Formal Verification Applied to the Renesas MCU Design Platform Using the OneSpin Tools

Toru Shimizu, Ph.D. and
Satoshi Nakano, Renesas Electronics Corp.
Colin Mason, OneSpin Solutions Japan K.K.

2013.02.03

Authors

- Toru Shimizu, Ph.D.

  Senior Chief Professional

  Technology Planning Div Technology Development Unit

  Renesas Electronics Corporation

  +81(3)4226-6591

  toru.shimizu.xn@renesas.com

- Satoshi Nakano,

  Section Manager, MCU Platform Development Department 1

  Platform Integration Division, Technology Development Unit

  Renesas Electronics Corporation

  +81(72)787-5230

  satoshi.nakano.vj@renesas.com

- Colin Mason

  Business Development Manager

  OneSpin Solutions Japan K.K.

  +81(3)4530-3865

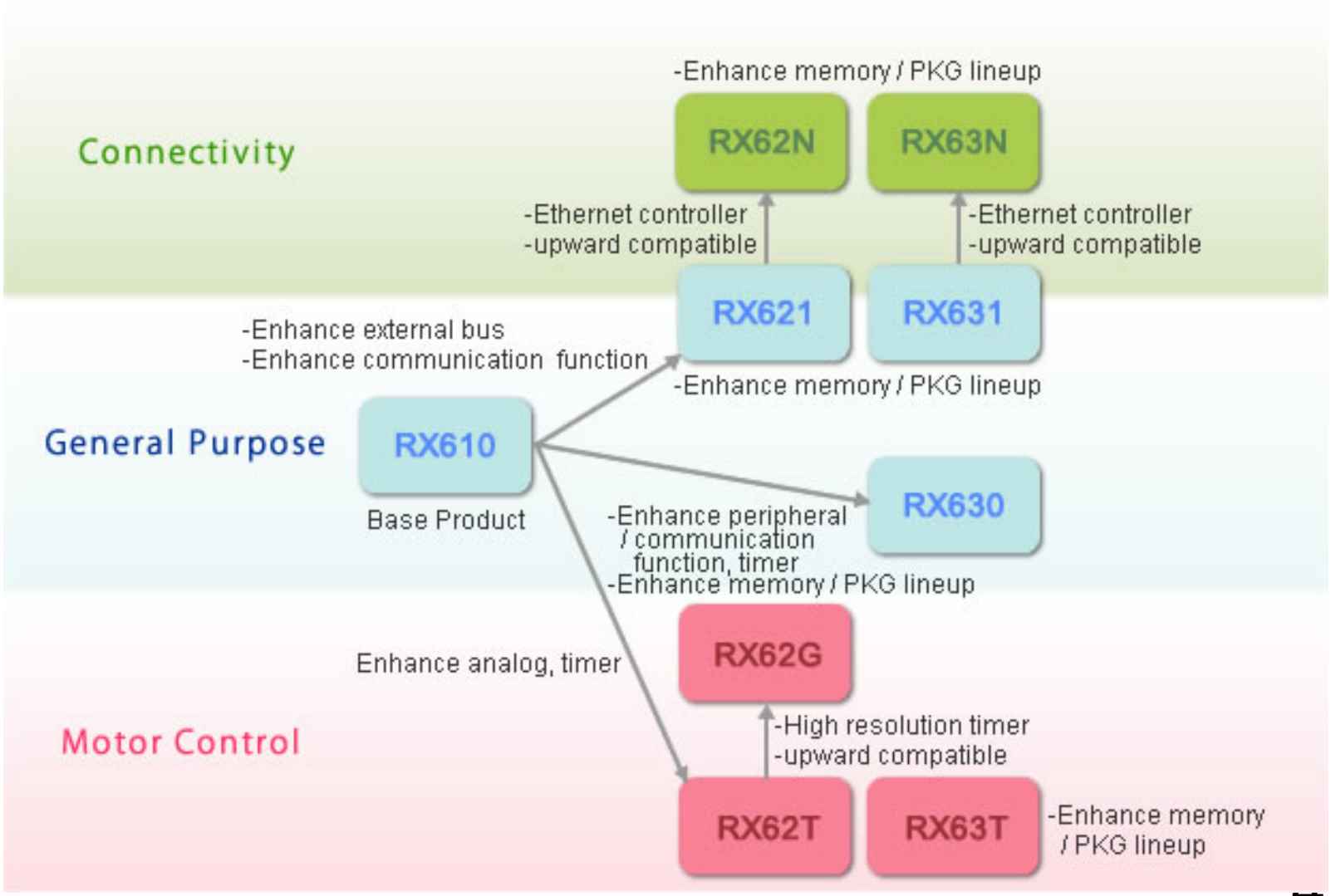
  colin.mason@onespin-solutions.com

Abstract

- Due to the wide range of different MCU types that are developed within a series of Renesas MCU's in order to satisfy different applications we have developed the MCU-PF platform for both