BREKER™ THE LEADER IN PORTABLE STIMULUS

SoC Verification and the Synthesizable VerificationOS

DVCon 2021 Workshop

2021 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

VIRTUAL | MARCH 1-4, 2021

Synthesizable VerificationOS DVCon21 Workshop

- High attendance workshop from DVCon, February 2021
  SoC Verification and the Synthesizable VerificationOS

  https://brekersystems.com/soc-verification-and-the-synthesizable-verificationos-workshop/

- Presenters
  - Introduction & Problem Discussion
    Mike Chin, Principal Engineer, Intel
  - Technology Overview & Practical Use Models
    Adnan Hamid, CTO, Breker Verification systems

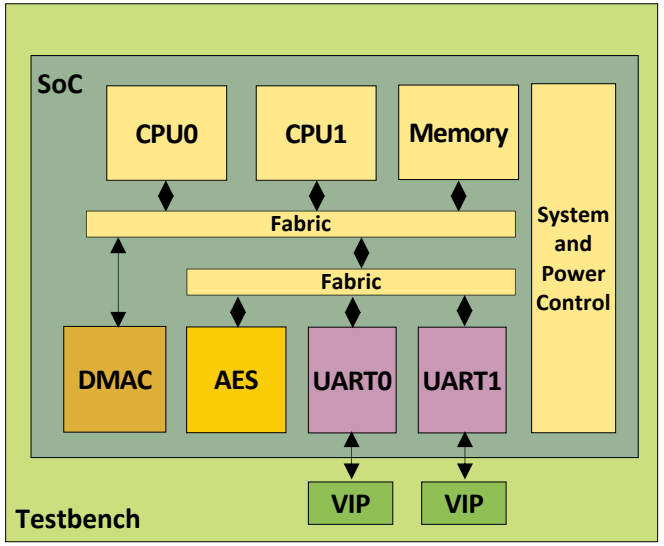
- Only Overview provided here (in 15 mins)

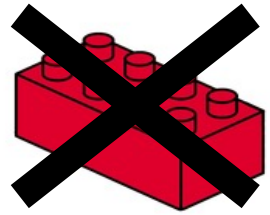
Evolving SoC Verification

- Functional verification often focused at the IP level
- SoC: interconnected, well-defined IP blocks
  - Could be re-verified using real data and software
  - Evolved from “ASIC on board” approach

However…

- Modern SoC complexity requires more verification
  - Infrastructure complexity:
    interconnect, coherency, security, etc.
  - Functionality across multiple blocks & SW
  - Concurrent profiling using same resources
  - Parallel firmware development
- SoC improvements could also drive other process phases
- SoC Verification requires a new look

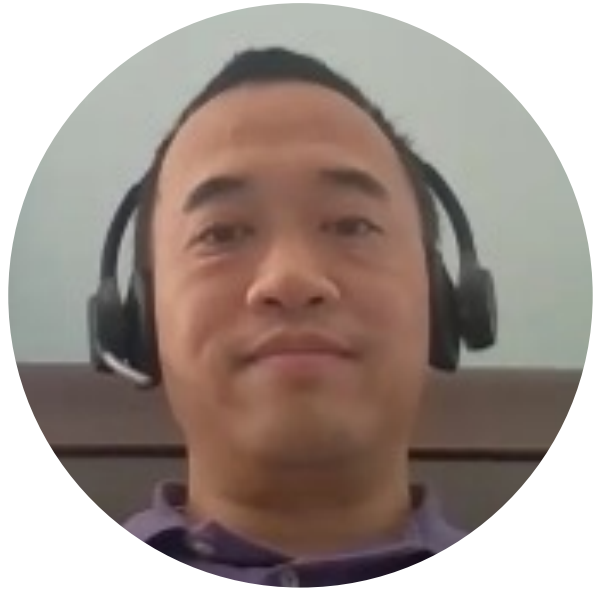




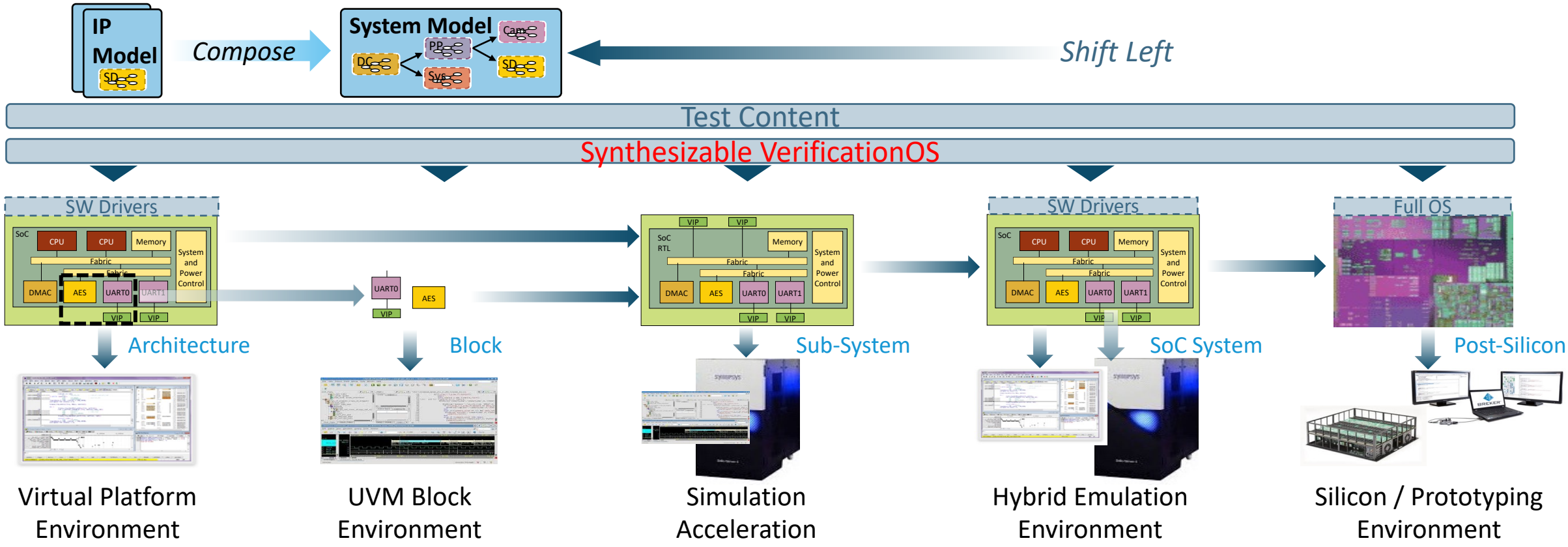
Introduction by Mike Chin, Principal Engineer, Intel

