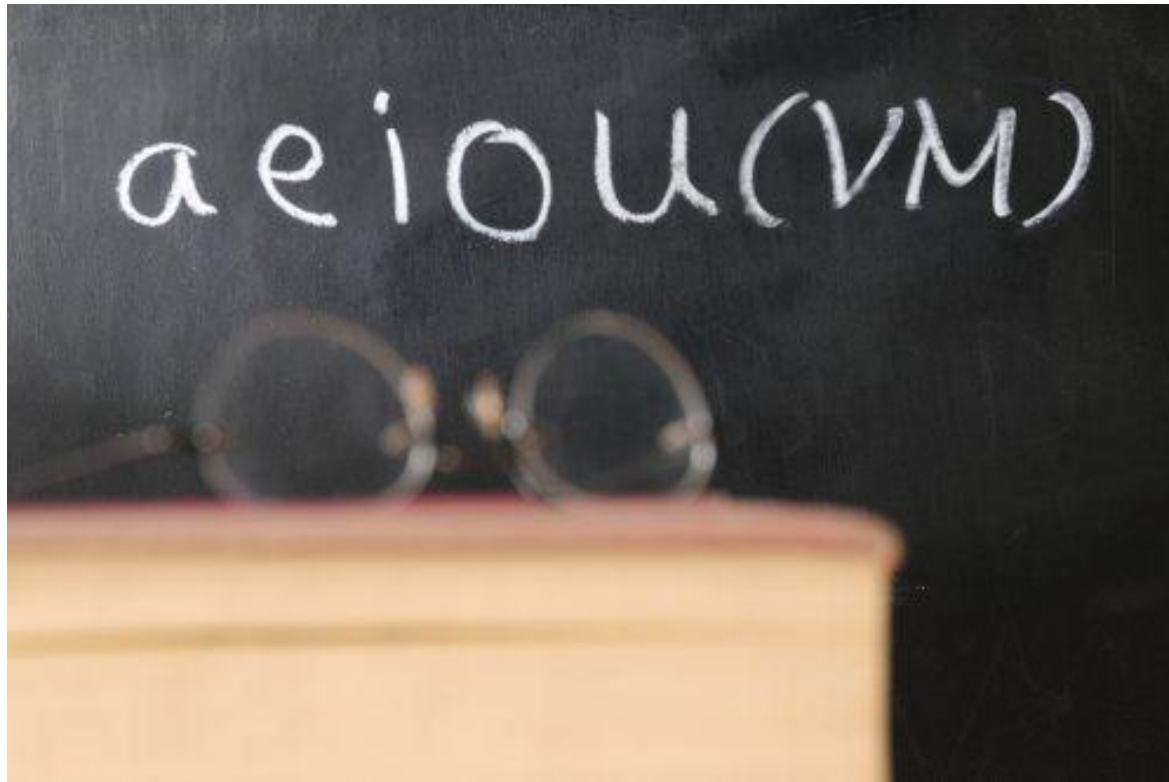




# AEIOU(VM) ANY VOWEL WILL DO



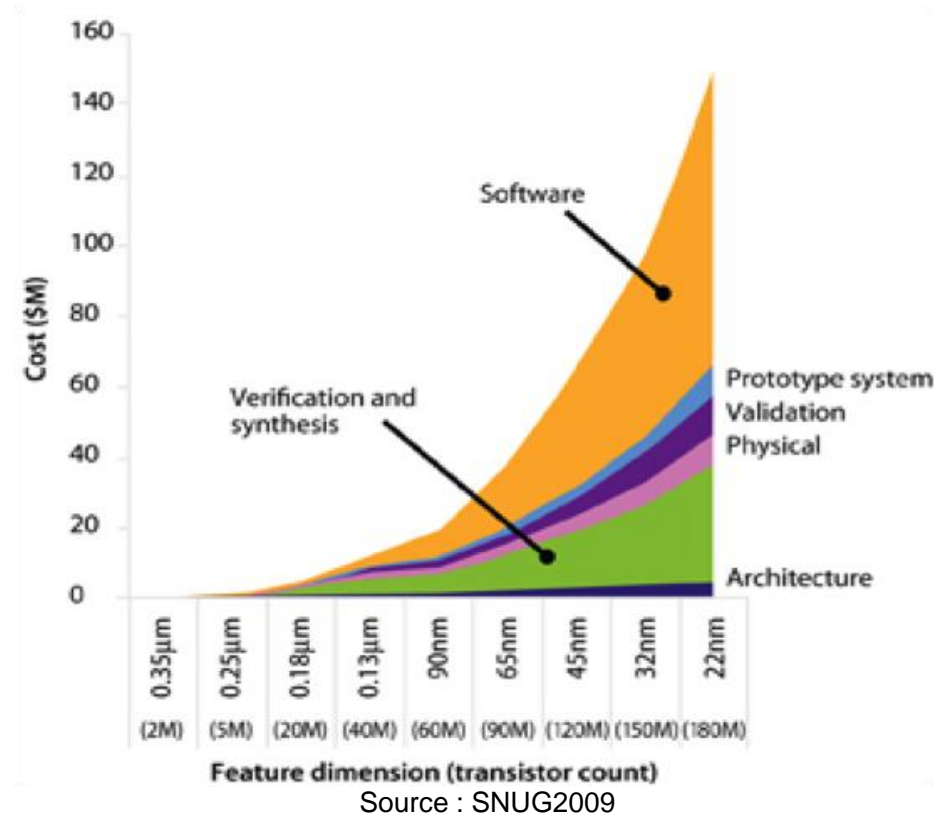
**OVMWORLD Presentation**

**David Murray**  
**CTO**  
**Duolog Technologies**

