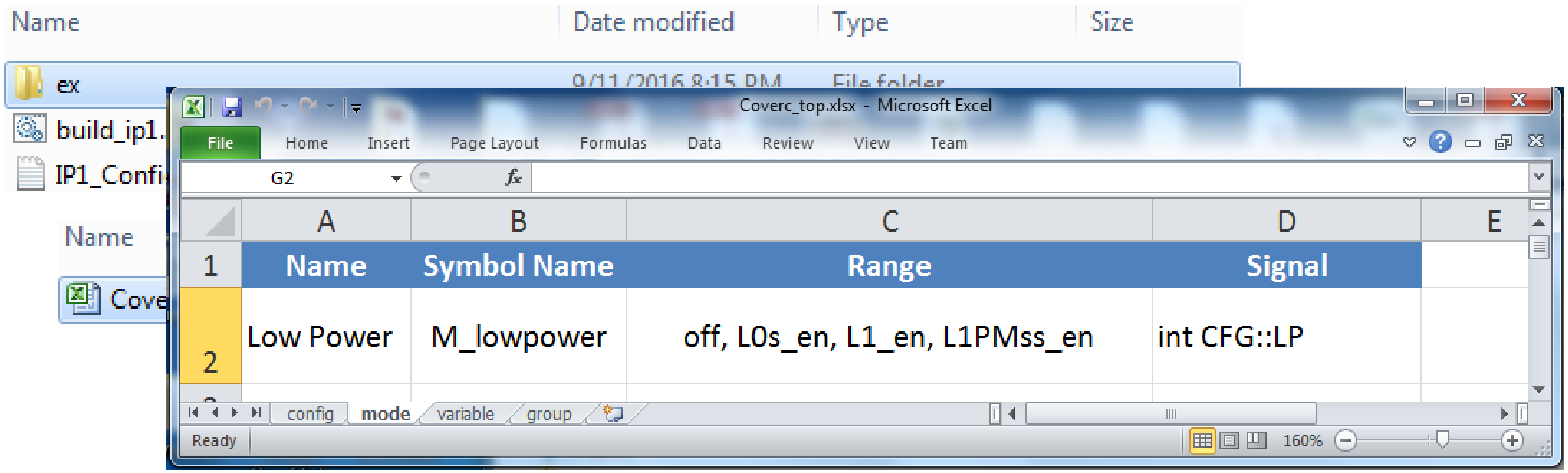
Name	Date modified	Type	Size
ex	9/11/2016 8:15 PM	File folder	
build_ip1.			
IP1_Confi			

Coverc_top.xlsx - Microsoft Excel

Name	Symbol Name	Range	Description
Low Power	C_lowpower	off, L0s_en, L1_en, L1PMss_en	IP supported low power modes

- config tab – lists configuration variables
- Identifies static (compile-time) configuration values for the DUT

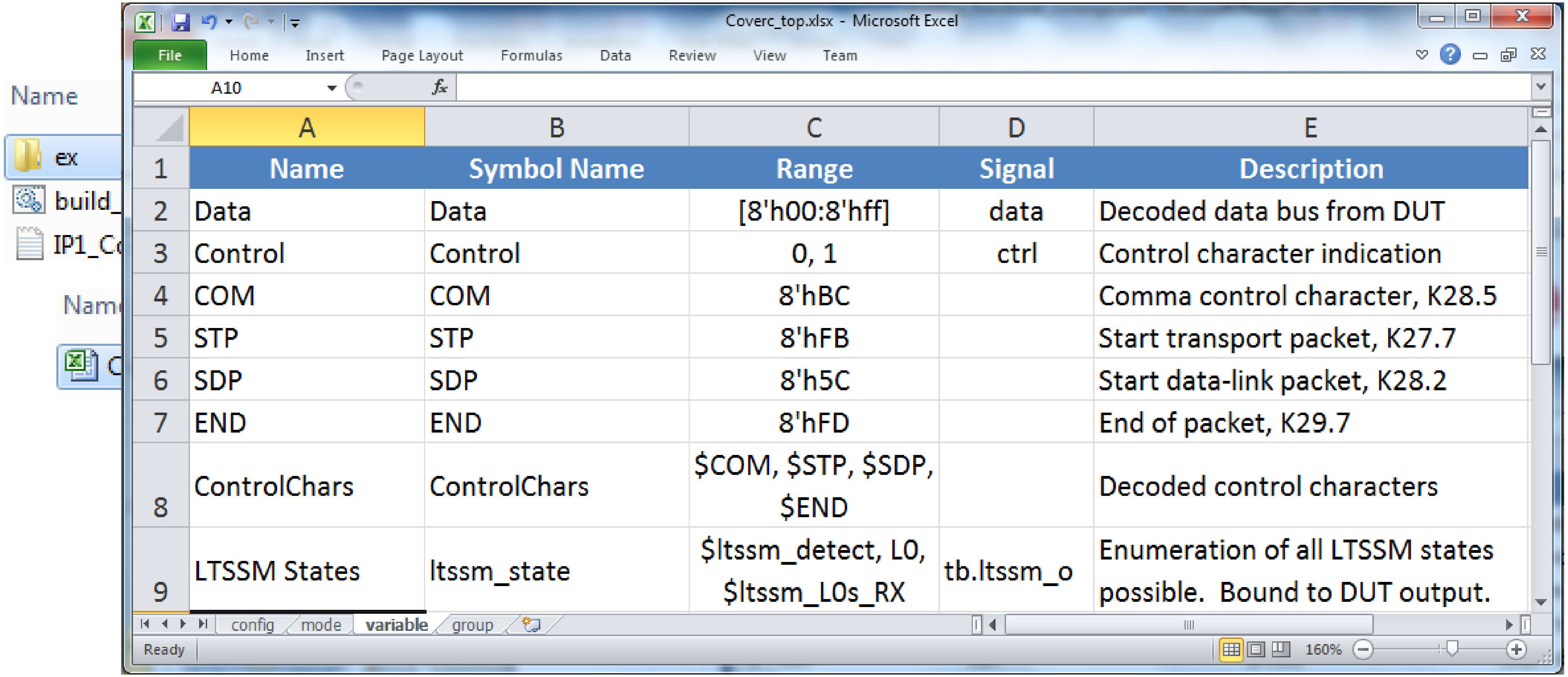
Root Block::Mode Tab



Name	Symbol Name	Range	Signal
Low Power	M_lowpower	off, L0s_en, L1_en, L1PMss_en	int CFG::LP

- mode tab – lists mode variables
- Mode variable range values are pass-thru filtered by config variables