Summarizing Mike's SoC Verification Issues

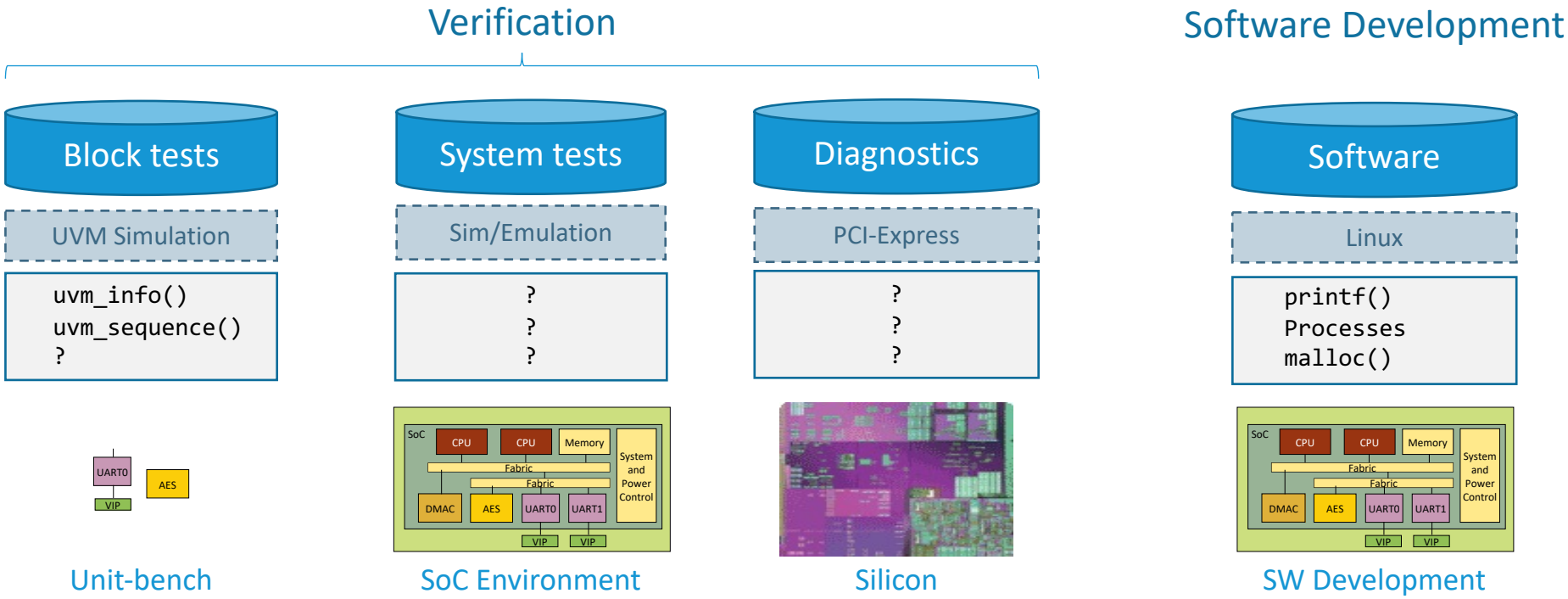
- Driving factor: Shifting test content from IP through SoC to Post-Silicon
- Industry Challenges
  - Time-to-market and the resulting competitive edge
  - High quality equating to validation completeness
  - TTM plus quality equals leadership
- Specific verification challenges
  - Scaling from IP verification through SoC
  - Scaling across verification platforms and OSs, including bare metal
  - High-coverage IP validation on its own and part of an SoC, post silicon validation
  - Talking the same language – especially with third party companies
  - Simplifying problems such as power management
- Synthesizable VerificationOS



Synthesizable VerificationOS
Shift-Left, Reuse, Repetition Elimination, Test Plan Focus