design and verification of a complete series of MCUs. A key issue is the effective verification for a combination of multiple IP components - this presents a significant task.
- The test-case based simulation method of verification is widely used, but test-cases are not reusable among MCU designs, and modification of the test-cases and the testbench takes a long time. Verification coverage is only partial when using test-case based simulation. As such we have developed the capability to use a Formal method of verification that provides full coverage of combinations of assertions while delivering significant reductions in both testing and verification time. Additional efficiency has been gained because the IP-Assertions are fully reusable along with the IP itself. This paper explains the methodology, its advantages and performance results compared to the simulation verification method.

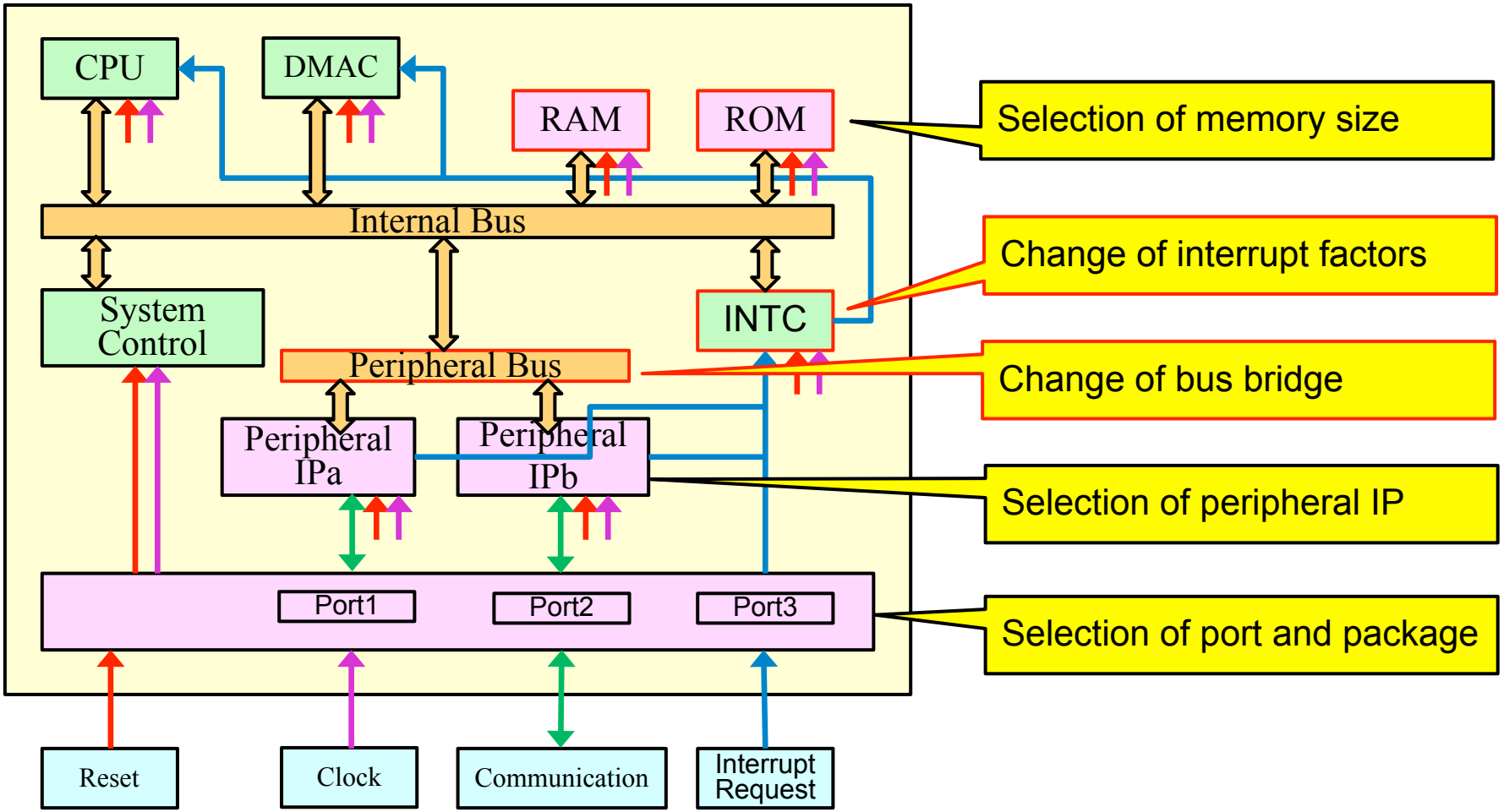
Within A Single MCU "Series" There Are A Wide Variety of Products:

– This Is The Driver For MCU-PF



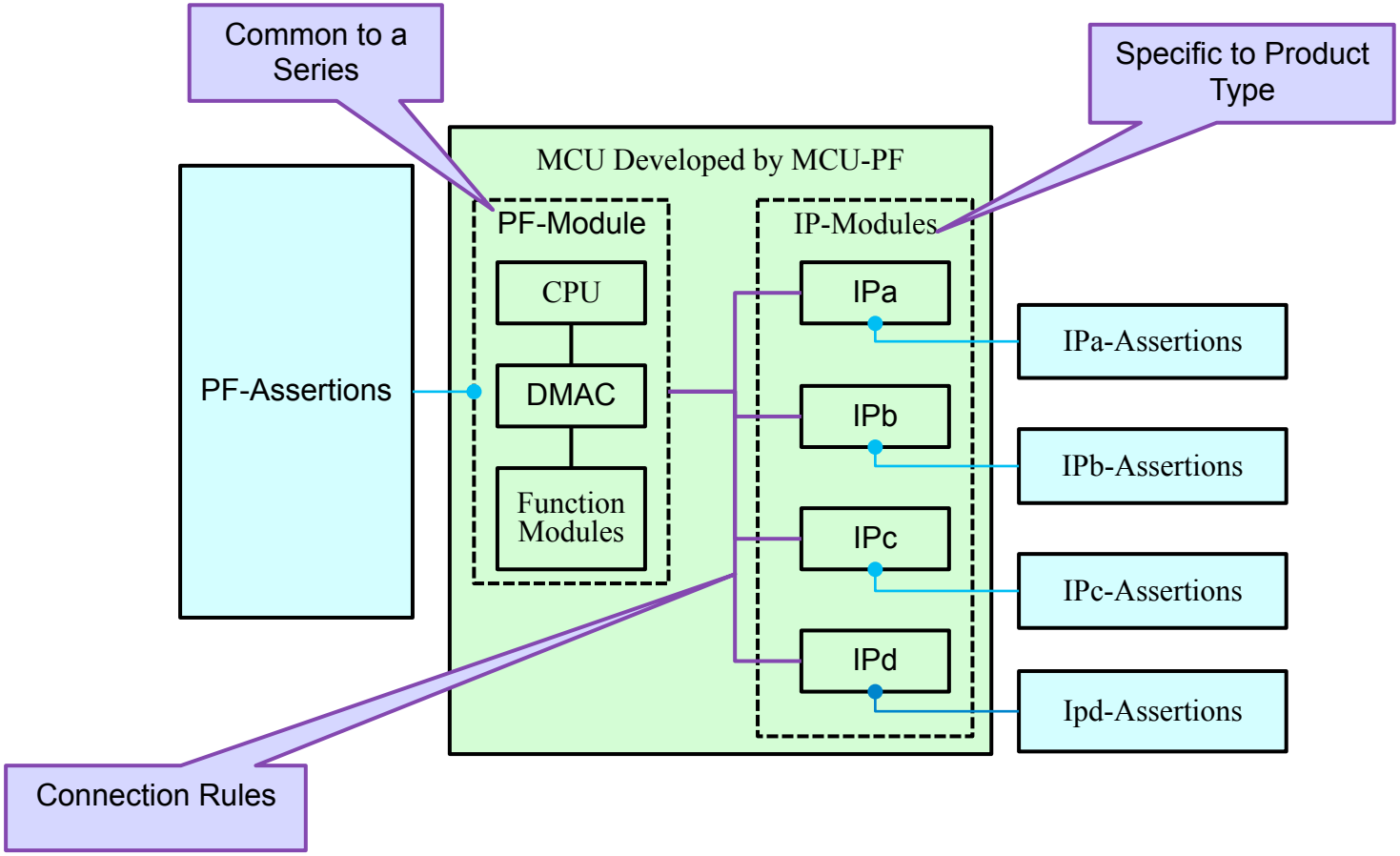
What is the MCU Platform (MCU-PF) ?