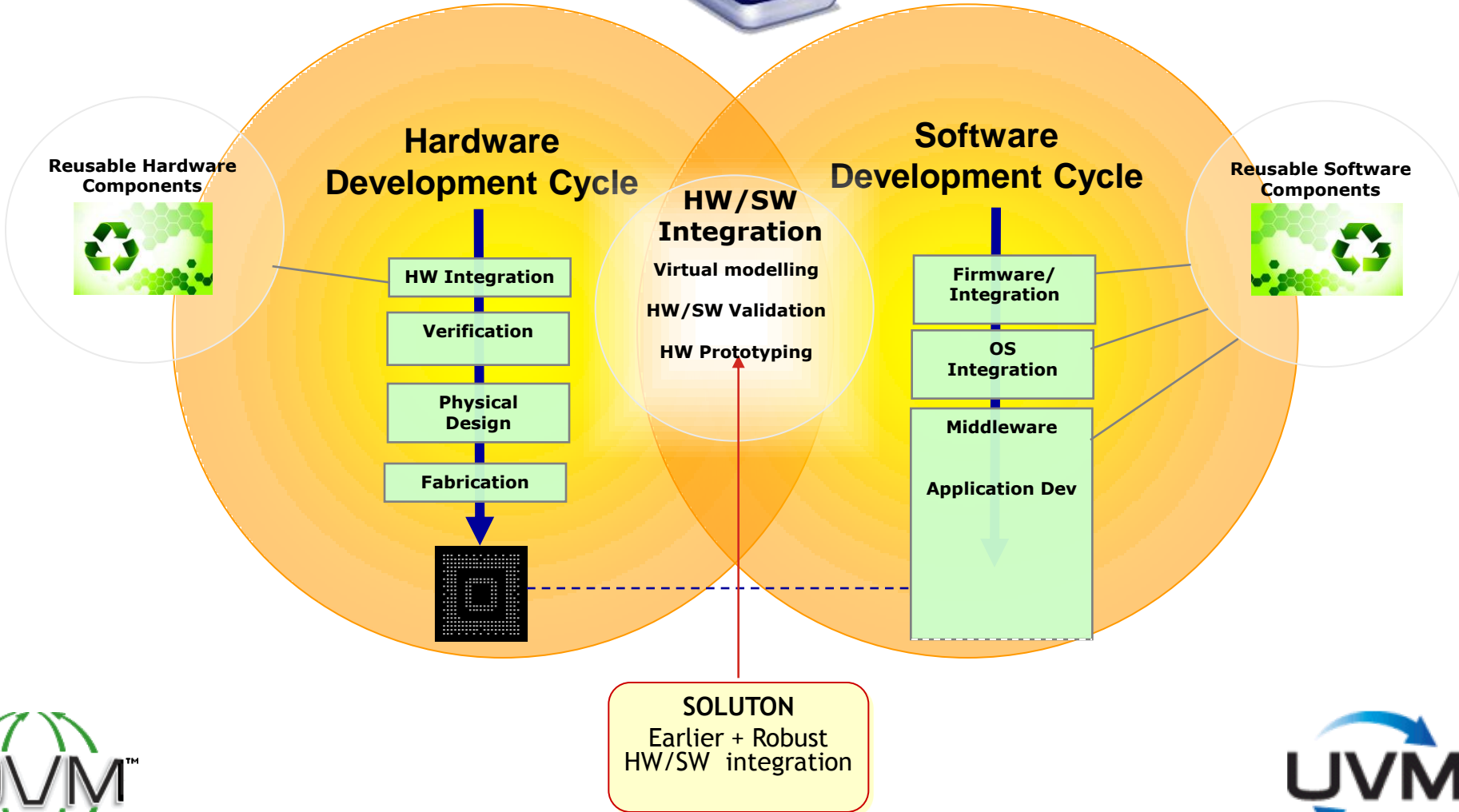


# SoC Design Trends

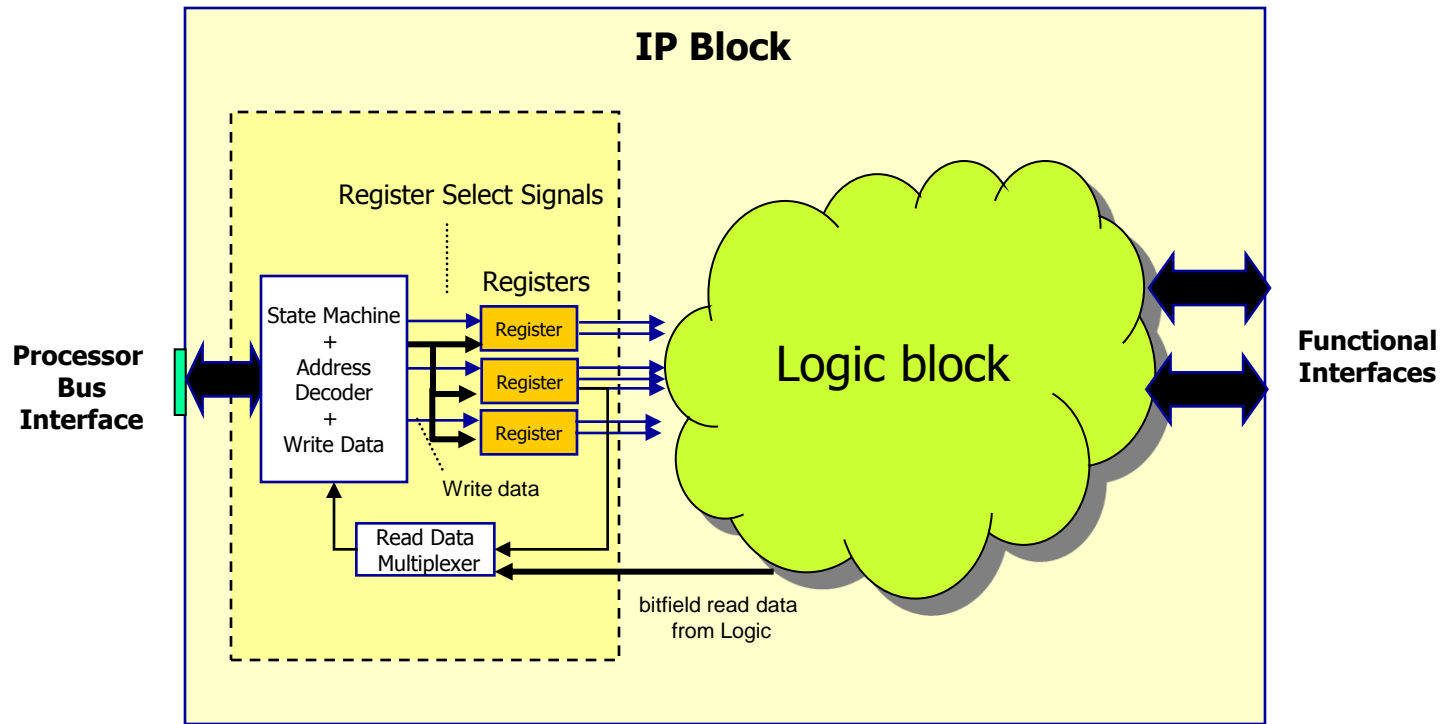
- Exploding Cost & Complexity
  - >3bn transistors
  - Development: \$50m - \$100m
  - Re-spin: \$5-10m + 6 months
  - Advanced verification flows
  - Increasing SW content
- IP-based Design
  - Higher levels of abstraction
  - Extensive IP use & reuse
- Early Software development