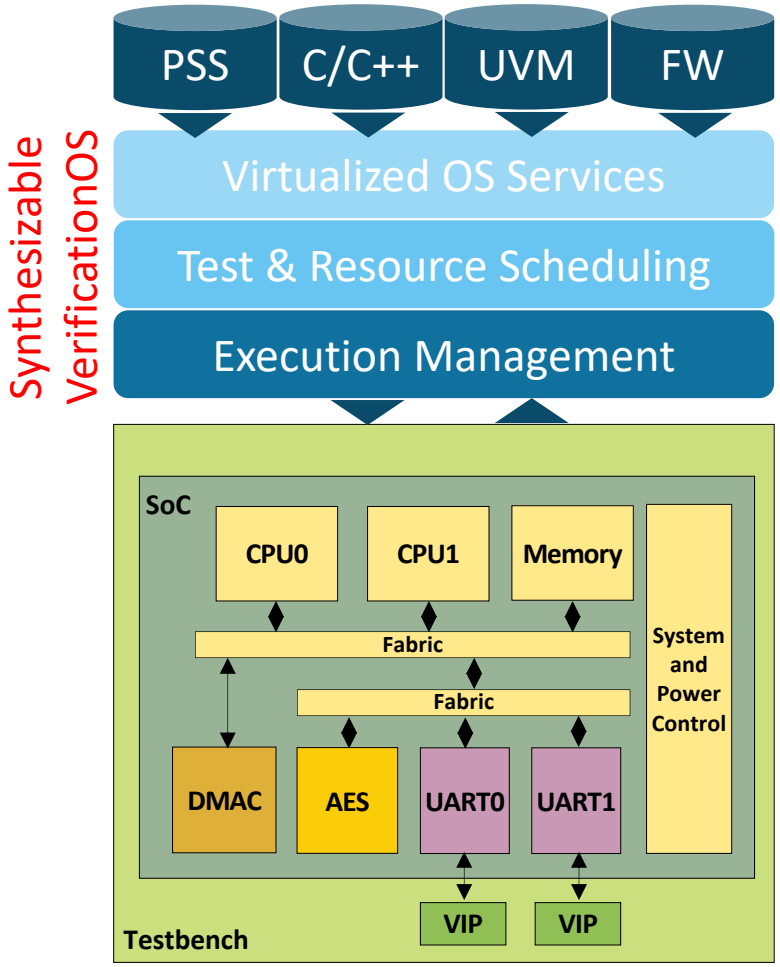
Analogies Across Software and Verification Environments



Synthesizable VerificationOS™ Overview

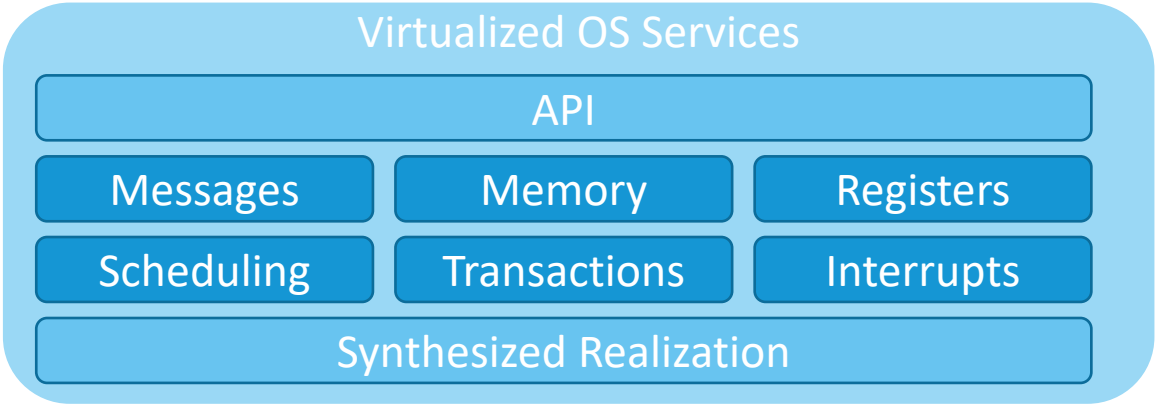
Verification-specific, lightweight OS kernel integrated with test content

- Necessary verification services
- Scheduling tests and resources
- Test synchronization & data execution management
- Multi-lingual/methodology support



Virtualized OS Services

- Portable Messages
- Portable Registers
- Portable Memory Scheduler
- Portable Task Scheduler
- Portable Transactions
- Portable Interrupt Management



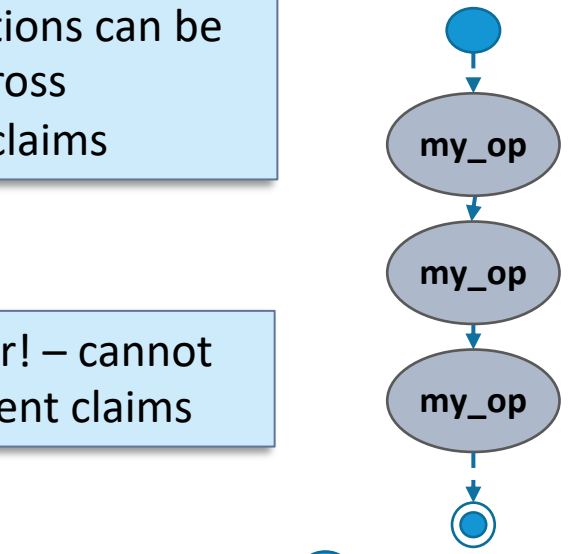
PSS Memory Allocation & Consistency

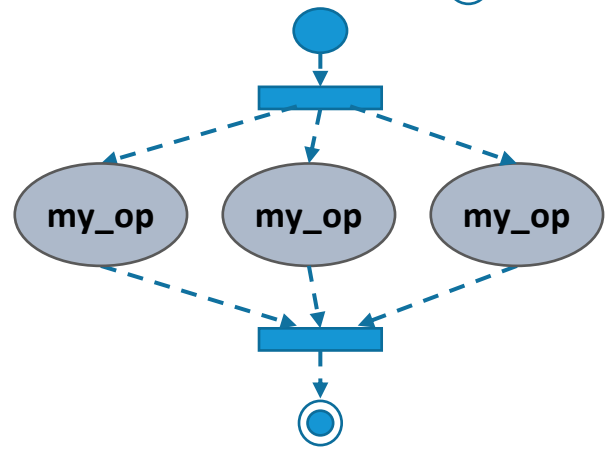
```
component pss_top {
  action my_op {
    rand addr_claim_s<> claim;
    constraint claim.size == 20;
  };

  contiguous_addr_space_c<> mem;

  exec init {
    addr_region_s<> region1, region2;
    region1.size = 50;
    mem.add_region(region1);
    region2.size = 10;
    mem.add_region(region2);
  }
};
```