- MCU-PF is a platform for the efficient development of new products within the same series -



MCU-PF Configuration

- Services provided by MCU-PF for MCU development are PF-Modules, IP Modules, Connection Rules and PF and IP Assertions.

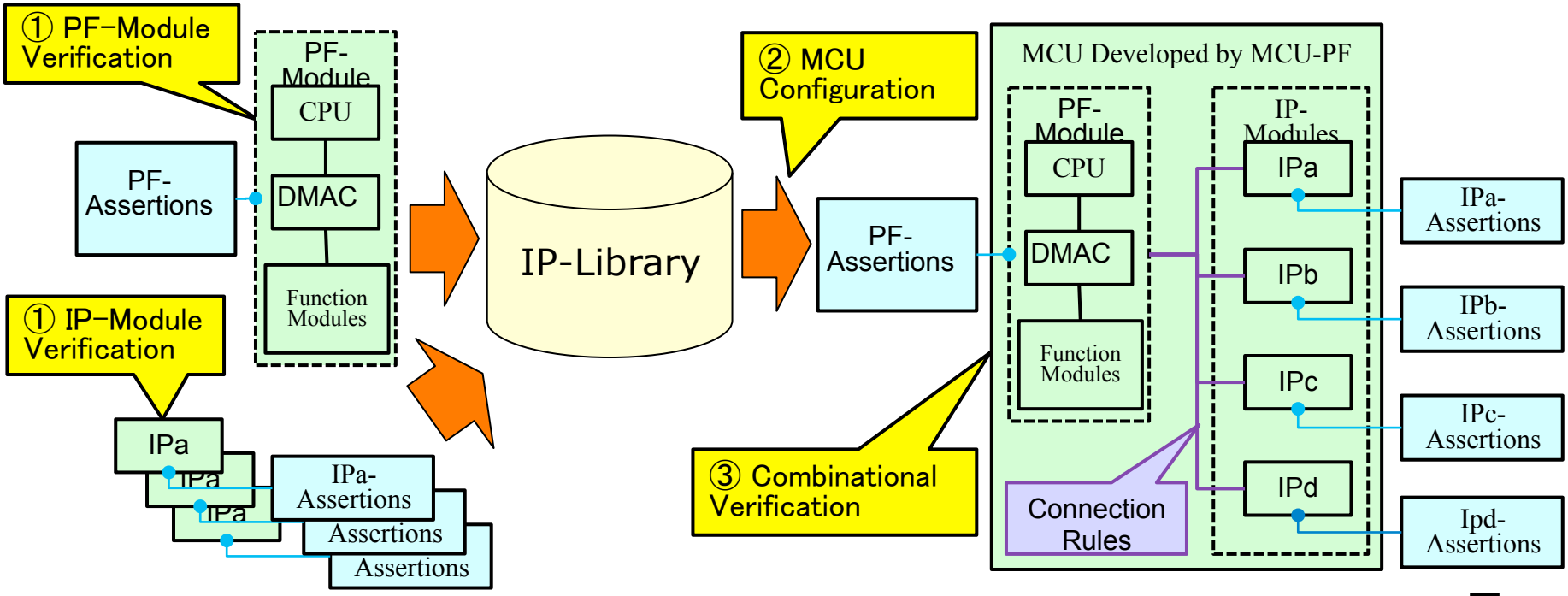


MCU Development-Flow Based on MCU-PF

① **Module Verification:** The PF-Module is sufficiently verified in advance and IP-Modules are verified before they are connected to PF-Modules.

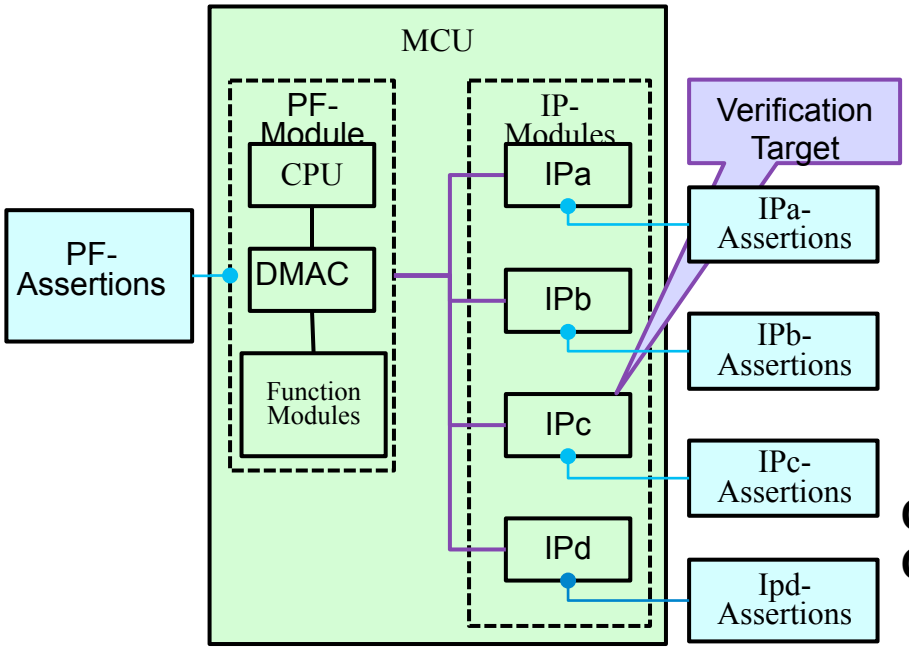