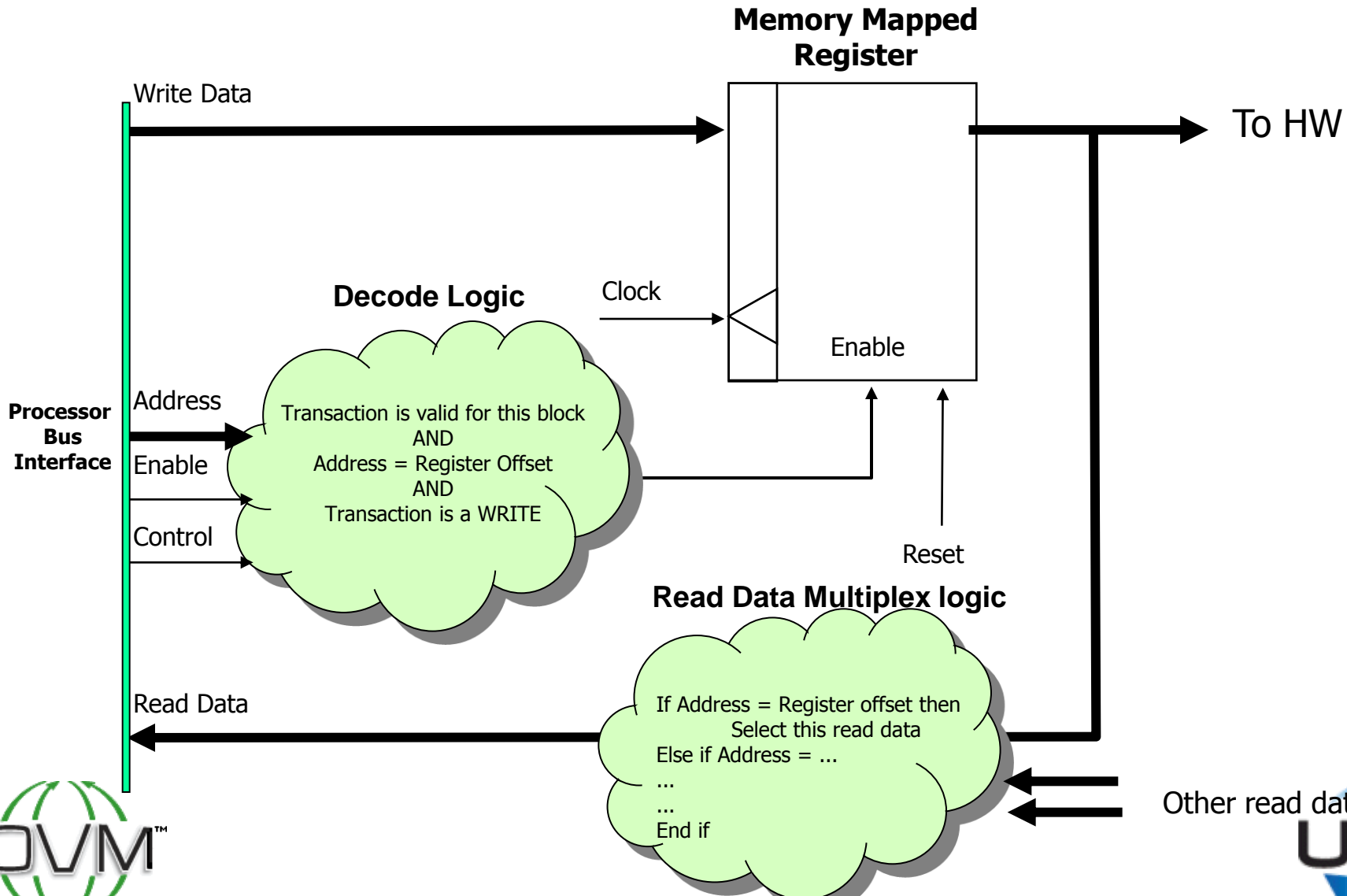
# SoC Integration



# HW/SW Interface (HW View)



# HW/SW Interface (HW View)

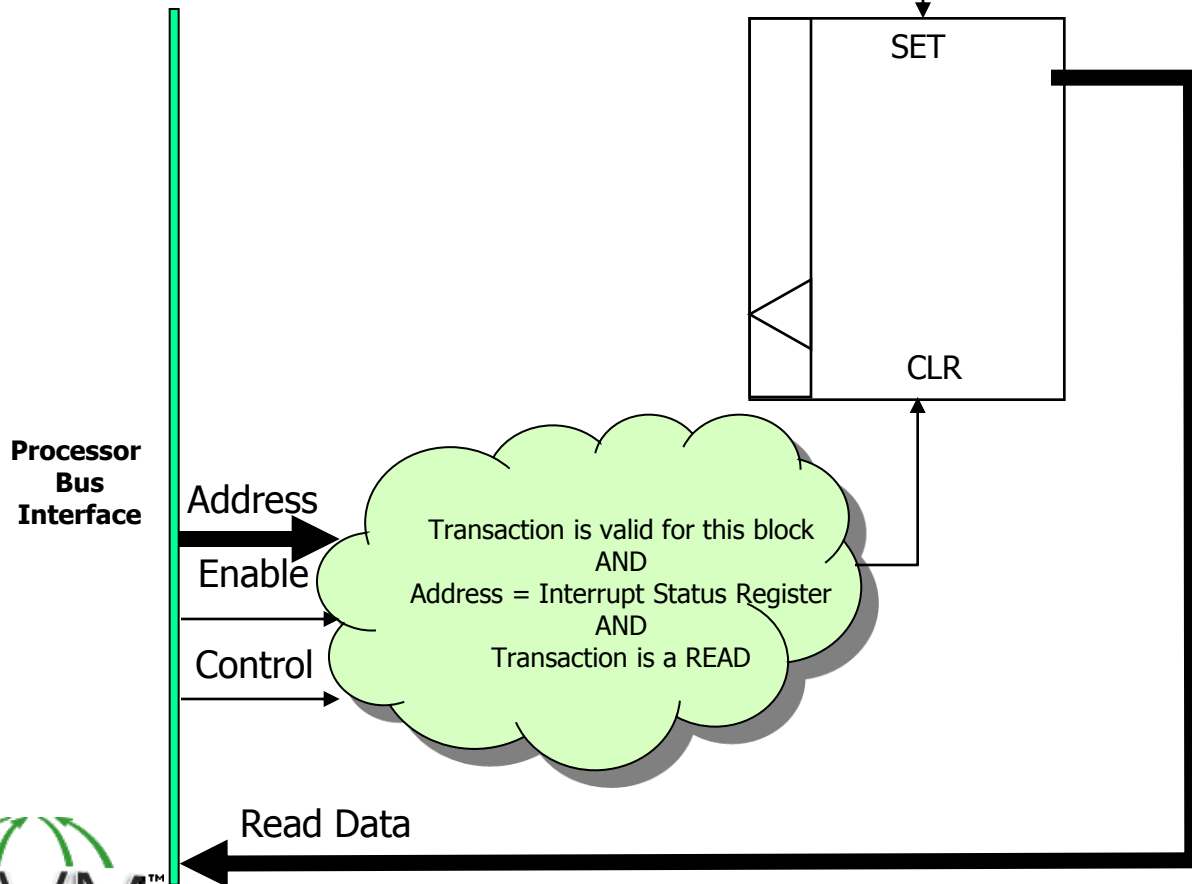