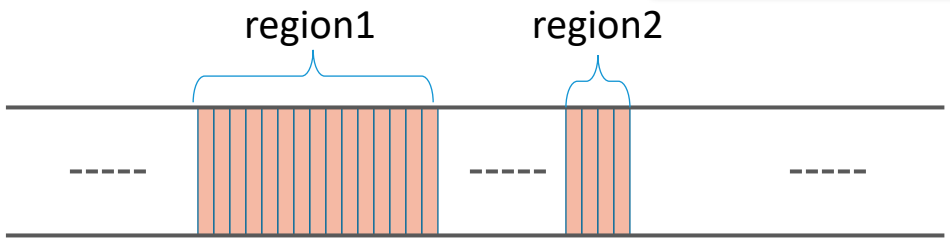
```
action test1 {
  activity {
    repeat (3) {
      do my_op;
    }
  }
};

action test2 {
  activity {
    parallel {
      replicate (3) {
        do my_op;
      }
    }
  }
};
```

OK – allocations can be recycled across sequential claims

Allocation error! – cannot satisfy concurrent claims







Courtesy: PSS Tutorial DVCon 2021

Portable Registers in Native C++

BREKER™

Ipxact register definitions

```
<spirit:register>
        <spirit:name>           UART_LCR                        </spirit:name>
        <spirit:addressOffset>  0x00000003                      </spirit:addressOffset>
        <spirit:size>           8                               </spirit:size>
        <spirit:access>         read-write                      </spirit:access>
        <spirit:description>    Line Control Register           </spirit:description>
        <spirit:reset>
          <spirit:value>        0x03                            </spirit:value>
        </spirit:reset>

        <spirit:field>
          <spirit:name>         CHAR_SIZE                       </spirit:name>
          <spirit:bitOffset>    0                               </spirit:bitOffset>
          <spirit:bitWidth>     2                               </spirit:bitWidth>
          <spirit:access>       read-write                      </spirit:access>
          <spirit:description>  Number of bits in each char     </spirit:description>
                  <spirit:values>
```