② **MCU Configuration:** An MCU is configured by connecting the PF-Module and selected IP-Modules from the IP-Library checking Connection-Rules.

③ **MCU Verification:** The MCU design is verified for operation combinations among the PF-Module and IP-Modules.

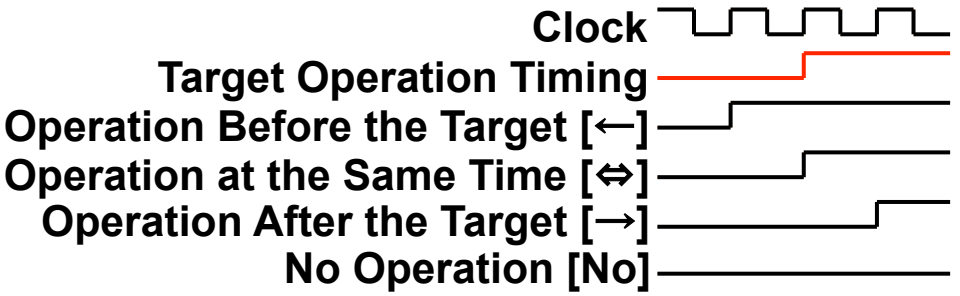


The Challenge of MCU Verification

- Verification of operation combinations of multiple IP-Modules is a significant task.
- If the IPc is the verification target, its operation should be verified on EVERY combination of parallel operations of IPa, IPb and IPd.
- The number of operation combinations becomes huge!