# HW/SW Interface (HW View)

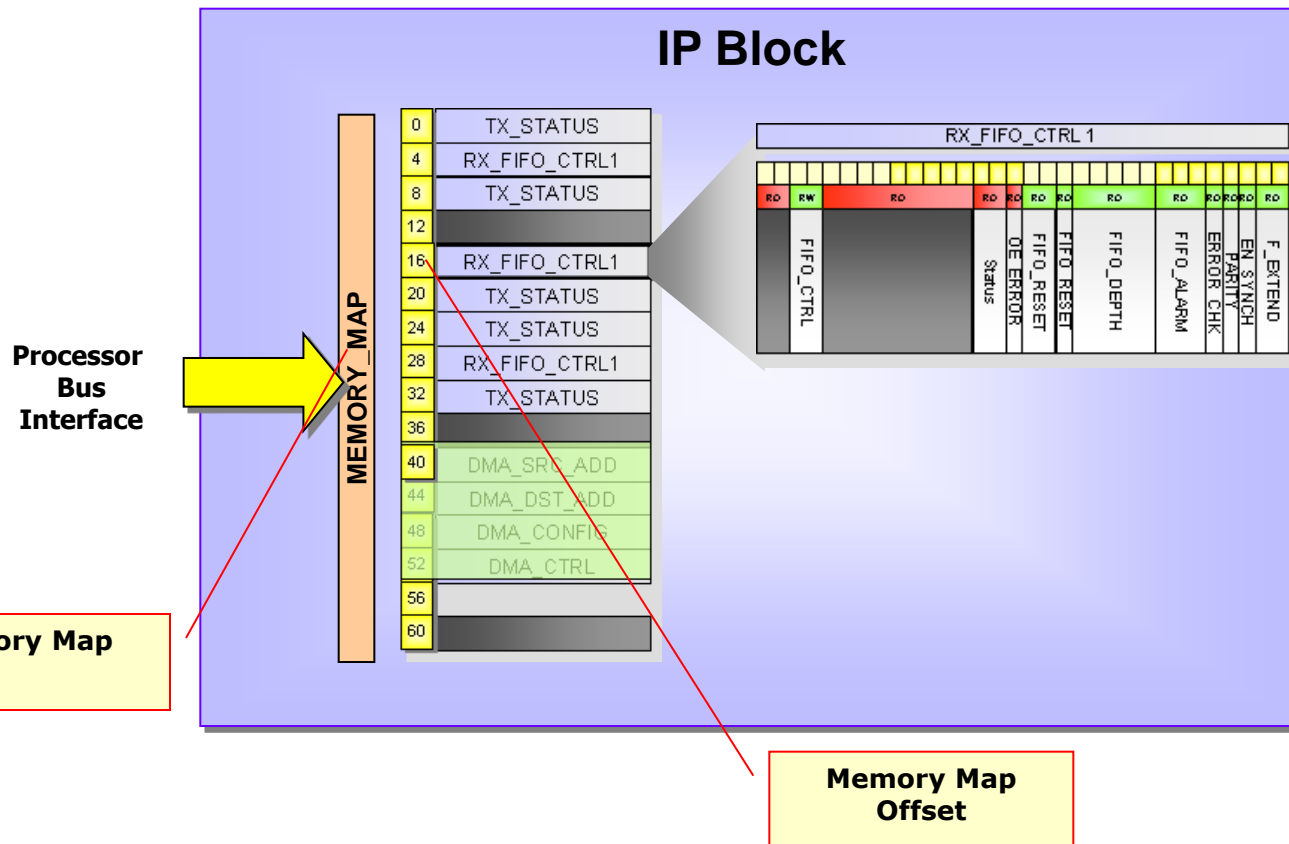


interrupt\_status\_register

Interrupt From HW