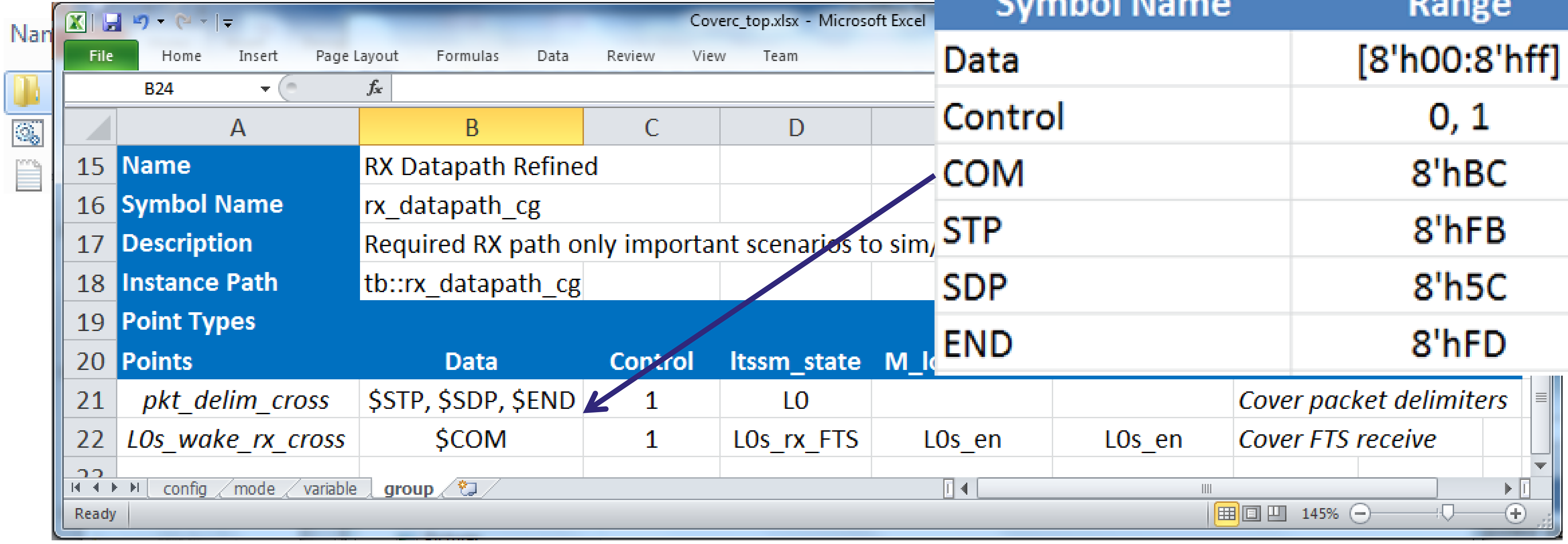
Root Block::Variable Tab



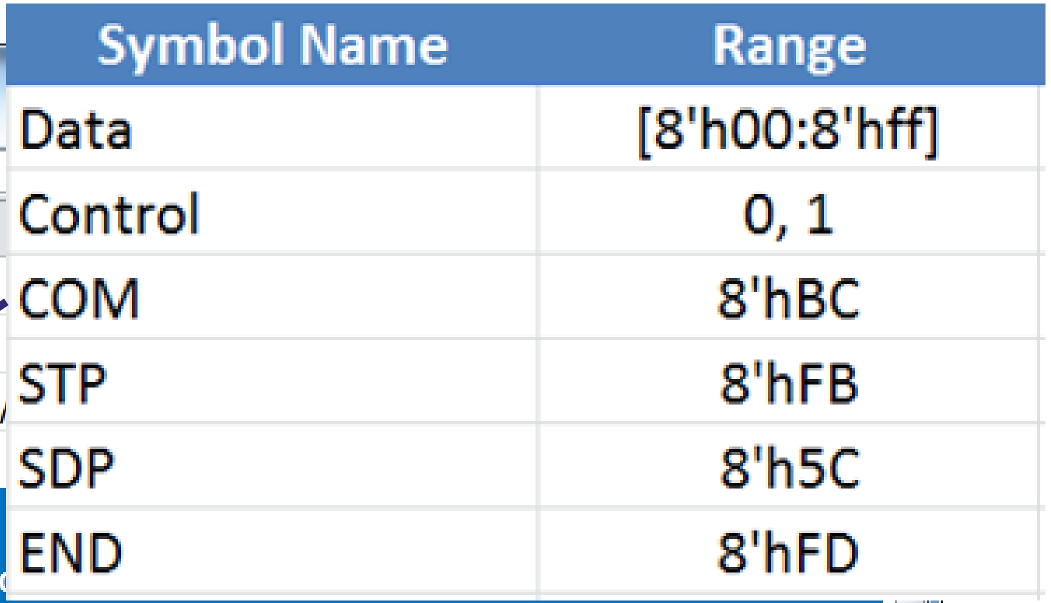
	A	B	C	D	E
1	Name	Symbol Name	Range	Signal	Description
2	Data	Data	[8'h00:8'hff]	data	Decoded data bus from DUT
3	Control	Control	0, 1	ctrl	Control character indication
4	COM	COM	8'hBC		Comma control character, K28.5
5	STP	STP	8'hFB		Start transport packet, K27.7
6	SDP	SDP	8'h5C		Start data-link packet, K28.2
7	END	END	8'hFD		End of packet, K29.7
8	ControlChars	ControlChars	$COM, $STP, $SDP, $END		Decoded control characters
9	LTSSM States	ltssm_state	$ltssm_detect, L0, $ltssm_L0s_RX	tb.ltssm_o	Enumeration of all LTSSM states possible. Bound to DUT output.

- variable – declare ranges for use in cover points
- Variables may be used in this coversheet and any child coversheet

Root Block::Group Tab



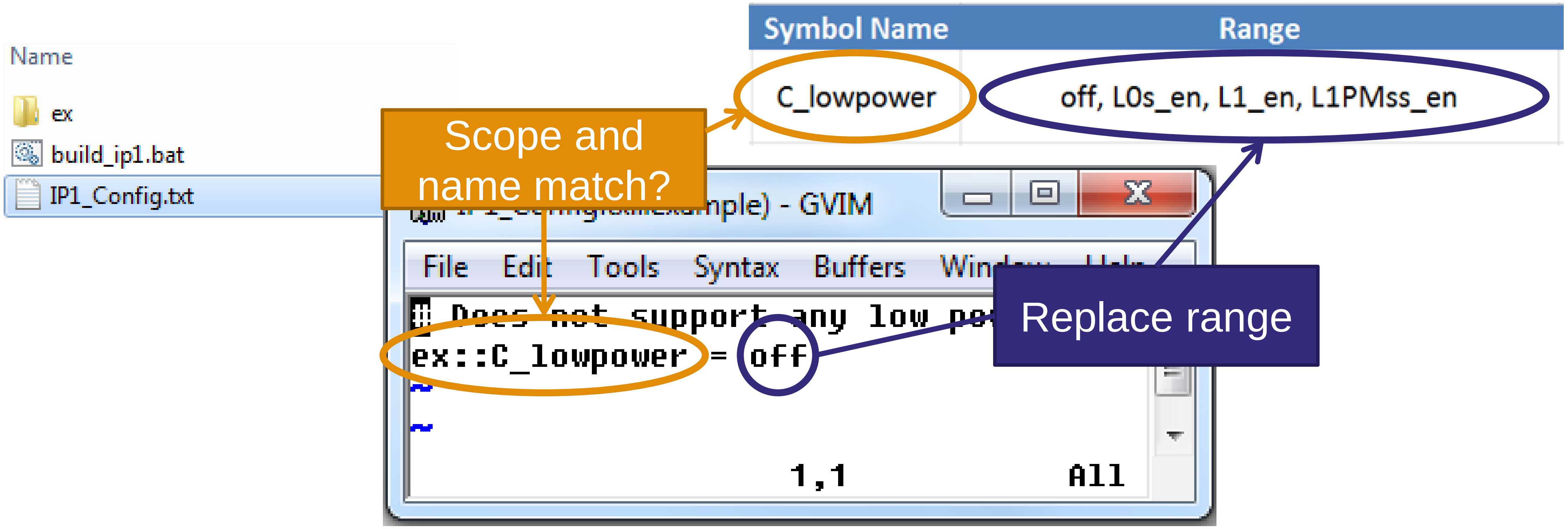
	A	B	C	D			
15	Name	RX Datapath Refined					
16	Symbol Name	rx_datapath_cg					
17	Description	Required RX path only important scenarios to sim/					
18	Instance Path	tb::rx_datapath_cg					
19	Point Types						
20	Points	Data	Control	ltssm_state	M_l		
21	*pkt_delim_cross*	$STP, $SDP, $END	1	L0			*Cover packet delimiters*
22	*L0s_wake_rx_cross*	$COM	1	L0s_rx_FTS	L0s_en	L0s_en	*Cover FTS receive*



Symbol Name	Range
Data	[8'h00:8'hff]
Control	0, 1
COM	8'hBC
STP	8'hFB
SDP	8'h5C
END	8'hFD

- group – instantiate variables in cross-scenarios used in a covergroup
- Any mode, config, or cover variable in this or any parent coversheet

