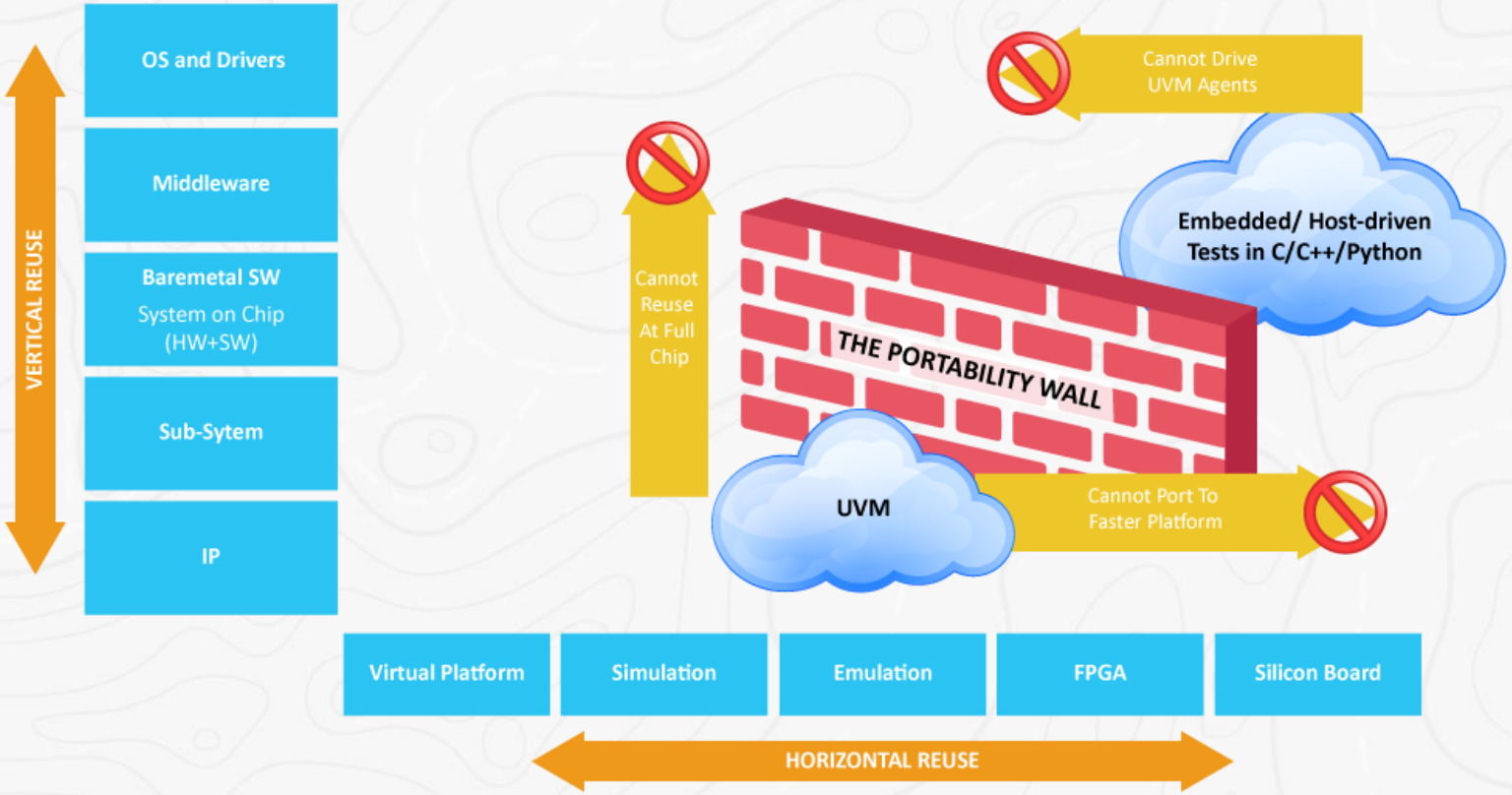
2022 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