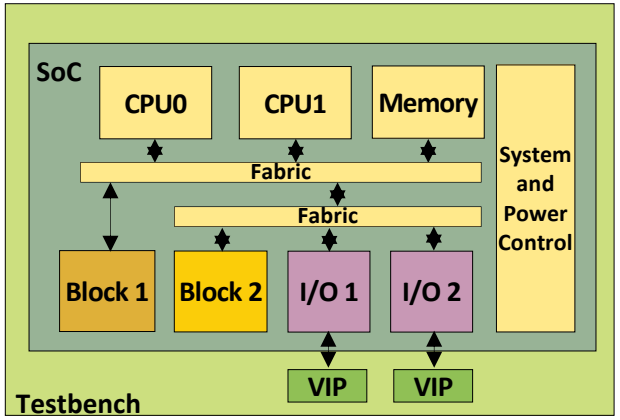
generate

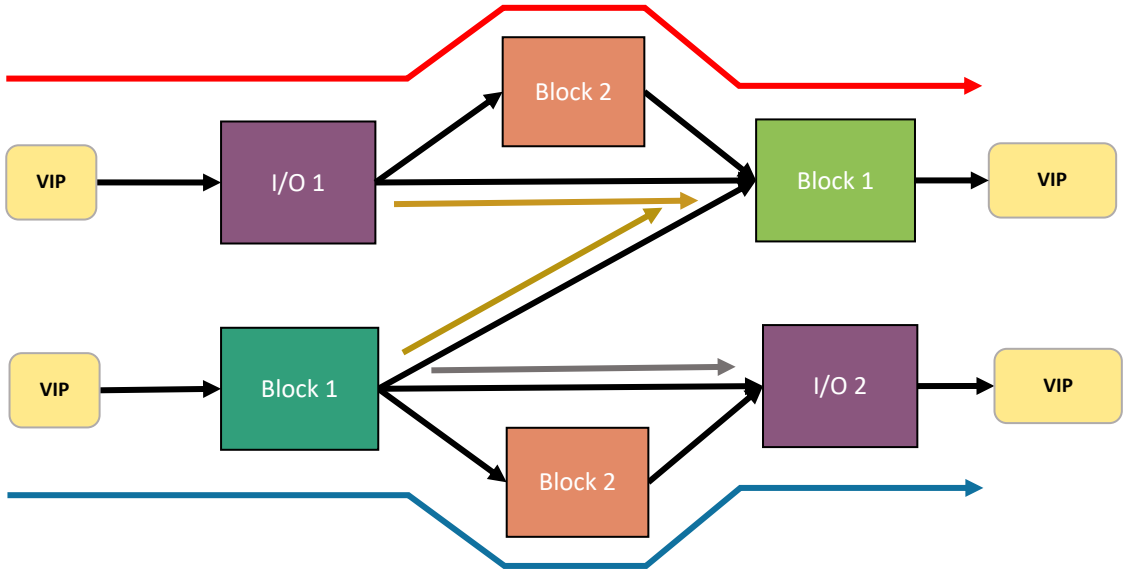
Portable register definitions

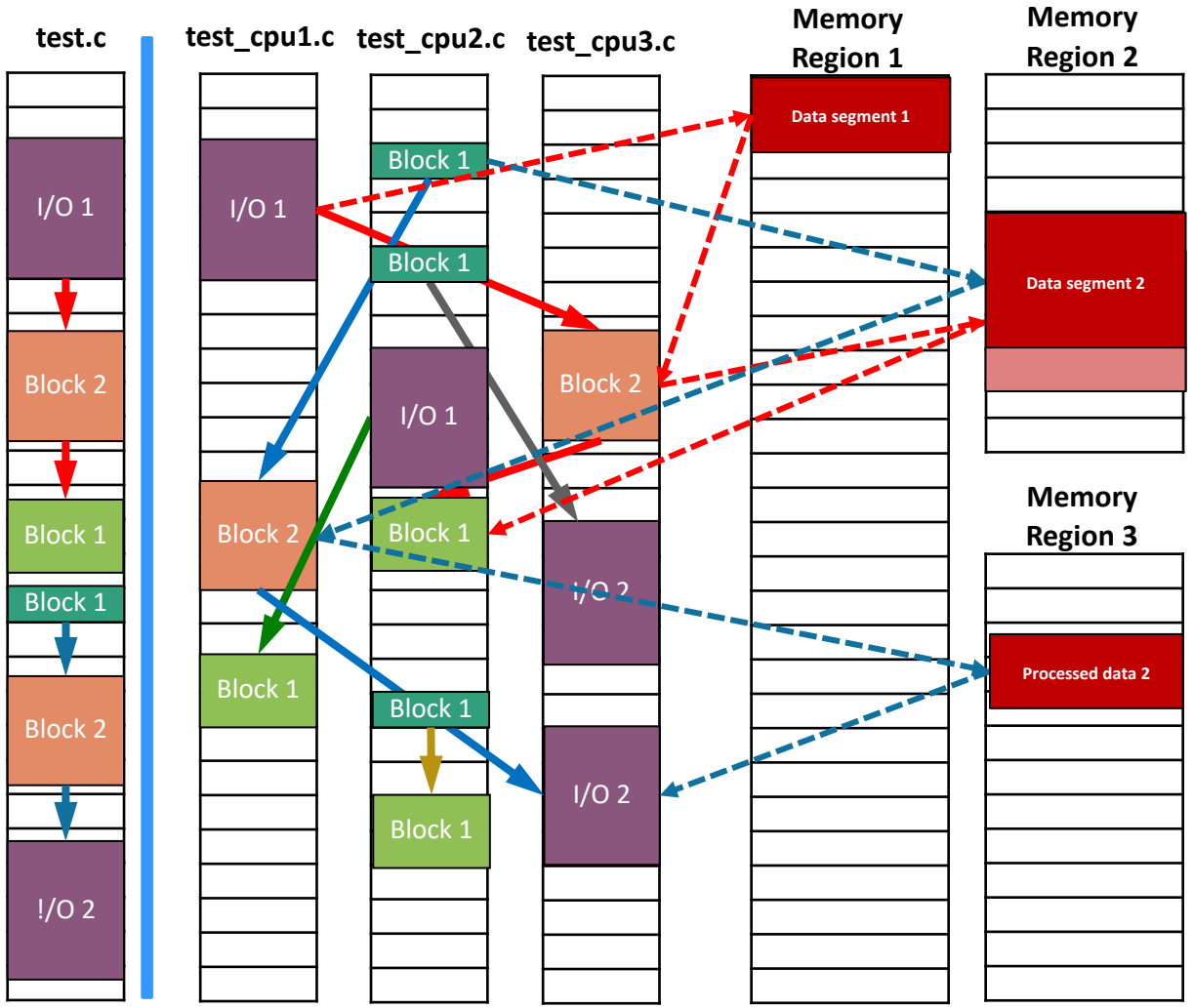
```
class reg_ALL_REGISTERS_MMAP__UART_LCR : public reg {
public:
  reg_ALL_REGISTERS_MMAP__UART_LCR(const scope& s) : reg(this,8) {};
  reg_field CHAR_SIZE { "CHAR_SIZE", 2, 0, "RW", 0, 0, 1, 0, 1 };
  reg_field STOP_BITS { "STOP_BITS", 1, 2, "RW", 0, 0, 1, 0, 1 };
  reg_field PARITY_ENABLE { "PARITY_ENABLE", 1, 3, "RW", 0, 0, 1, 0, 1 };
  reg_field PARITY_EVEN { "PARITY_EVEN", 1, 4, "RW", 0, 0, 1, 0, 1 };
  reg_field PARITY_STICK { "PARITY_STICK", 1, 5, "RW", 0, 0, 1, 0, 1 };
  reg_field BREAK_CONTROL { "BREAK_CONTROL", 1, 6, "RW", 0, 0, 1, 0, 1 };
  reg_field DIVISOR_ACCESS { "DIVISOR_ACCESS", 1, 7, "RW", 0, 0, 1, 0, 1 };
};

class ALL_REGISTERS_MMAP_reg_block : public reg_block {
public:
  ALL_REGISTERS_MMAP_reg_block(… ) {
    …
    ALL_REGISTERS_MMAP.add_reg ( _UART_LCR, 0x00000003 );
  }
};
```

```
// Modeled after UVM registers
blk->regA.fieldF.set(3);
blk->regA.write();
```