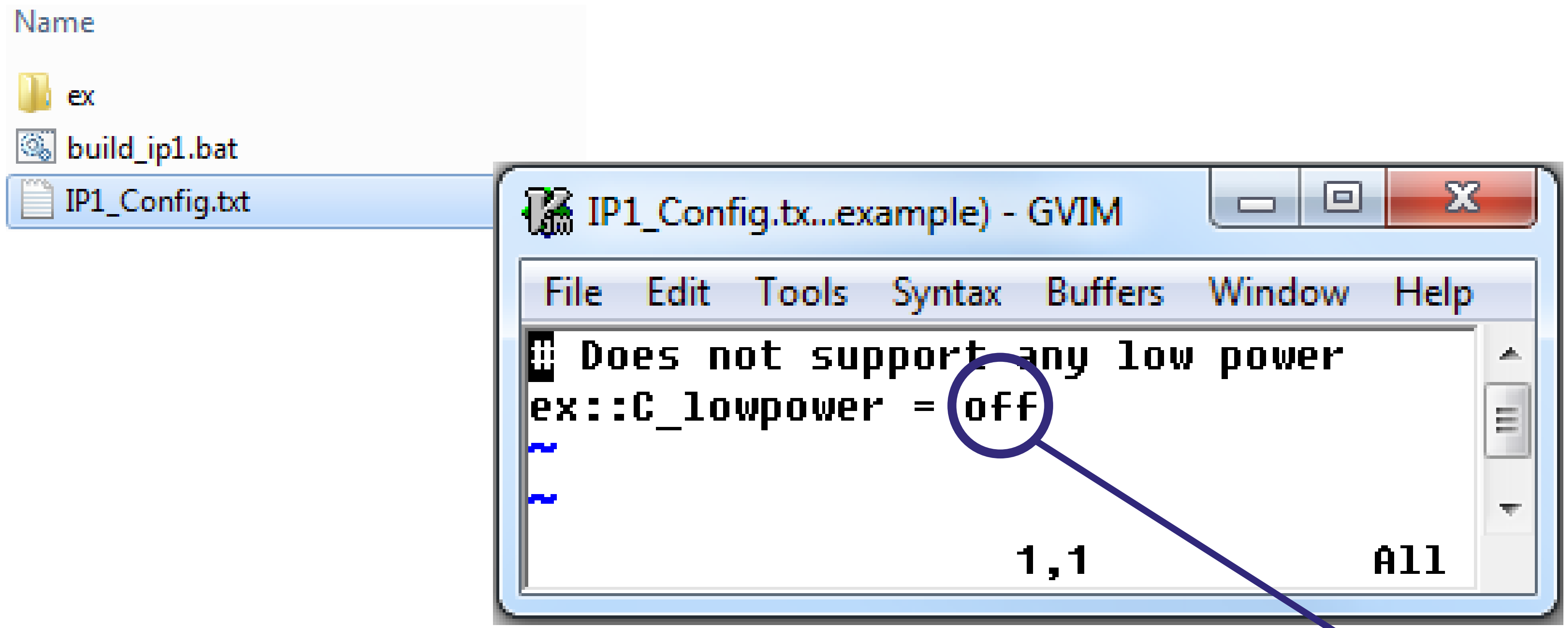
IP #1 Configuration Override



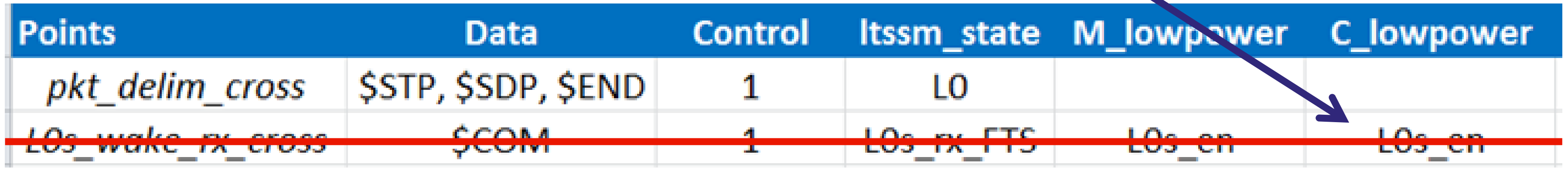
Symbol Name	Range
C_lowpower	off, L0s_en, L1_en, L1PMss_en

```
# Does not support any low po
ex::C_lowpower = off
~
~
                         1,1            All
```

IP #1 Configuration Affect

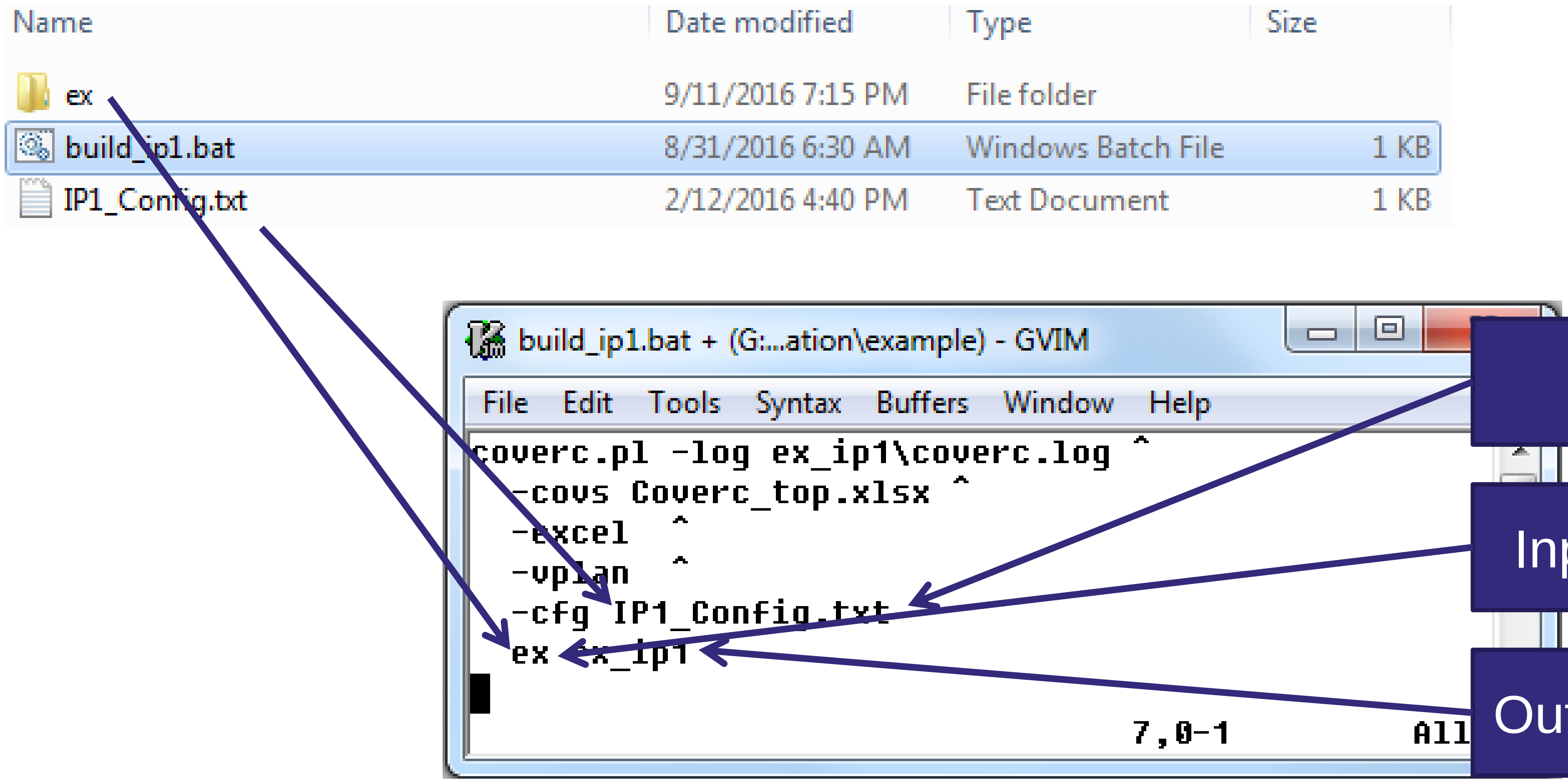

Name
ex
build_ip1.bat
IP1_Config.txt
IP1_Config.tx...example) - GVIM
File Edit Tools Syntax Buffers Window Help

```
# Does not support any low power
ex::C_lowpower = off
~
~
                        1,1           All
```



Points	Data	Control	ltssm_state	M_lowpower	C_lowpower
pkt_delim_cross	$STP, $SDP, $END	1	L0		
~~*L0s_wake_rx_cross*~~	~~$COM~~	~~1~~	~~L0s_rx_FTS~~	~~L0s_en~~	~~L0s_en~~