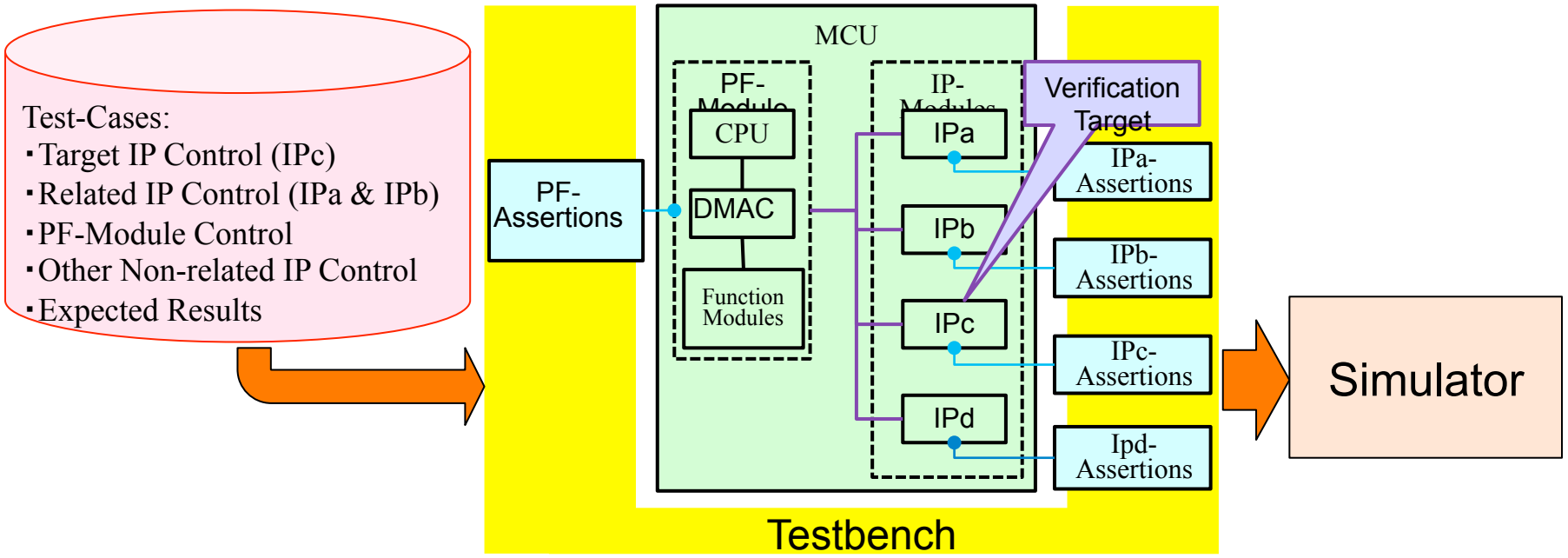


IP Combination	IPa	IPb	IPc	IPd
Operation Combination	←/⇔/→	No	Target	No
	No	←/⇔/→	Target	No
	←/⇔/→	←/⇔/→	Target	No
	:	:	:	:



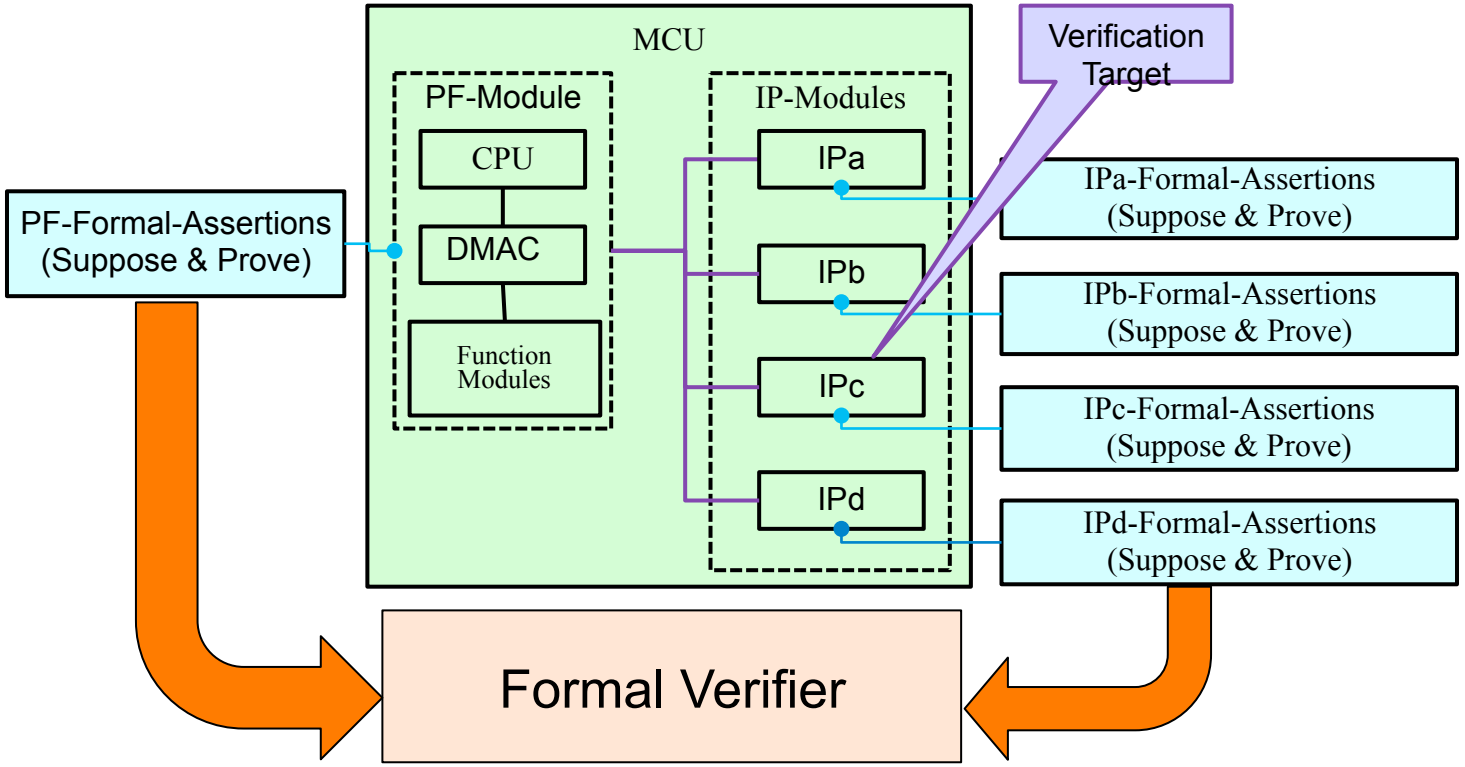
Simulation Verification of MCUs

- Verification is done by simulating test-cases on a testbench
- Each test-case controls all IPs, including target and related IP
- All test-cases and the testbench must be revised for a different MCU configuration, even if most of the IP is reused
- The verification coverage depends on the test-cases



Formal Verification of MCUs

- Verification proves consistency of formal assertions for given IP combinations.
- Each set of formal assertions includes only descriptions of the specified IP. It is reusable, similar to the IP.
- The verification coverage is perfect, if it is proved.