Task & Resource Scheduling

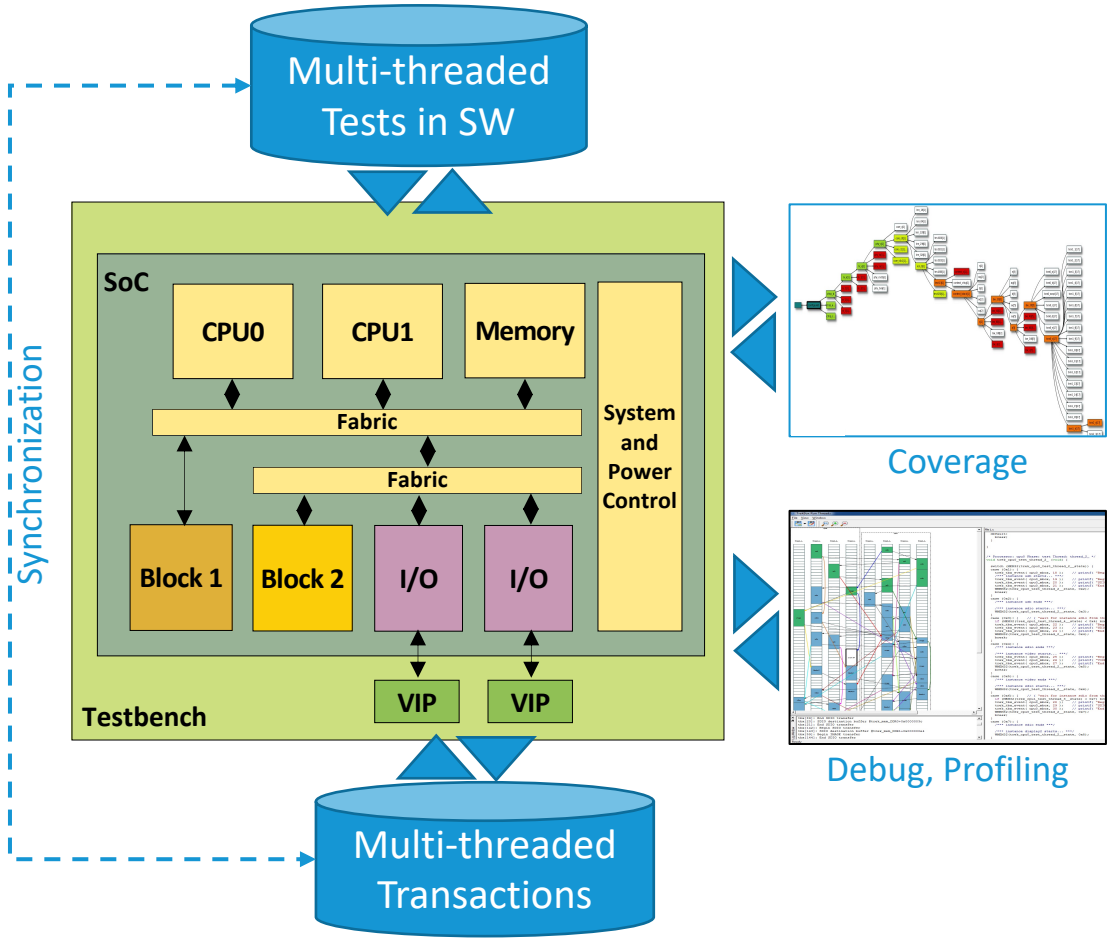






Execution Management

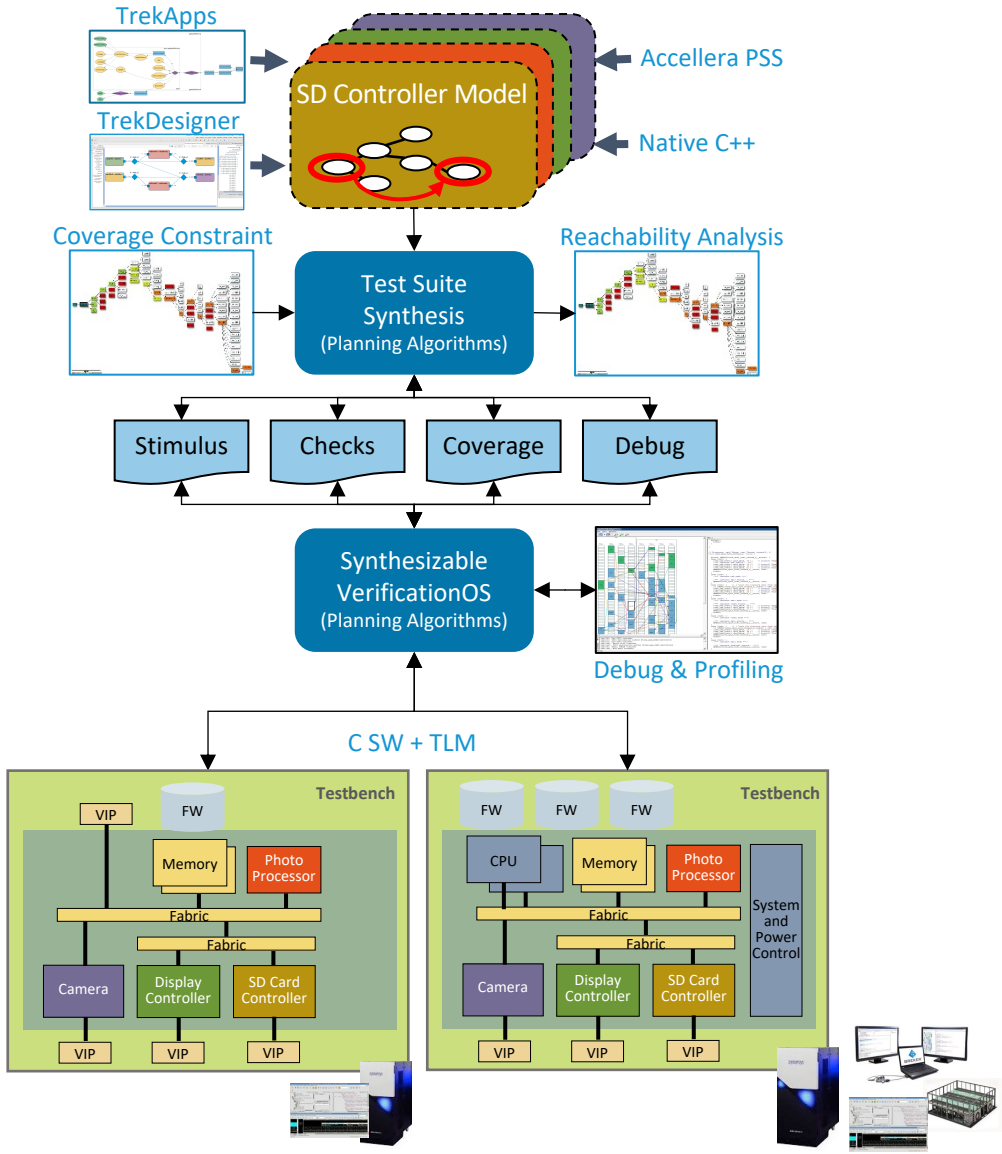
- Multi-threaded test execution
- SW test to I/O transactions synchronization
- Scenario-aware debug
- Coverage driven
- Trickboxing / memory access