# HW/SW Interface (SW View)



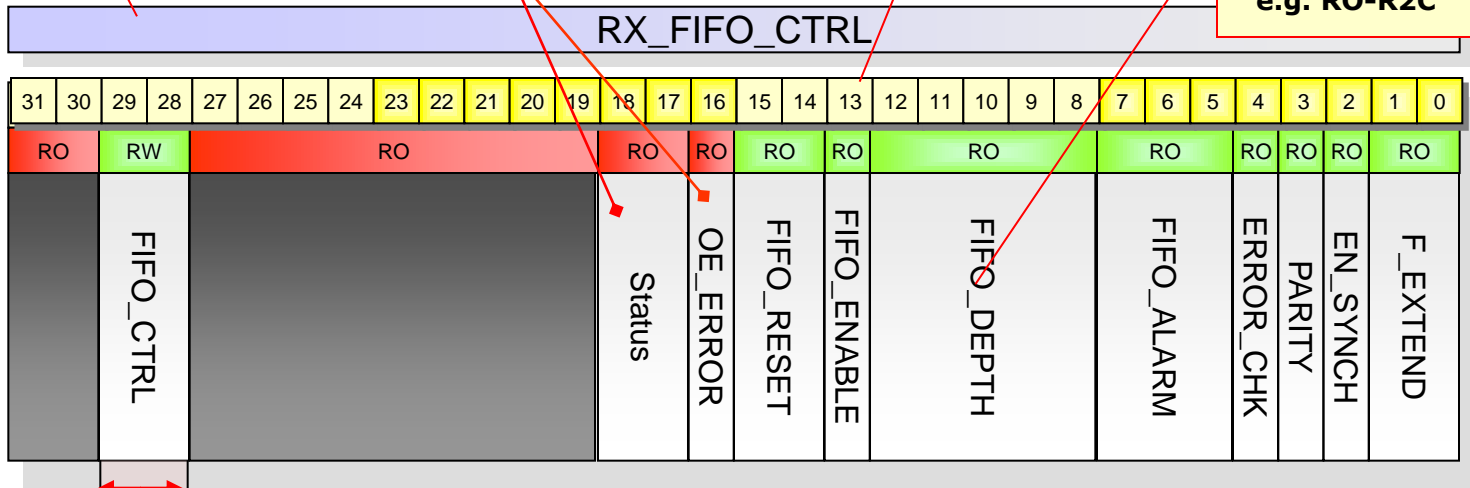
# Register Definition

Register name

Bitfields

Bitfield Offset

Bitfield Access Type  