Generate Coverage for IP #1



Name	Date modified	Type	Size
ex	9/11/2016 7:15 PM	File folder	
build_ip1.bat	8/31/2016 6:30 AM	Windows Batch File	1 KB
IP1_Config.txt	2/12/2016 4:40 PM	Text Document	1 KB

build_ip1.bat + (G:...ation\example) - GVIM
File Edit Tools Syntax Buffers Window Help

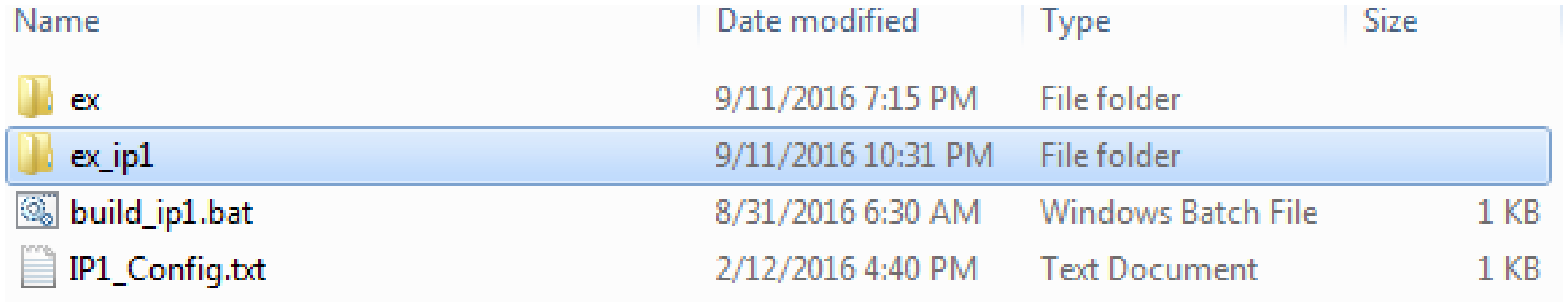
```
coverc.pl -log ex_ip1\coverc.log ^
  -covs Coverc_top.xlsx ^
  -excel  ^
  -vplan  ^
  -cfg IP1_Config.txt
  ex ex_ip1
```

IP Config

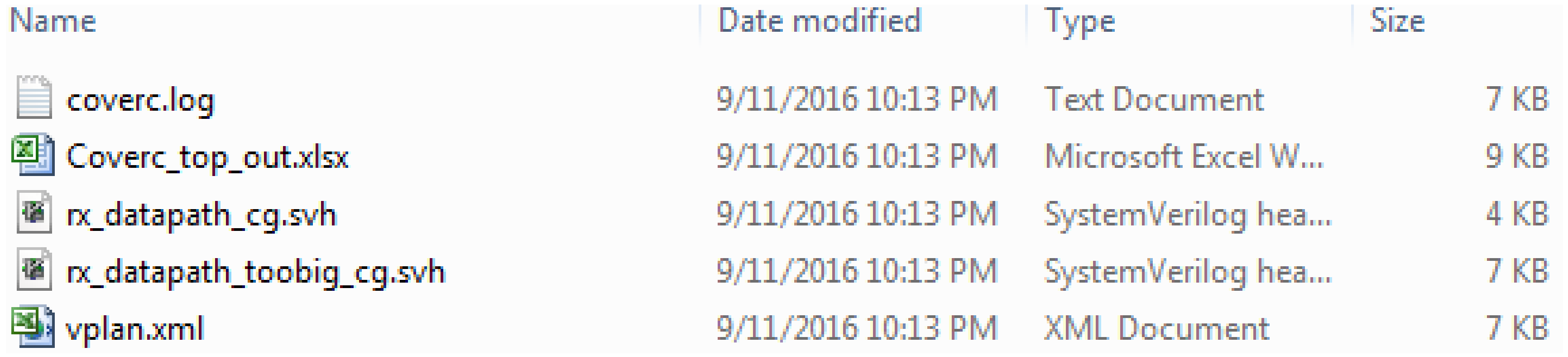
Input root block

Output root block

Generated Coverage for IP #1

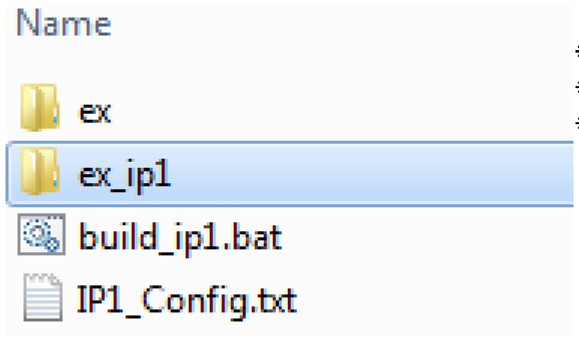


Name	Date modified	Type	Size
ex	9/11/2016 7:15 PM	File folder	
ex_ip1	9/11/2016 10:31 PM	File folder	
build_ip1.bat	8/31/2016 6:30 AM	Windows Batch File	1 KB
IP1_Config.txt	2/12/2016 4:40 PM	Text Document	1 KB

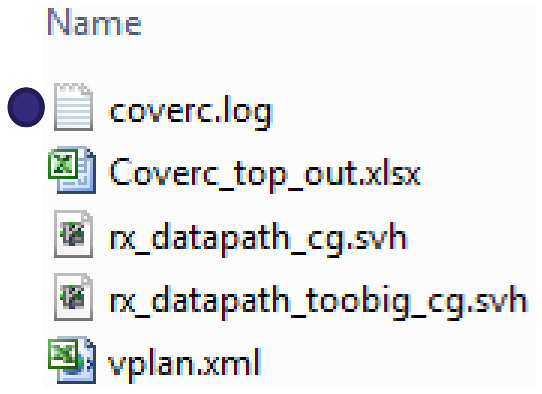


Name	Date modified	Type	Size
coverc.log	9/11/2016 10:13 PM	Text Document	7 KB
Coverc_top_out.xlsx	9/11/2016 10:13 PM	Microsoft Excel W...	9 KB
rx_datapath_cg.svh	9/11/2016 10:13 PM	SystemVerilog hea...	4 KB
rx_datapath_toobig_cg.svh	9/11/2016 10:13 PM	SystemVerilog hea...	7 KB
vplan.xml	9/11/2016 10:13 PM	XML Document	7 KB