Synthesizable VerificationOS for SoC Integration Validation

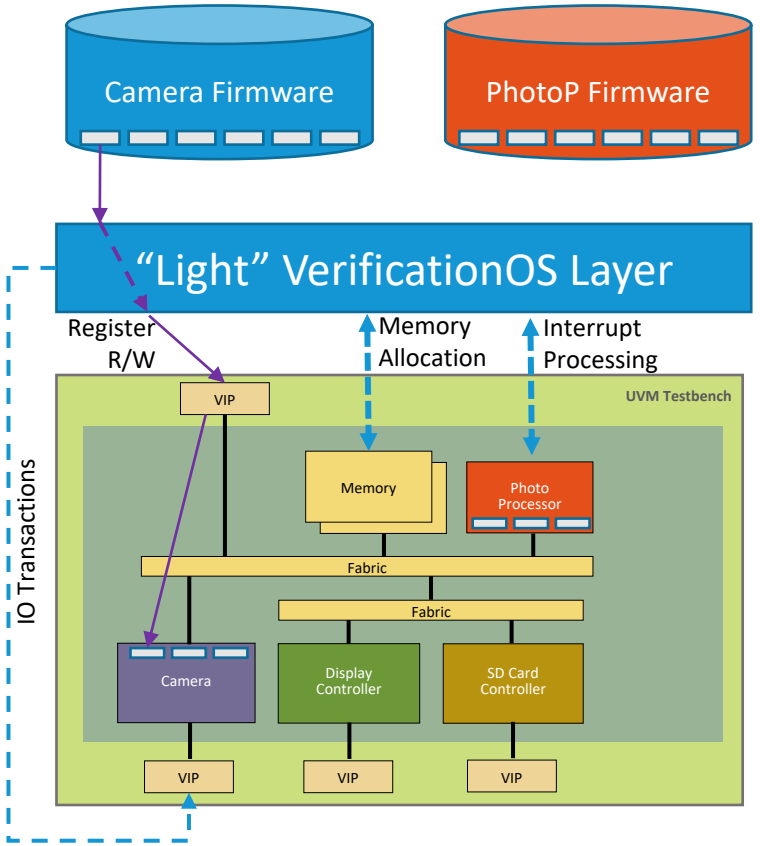
SoC and headless sub-system automated, high-coverage integration verification

- Enable reuse, including pre-packaged configurable scenario ”apps”, porting from UVM and to Post Silicon
- Efficient, high-throughput SoC automated verification for emulation performance, optimized test services
- High coverage, corner-case scenario testing, with SoC specialized debug and profiling



Firmware Verification

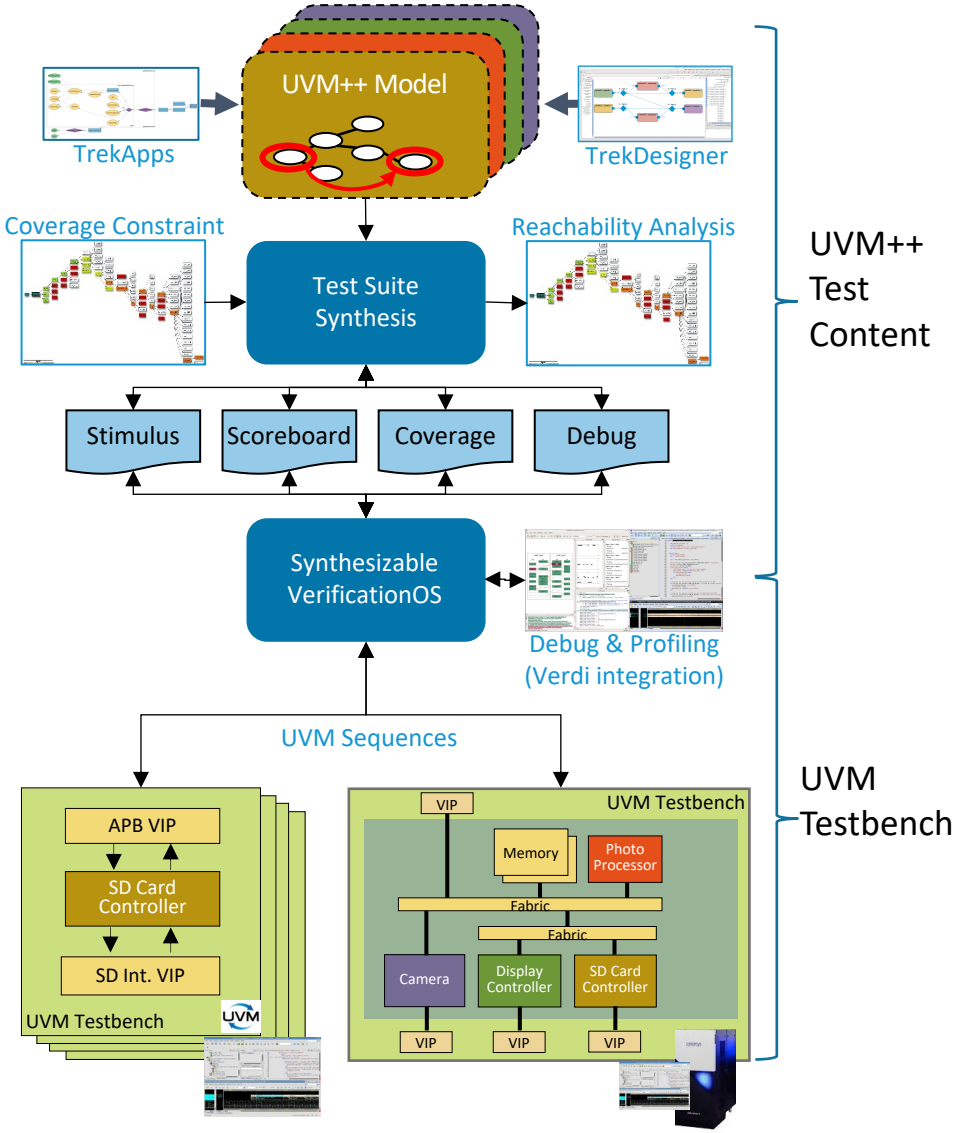
- The VerificationOS provides lightweight firmware execution services
- The VerificationOS allows firmware to execute on subsystems and blocks with or without the processor
- Firmware and the testbench may be combined for a full SoC test on any platform