An Example of a Formal Assertion

- Only INTC resources specified in assertion

```
////////// timing ////////////////////////////
 sequence t_req_a; await(nxt(t,1), !ir_sig, max_a_wait); endsequence
 sequence t_req_n; await(nxt(t_req_a,1), ir_sig, max_n_wait);
endsequence
 ////////// property ////////////////////////////
property ir_cpu_negedge(ir_sig,ir_num,ier,ier_v,ipr,level,irqmd,ir);
  disable iff(`intc.RESET_BUS == 1'b0)
  // set SFR of INTC
  during(t,nxt(t_req_n,2),ier == ier_v) and
  during(t,nxt(t_req_n,2),ipr == level) and
  during(t,nxt(t_req_n,2),irqmd ==  2'b01) and
  during(t, nxt(t, 0),  ir ==  1'b0) and
  // set Interrupt input (ir_sig)
  during_excl(t_idle, t_req_a, ir_sig ) and
  during_excl(t_req_a, t_req_n, !ir_sig ) and
  during(t_req_n, nxt(t_req_n,1), ir_sig ) and
implies
  // check Interrupt output
  during(nxt(t_req_a,2),nxt(t_req_n,2),`INTNUM[7:0]==ir_num) and
  during(nxt(t_req_a,2),nxt(t_req_n,2),`INTRQLV[3:0]==level) ;
endproperty
```

Setup interrupt timing

Initialize interrupt controller INTC

Setup interrupt source

Check that INTC accepts interrupt

Evaluation of Our Formal Method

- Test development and execution data for "INTC."
- The formal method achieves full coverage verification for any combination of interrupt sources within a practical time.

Test Spec. of Each Test-case or Assertion		1. Simulation	2. Formal Verification	
			a. Prove an IP-assertion on the PF-assertion	b. Prove combination of all IP-assertions on the PF-assertion
Interrupt Timing x Interrupt Req. Src. x Interrupt Ack. Dest.		1 timing x 1 source x 5 destination x 547 test-cases	1 timing x 1 source x 5 destinations, on PF-assertion	1 timing x 1 source x 5 destinations x 2^547 combinations
Development Test-case or Assertion	Language	Assembly Lang.	Operational ABV	
	# Tests	547	547	
	# Lines	131,280	12,729	
	Development	40.8 days	4.5 days	
Testing	Total Time	547.0 Hour	9.0 Hour	72.9 Hour

- An interrupt source is selected from 547 request events in the INTC.
- An interrupt destination is selected from CPU and 4 channels of DMAC in the PF.

Evaluation of Our Formal Method

- Test development and execution data for "INTC."
- The formal method achieves full coverage verification for any combination of interrupt sources within a practical time.

Specification of each testcase or assertion		**Simulation**	**Formal Verification**	
			Prove an IP-assertion on the PF-assertion	Prove combination of all IP-assertions on the PF-assertion
Interrupt Timing **Interrupt Req. Source** **Interrupt Ack. Destination**		1 timing x 1 source x 5 destination x 547 testcases	1 timing x 1 source x 5 destinations x on PF-assertion	1 timing x 1 source x 5 destinations x 2^547 combinations
Develop. Statistics	**Language**	Assembly Lang.	Operational ABV	
	# Tests	547	547	
	# Lines	131,280	12,729	
	Dev. Time	40.8 days	4.5 days	
Execution	**Total Time**	547.0 hours	9.0 hours	72.9 hours

- An interrupt source is selected from 547 request events in the INTC.
- An interrupt destination is selected from CPU and 4 channels of DMAC in the PF.

Summary

- MCU-PF has been developed for both design and verification of a series of MCUs. Effective verification for a combination of multiple IP components is a significant task.

- The test-case simulation method is widely used, but test-cases are not reusable among MCU designs, and modification of the test-cases and the testbench takes a long time. Verification coverage is partial in the test-case simulation.

- Our proposed use of the Formal Methods (OneSpin) provides full coverage of combinations of assertions while delivering significant reductions in both testing and verification time. Additional efficiency is delivered because the IP-Assertions are reusable along with the IP itself.

Platform IP Combination

- MCU-PF provides a standardized design and verification platform for a series of MCUs. Verification of IP combination significant task
- Simulation widely used, but testcases are not reusable, time consuming to develop, and provide only partial coverage
- Formal Methods provide full coverage of IP combinations while delivering 9X reduction development time, 7X reduction execution time, and reusable tests

Thank you!