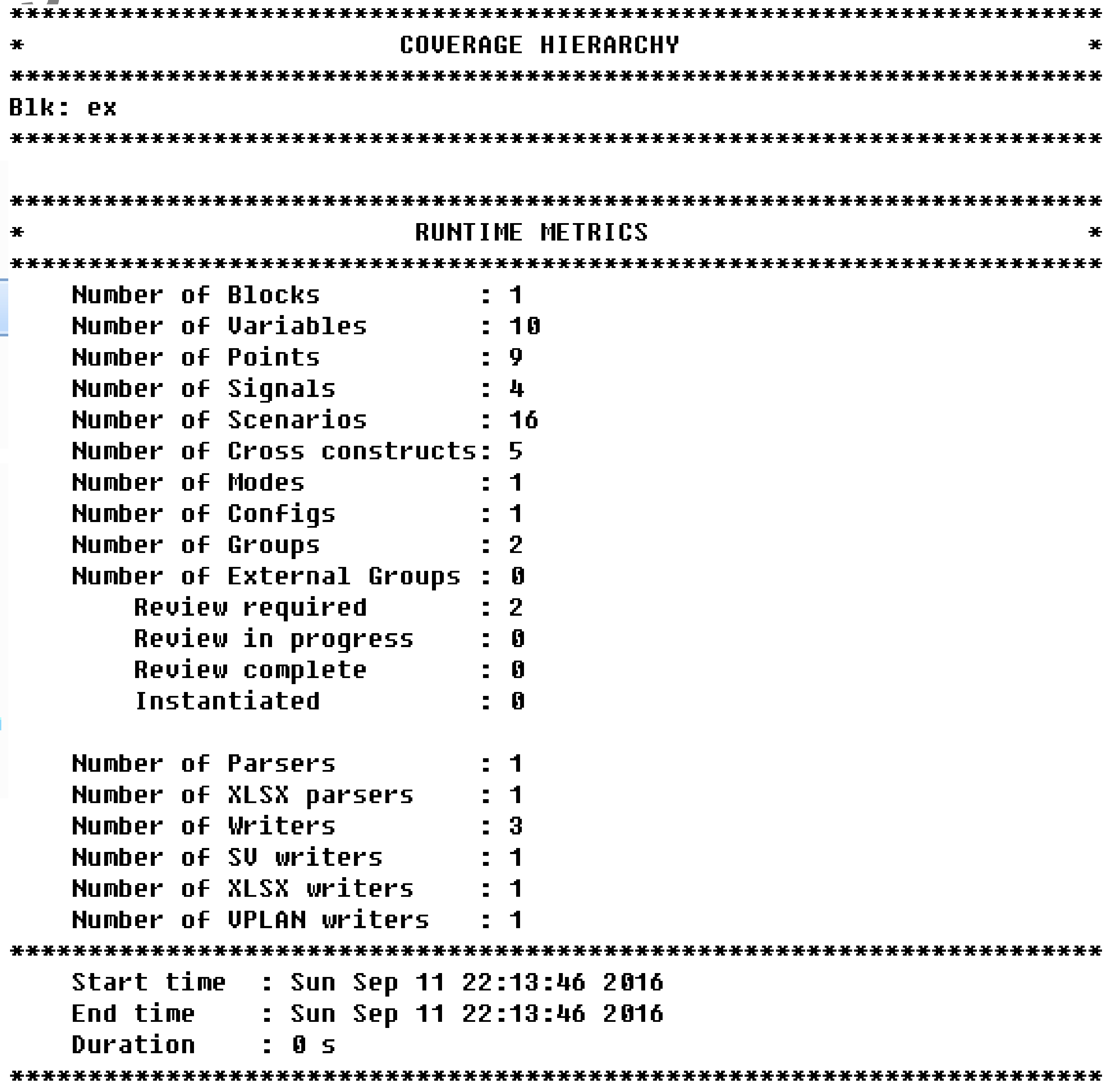
IP #1 Coverage


Name

ex
ex_ip1
build_ip1.bat
IP1_Config.txt


Name

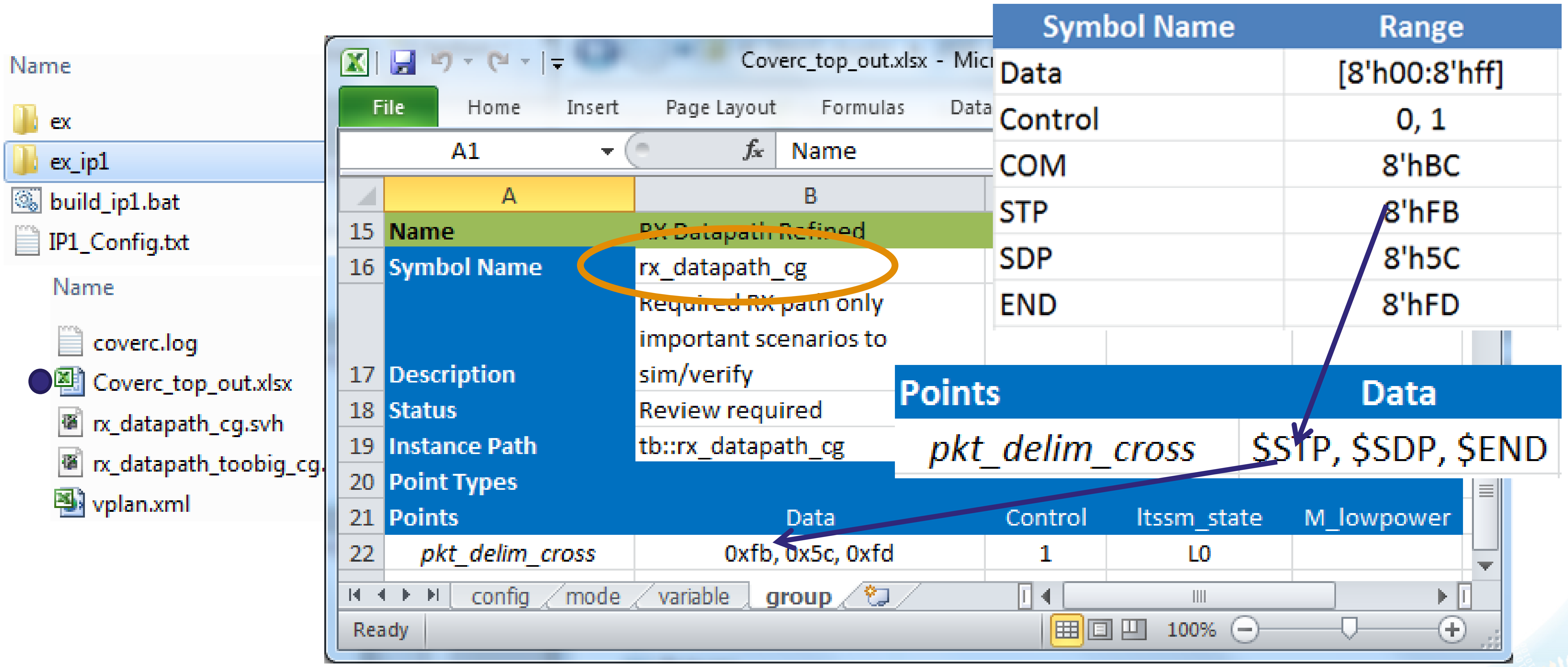
coverc.log
Coverc_top_out.xlsx
rx_datapath_cg.svh
rx_datapath_toobig_cg.svh
vplan.xml


```
**********************************************************************
*                        COVERAGE HIERARCHY                          *
**********************************************************************
Blk: ex
**********************************************************************

**********************************************************************
*                         RUNTIME METRICS                            *
**********************************************************************
    Number of Blocks          : 1
    Number of Variables       : 10
    Number of Points          : 9
    Number of Signals         : 4
    Number of Scenarios       : 16
    Number of Cross constructs: 5
    Number of Modes           : 1
    Number of Configs         : 1
    Number of Groups          : 2
    Number of External Groups : 0
        Review required       : 2
        Review in progress    : 0
        Review complete       : 0
        Instantiated          : 0

    Number of Parsers         : 1
    Number of XLSX parsers    : 1
    Number of Writers         : 3
    Number of SV writers      : 1
    Number of XLSX writers    : 1
    Number of VPLAN writers   : 1
**********************************************************************
    Start time   : Sun Sep 11 22:13:46 2016
    End time     : Sun Sep 11 22:13:46 2016
    Duration     : 0 s
**********************************************************************
```

IP #1 Coverage



Symbol Name	Range
Data	[8'h00:8'hff]
Control	0, 1
COM	8'hBC
STP	8'hFB
SDP	8'h5C
END	8'hFD

Points	Data
pkt_delim_cross	$STP, $SDP, $END

Name
- ex
- ex_ip1
- build_ip1.bat
- IP1_Config.txt

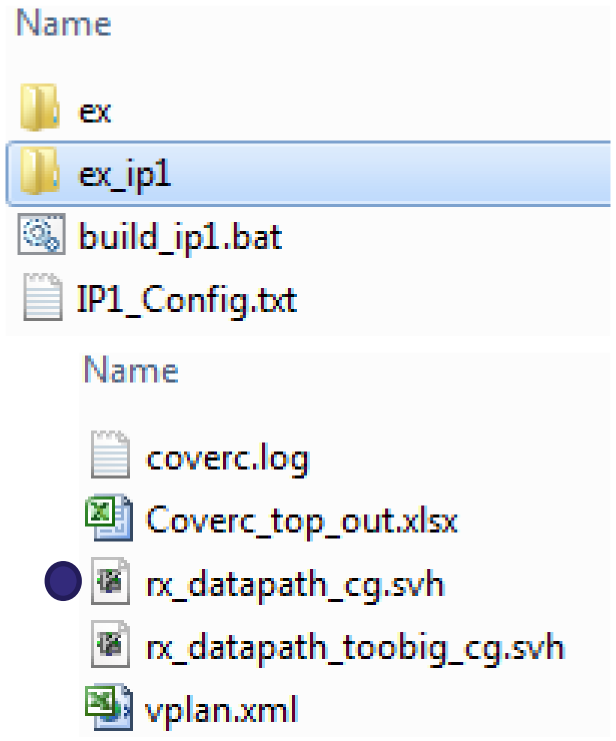
Name
- coverc.log
- Coverc_top_out.xlsx
- rx_datapath_cg.svh
- rx_datapath_toobig_cg.
- vplan.xml