Portable Stimulus Standard's Hardware-Software Interface (HSI) to validate 4G-5G Forward Error Correction Encoder/Decoder IP in emulation & silicon

Joydeep Maitra, Suresh Vasu, Nithin Venkatesh, Suhas Reddy,
Luis Campos, Vinit Shenoy

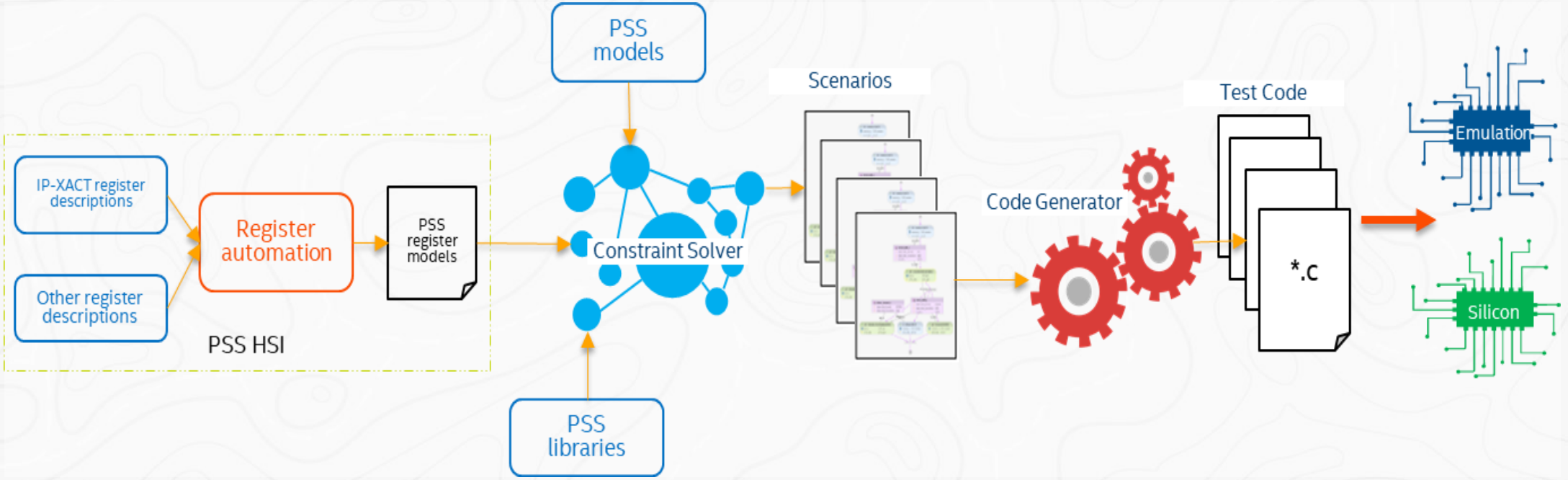
accellera SYSTEMS INITIATIVE

Problem Statement



- Configuration Space
- Memory Allocation
- Scheduling Flow
- Re-use across platforms
- Driver development

HSI Development Flow




```
package ip_dma_MEM_regs_pkg {
 component ip_dma_MEM_c : reg_group_c {

  struct IP_DMA_CONFIG_0_REG_reg_s : packed_s<> {
     bit[30] RESERVED_0;
     bit[2] QUEUE_MODE;
  }

  reg_c<IP_DMA_CONFIG_0_REG_reg_s> IP_DMA_CONFIG_0_REG ;
     exec init {
          IP_DMA_CONFIG_0_REG.offset = 0x0;
     }
  }
}
```