UVM++ Putting the UVM Engineer In Charge

Test content sequence synthesizer for existing UVM testbenches

- Enables UVM reuse to emulation and prototyping
  - Layer between portable sequences and standard UVM
- Pre-execution coverage selection and closure
- High-throughput use model, built on UVM
  - Random emulation without simulator for accelerated test



UVM IP Testbench Mechanics

Synthesis

test.tbx

UVM environment

```
DPI_LIB      := ${BREKER_HOME2}/build/lib/libtrek.so
INCDIR       += ${BREKER_HOME2}/target/sv
VLOGSRC      +=
${BREKER_HOME2}/target/sv/trek_uvm.sv
SIM_OPTION   += +TREK_TBX_FILE="${TBX_FILE}"
```

UVM sequence detail to interface w/ VIPs

```
class trek_apb_master_seq extends uvm_sequence #(apb_pkg::apb_transfer);
  `uvm_object_utils(trek_apb_master_seq)

  virtual task body();
    forever begin
      req.get(m_tb_path, trek_done);
      if (trek_done) break;
      `uvm_send( req )
      if ( req.direction == APB_READ ) begin
        rsp.send(m_tb_path);
      end
      req.item_done( m_tb_path );
    end
  endtask

endclass
```

Execution Engine

APB VIP

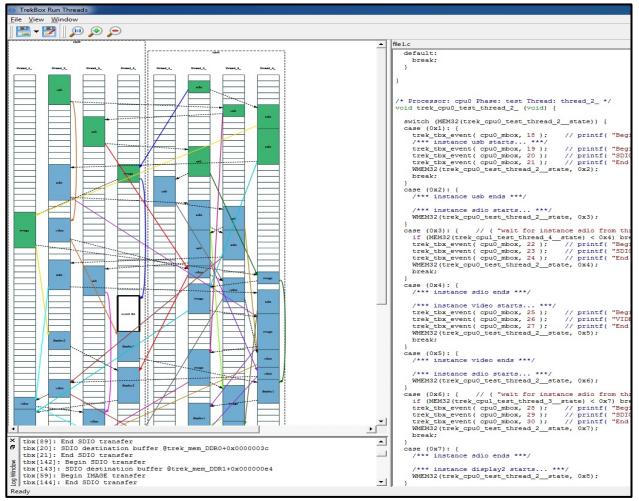
UART RTL

UART0

UART VIP

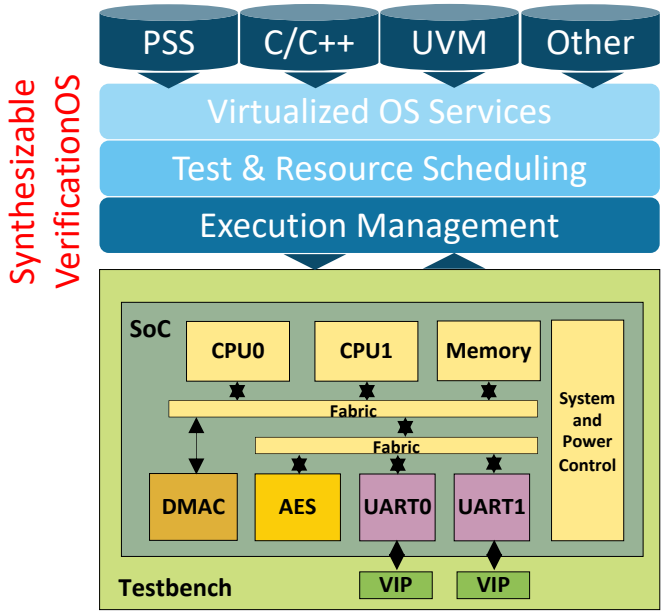
UVM Testbench

ICE / FPGA Prototype / Post-Silicon Diagnostics



Summary

- Evolving SoC verification
- Evolving SoC verification requirements
  Mike Chin, Principal Software Engineer, Intel
- Driving practical SoC verification
  Adnan Hamid, CTO, Breker Verification Systems
- SoC verification requirements and the
  Synthesizable VerificationOS



https://brekersystems.com/soc-verification-and-the-synthesizable-verificationos-workshop/

Thanks for Listening!

For further info: info@brekersystems.com

www.brekersystems.com