Coverc_top_out.xlsx - Mic

A1 Name

	A	B			
15	Name	RX Datapath Refined			
16	Symbol Name	rx_datapath_cg			
17	Description	Required RX path only important scenarios to sim/verify			
18	Status	Review required			
19	Instance Path	tb::rx_datapath_cg			
20	Point Types				
21	Points	Data	Control	ltssm_state	M_lowpower
22	*pkt_delim_cross*	0xfb, 0x5c, 0xfd	1	L0	

IP #1 Coverage



Name

- ex
- ex_ip1
- build_ip1.bat
- IP1_Config.txt

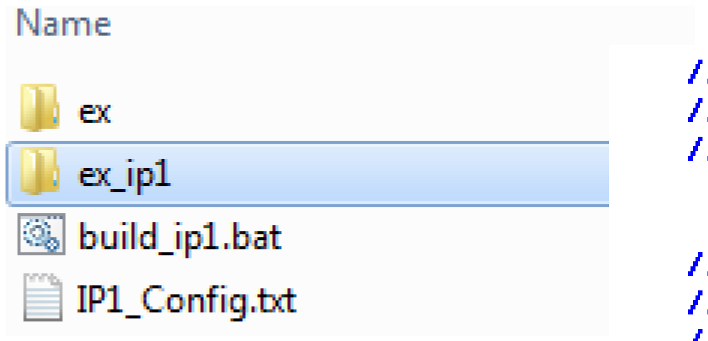
Name

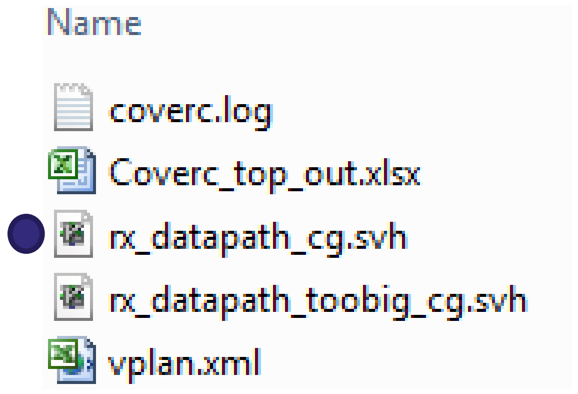
- coverc.log
- Coverc_top_out.xlsx
- ● rx_datapath_cg.svh
- rx_datapath_toobig_cg.svh
- vplan.xml

```
`ifndef RX_DATAPATH_CG__SVH
`define RX_DATAPATH_CG__SVH

// ----------------------------------------------------------------------
// Class: rx_datapath_cg
//
// Required RX path only important scenarios to sim/verify
//
// Summary:
//         Cover model scope  - ex::rx_datapath_cg
//         Total cover points - 3
//      Total cross scenarios - 3
//     Total cross constructs - 1
covergroup rx_datapath_cg;
```

IP #1 Coverage

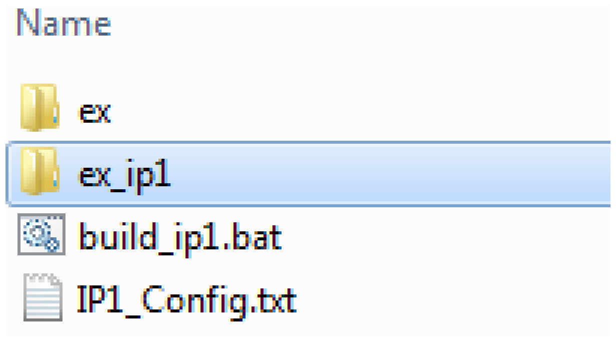

Name
ex
ex_ip1
build_ip1.bat
IP1_Config.txt

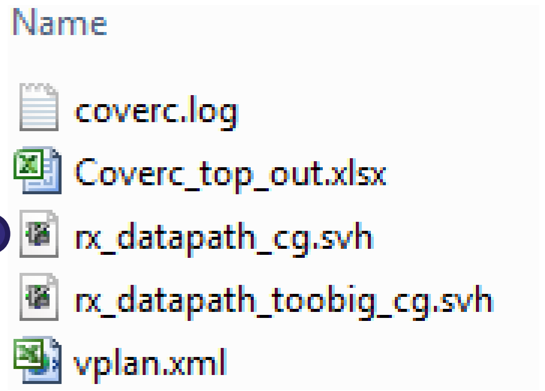

Name
coverc.log
Coverc_top_out.xlsx
● rx_datapath_cg.svh
rx_datapath_toobig_cg.svh
vplan.xml

```
// ------------------------------------------------------------------
// Group: Cover points
// ------------------------------------------------------------------


// Variable: Data
//
// Decoded data bus from DUT
//
// Summary:
//    Variable model scope - ex::Data
//    Point model scope    - ex::rx_datapath_cg::Data
//    Point bound to       - signal specified at variable
//                           (no port, local scope)
//    Total point bins     - 3
/* int signed */ Data: coverpoint data
{
    bins Data_0 = { 'hfb };
    bins Data_1 = { 'h5c };
    bins Data_2 = { 'hfd };
} // coverpoint Data
```

IP #1 Coverage

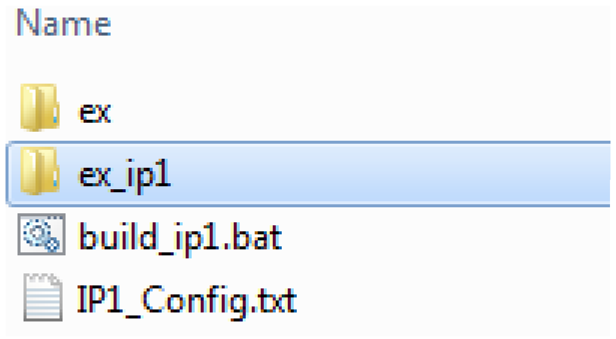

Name
- ex
- ex_ip1
- build_ip1.bat
- IP1_Config.txt


Name
- coverc.log
- Coverc_top_out.xlsx
- ● rx_datapath_cg.svh
- rx_datapath_toobig_cg.svh
- vplan.xml

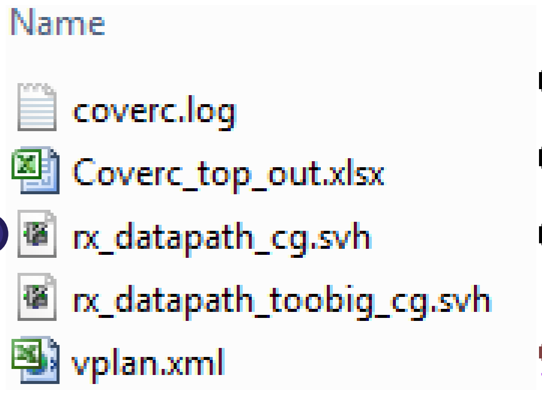
```
// Variable: Control
//
// Control character indication
//
// Summary:
//    Variable model scope - ex::Control
//    Point model scope    - ex::rx_datapath_cg::Control
//    Point bound to       - signal specified at variable
//                           (no port, local scope)
//    Total point bins     - 1
/* int signed */ Control: coverpoint ctrl
{
    bins Control_0 = { 'd1 };
} // coverpoint Control


// Variable: ltssm_state
//
// Enumeration of all LTSSM states possible.  Bound to DUT output.
//
// Summary:
//    Variable model scope - ex::ltssm_state
//    Point model scope    - ex::rx_datapath_cg::ltssm_state
//    Point bound to       - signal specified at variable
//                           (no port, hierarchical scope)
//    Total point bins     - 1
/* int signed */ ltssm_state: coverpoint tb.ltssm_o
{
    bins ltssm_state_0 = { L0 };
} // coverpoint ltssm_state
```