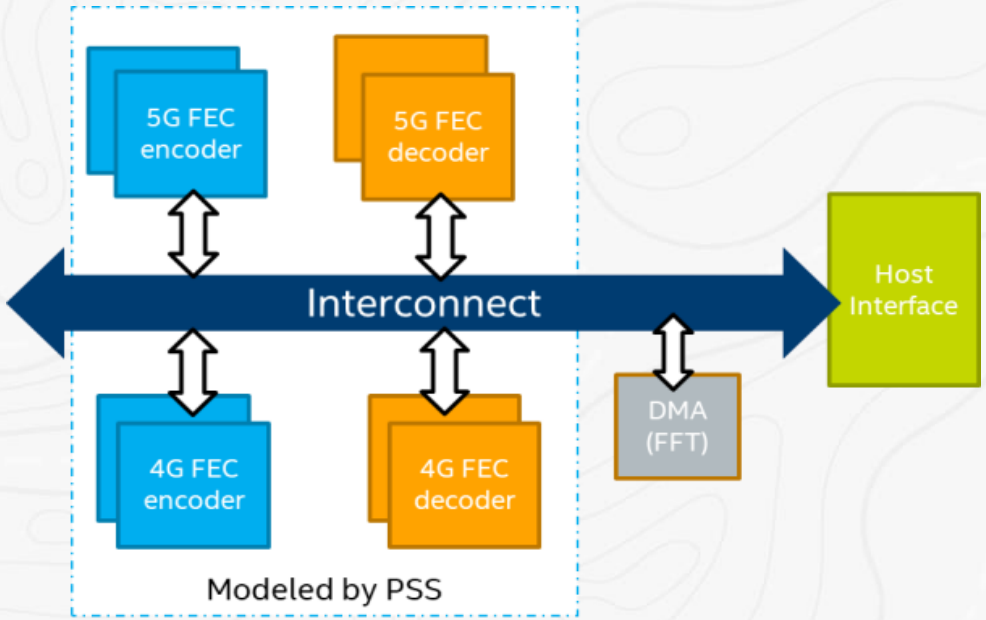
```
action ip_reg_write {
  rand bit[2] qmode;

  // DMA register struct handle
  ip_dma_MEM_c::IP_DMA_CONFIG_0_REG_reg_s reg_inst;

  exec post_solve {
     reg_inst.QUEUE_MODE = qmode;
  }

  exec body {
     comp.dma_regs.IP_DMA_CONFIG_0_REG.write(reg_inst);
  }
}
```

Scenario : 4G-5G Accelerator Block (DUT)




```
// Instantiate FEC-SS Register Files
pci_configreg_CFG_c pf_regs;
ip_fecdl4g_configreg_MEM_c dl_4g_regs[NUM_OF_4G_FEC_DL];
ip_fecdl5g_configreg_MEM_c dl_5g_regs[NUM_OF_5G_FEC_DL];
ip_fecul4g_configreg_MEM_c ul_4g_regs[NUM_OF_4G_FEC_UL];
ip_fecul5g_configreg_MEM_c ul_5g_regs[NUM_OF_5G_FEC_UL];
pci_configreg_CFG_c vf_cfg_regs[NUM_OF_VF];
```

table: ip_4g_fec_dl		
@package: ip_pkg	@size_const: NUM_OF_4G_FEC_DL	@struct: ip_core_info_s
#inst_id	#inst_name	#base_addr
0	"4G_FEC_DL_0"	0x2000_A000_0000
1	"4G_FEC_DL_1"	0x2000_B000_0000
2	"4G_FEC_DL_2"	0x2000_C000_0000

Summary & Results

HSI Features Used :

- Register File Instantiation and Address Factorization
- Handling Non-Contiguous Register Arrays
- Combining PSS Randomization with HSI Sequences
- Incorporating Interrupt Service Routines (ISRs) into HSI
- HSI and Efficient Target Code Generation

```
reg_c<IP_INGRESS_AQ_reg_s> IP_INGRESS_AQ[INGRESS_QUEUE_SIZE];
exec init {
  foreach(IP_INGRESS_AQ[i]) {
    IP_INGRESS_AQ[i].offset = i * 0x8;
  }
 }

comp.hi_vf_regs[vf_id].QMGR_INGRESS_AQ[queue_id].write_val(enqueue_data);
```

- ✓ Reduction of Test development time by 70%
- ✓ Re-use across execution platforms
- ✓ IP coverage checkout
- ✓ Concurrency and stress-scenarios
- ✓ Scalable test framework

Q & A