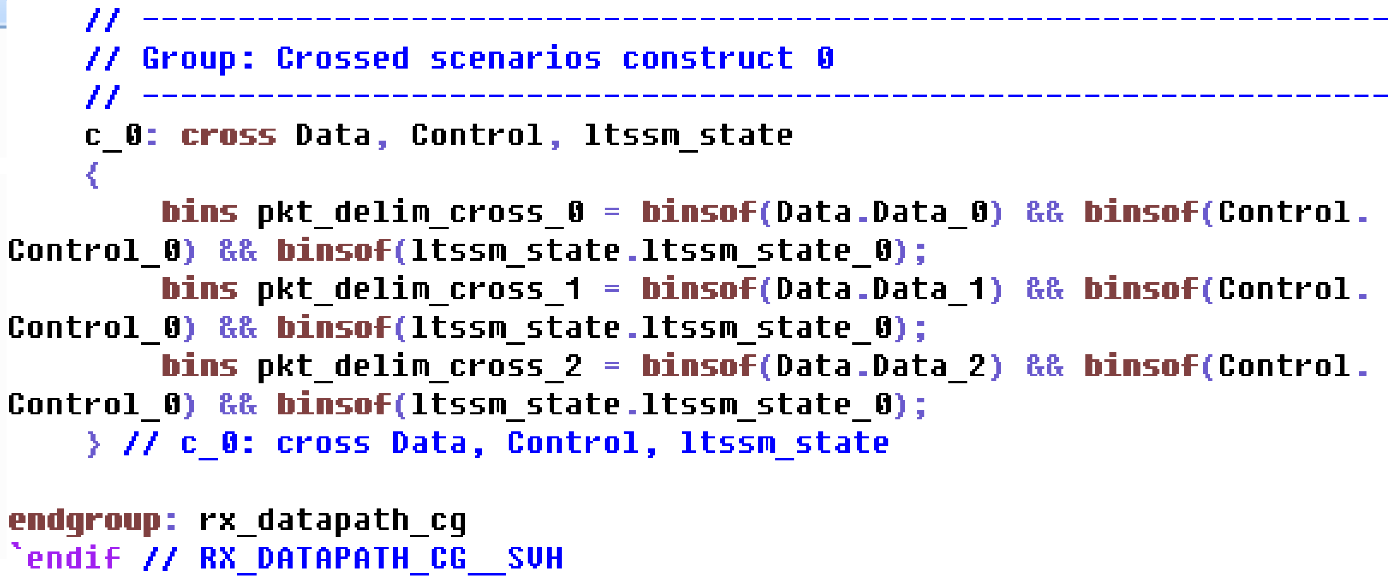
IP #1 Coverage


Name

ex
ex_ip1
build_ip1.bat
IP1_Config.txt

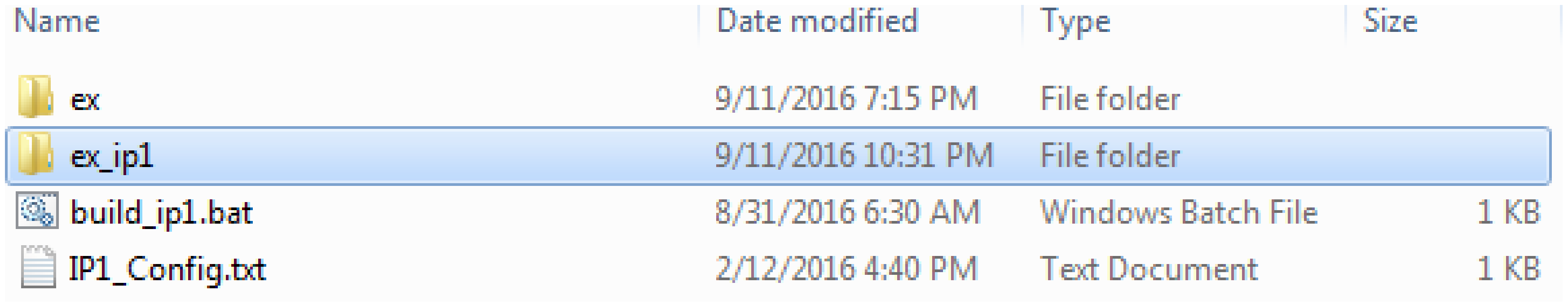

Name

coverc.log
Coverc_top_out.xlsx
● rx_datapath_cg.svh
rx_datapath_toobig_cg.svh
vplan.xml

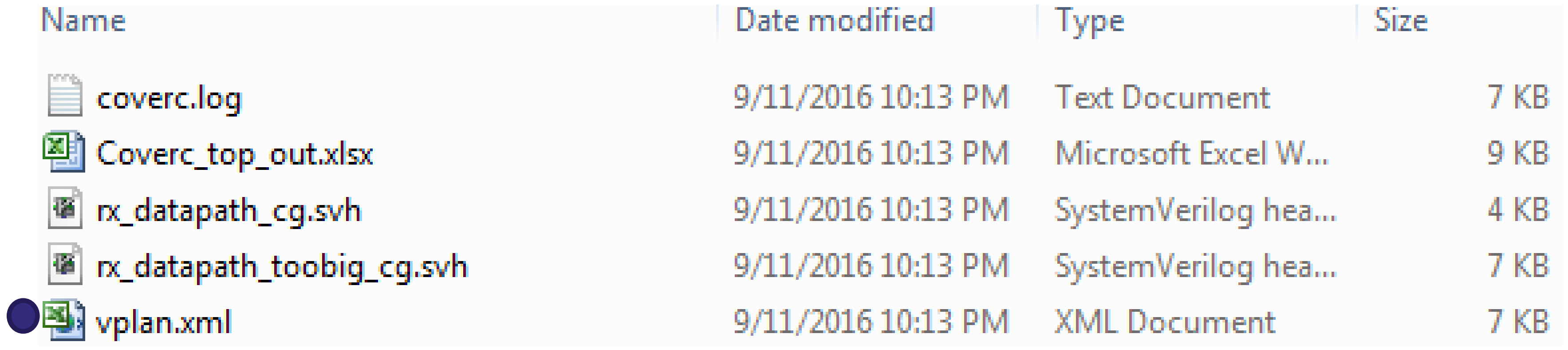

```
    // ----------------------------------------------------------------
    // Group: Crossed scenarios construct 0
    // ----------------------------------------------------------------
    c_0: cross Data, Control, ltssm_state
    {
        bins pkt_delim_cross_0 = binsof(Data.Data_0) && binsof(Control.
Control_0) && binsof(ltssm_state.ltssm_state_0);
        bins pkt_delim_cross_1 = binsof(Data.Data_1) && binsof(Control.
Control_0) && binsof(ltssm_state.ltssm_state_0);
        bins pkt_delim_cross_2 = binsof(Data.Data_2) && binsof(Control.
Control_0) && binsof(ltssm_state.ltssm_state_0);
    } // c_0: cross Data, Control, ltssm_state

endgroup: rx_datapath_cg
`endif // RX_DATAPATH_CG__SVH
```

IP #1 Coverage

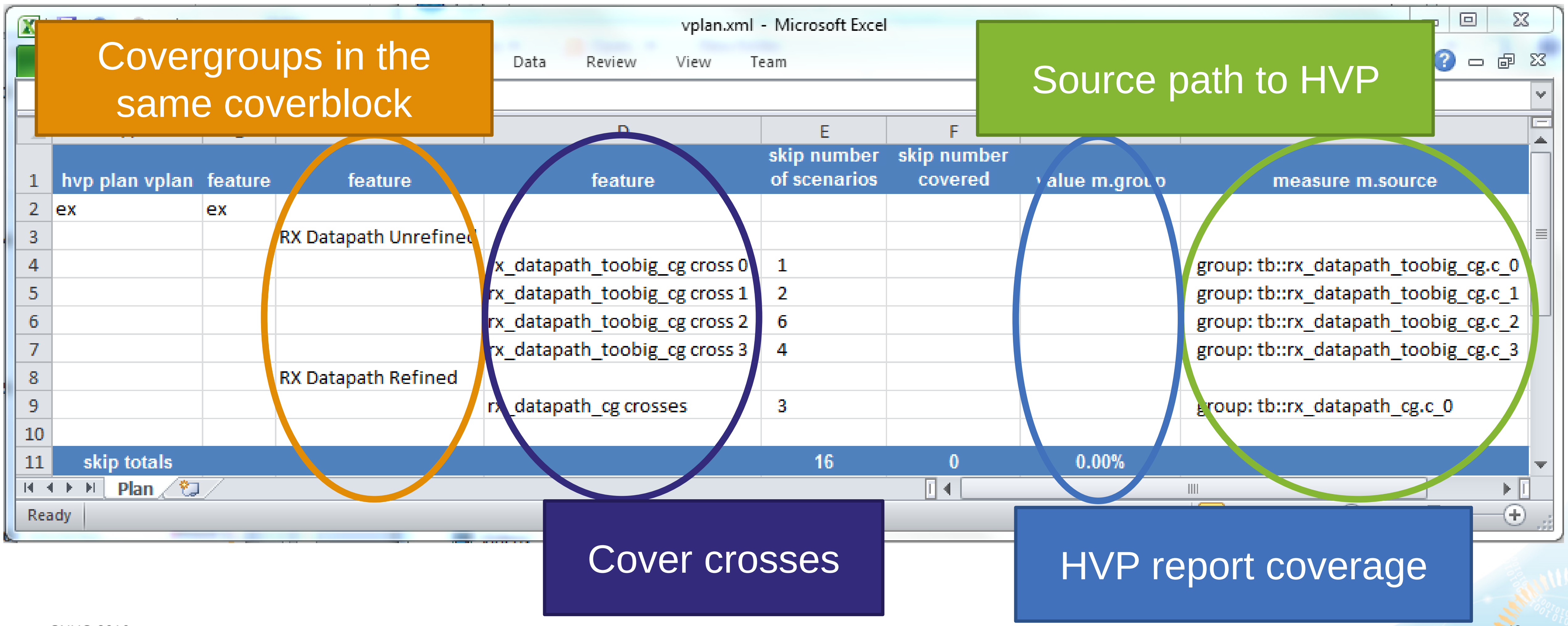


Name	Date modified	Type	Size
ex	9/11/2016 7:15 PM	File folder	
ex_ip1	9/11/2016 10:31 PM	File folder	
build_ip1.bat	8/31/2016 6:30 AM	Windows Batch File	1 KB
IP1_Config.txt	2/12/2016 4:40 PM	Text Document	1 KB



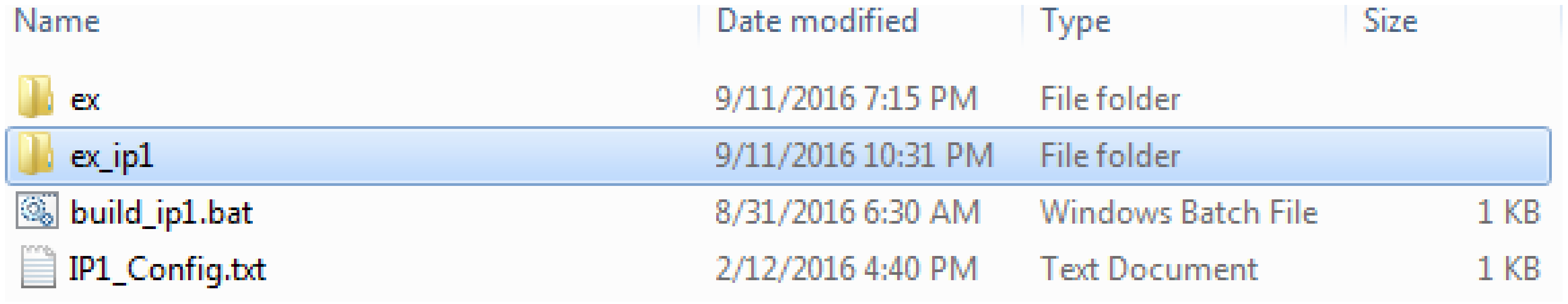
Name	Date modified	Type	Size
coverc.log	9/11/2016 10:13 PM	Text Document	7 KB
Coverc_top_out.xlsx	9/11/2016 10:13 PM	Microsoft Excel W...	9 KB
rx_datapath_cg.svh	9/11/2016 10:13 PM	SystemVerilog hea...	4 KB
rx_datapath_toobig_cg.svh	9/11/2016 10:13 PM	SystemVerilog hea...	7 KB
vplan.xml	9/11/2016 10:13 PM	XML Document	7 KB

IP #1 Coverage

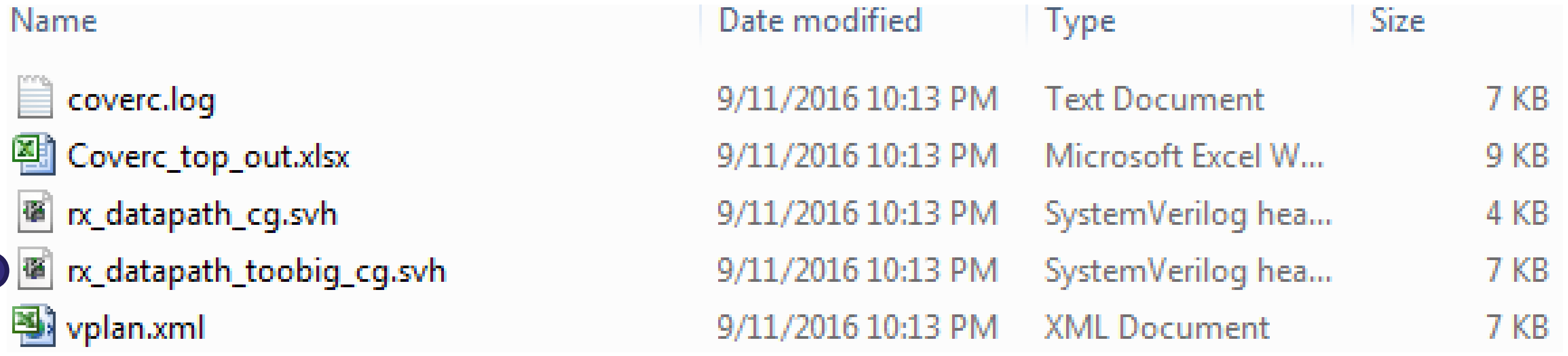


	A	B	C	D	E	F	G	H
1	hvp plan vplan	feature	feature	feature	skip number of scenarios	skip number covered	value m.group	measure m.source
2	ex	ex						
3			RX Datapath Unrefined					
4				x_datapath_toobig_cg cross 0	1			group: tb::rx_datapath_toobig_cg.c_0
5				rx_datapath_toobig_cg cross 1	2			group: tb::rx_datapath_toobig_cg.c_1
6				rx_datapath_toobig_cg cross 2	6			group: tb::rx_datapath_toobig_cg.c_2
7				rx_datapath_toobig_cg cross 3	4			group: tb::rx_datapath_toobig_cg.c_3
8			RX Datapath Refined					
9				rx_datapath_cg crosses	3			group: tb::rx_datapath_cg.c_0
10								
11	skip totals				16	0	0.00%	

IP #1 Coverage



Name	Date modified	Type	Size
ex	9/11/2016 7:15 PM	File folder	
ex_ip1	9/11/2016 10:31 PM	File folder	
build_ip1.bat	8/31/2016 6:30 AM	Windows Batch File	1 KB
IP1_Config.txt	2/12/2016 4:40 PM	Text Document	1 KB



Name	Date modified	Type	Size
coverc.log	9/11/2016 10:13 PM	Text Document	7 KB
Coverc_top_out.xlsx	9/11/2016 10:13 PM	Microsoft Excel W...	9 KB
rx_datapath_cg.svh	9/11/2016 10:13 PM	SystemVerilog hea...	4 KB
● rx_datapath_toobig_cg.svh	9/11/2016 10:13 PM	SystemVerilog hea...	7 KB
vplan.xml	9/11/2016 10:13 PM	XML Document	7 KB

Demos upon request!

Conclusions

- Handling functional coverage for highly configurable IP is HARD
  - Coverpoints are not in a vacuum
  - Static configuration
  - Dynamic mode-of-operation

- This is one solution we have employed in:
  - 2 programs (currently active)
  - 7 customers (so far)
  - Up to 2 at one time (so far)

Thank You