e.g. RO-R2C



Bitfield Width

0	FIFO_RESET
1	FIFIO_CLEAR
2	FIFO_TX_EN
3	FIFO_RX_EN

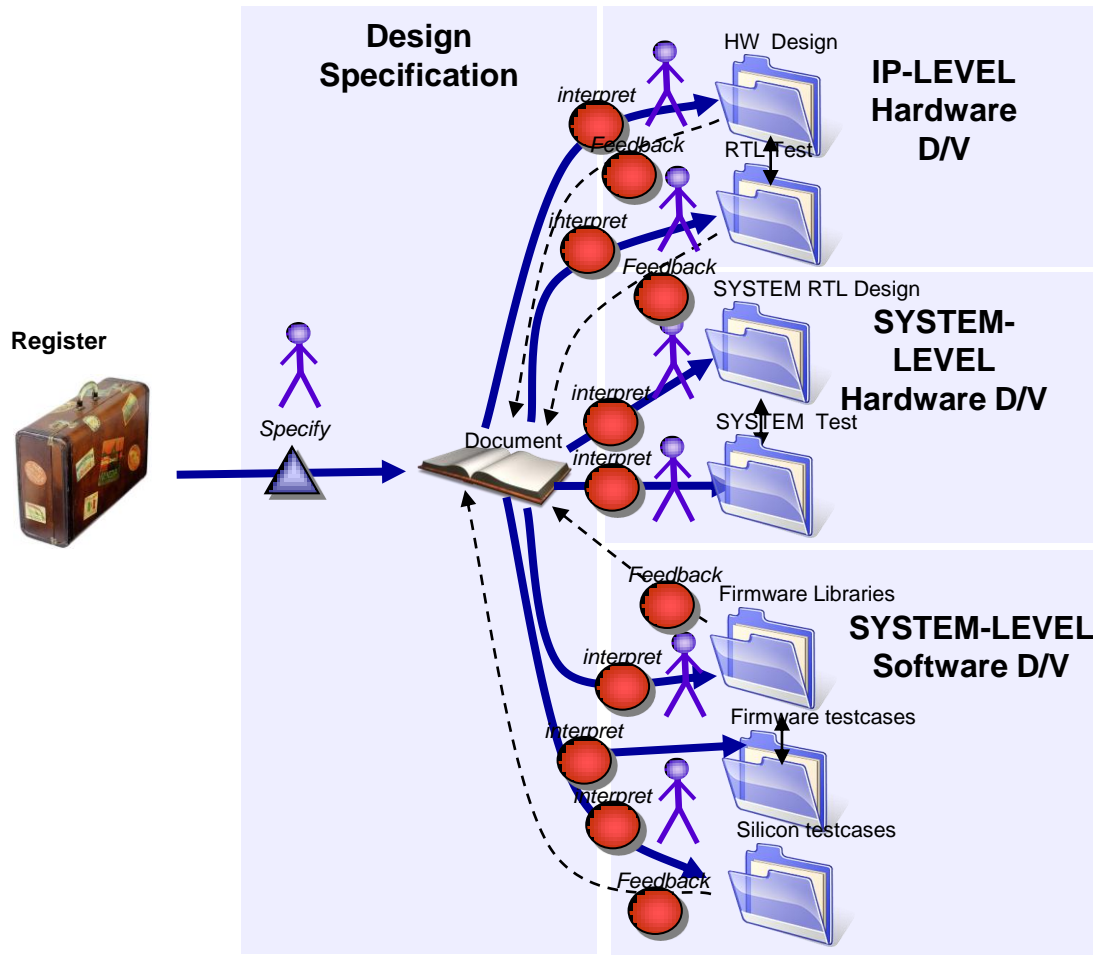
Bitfield Enumerated Values



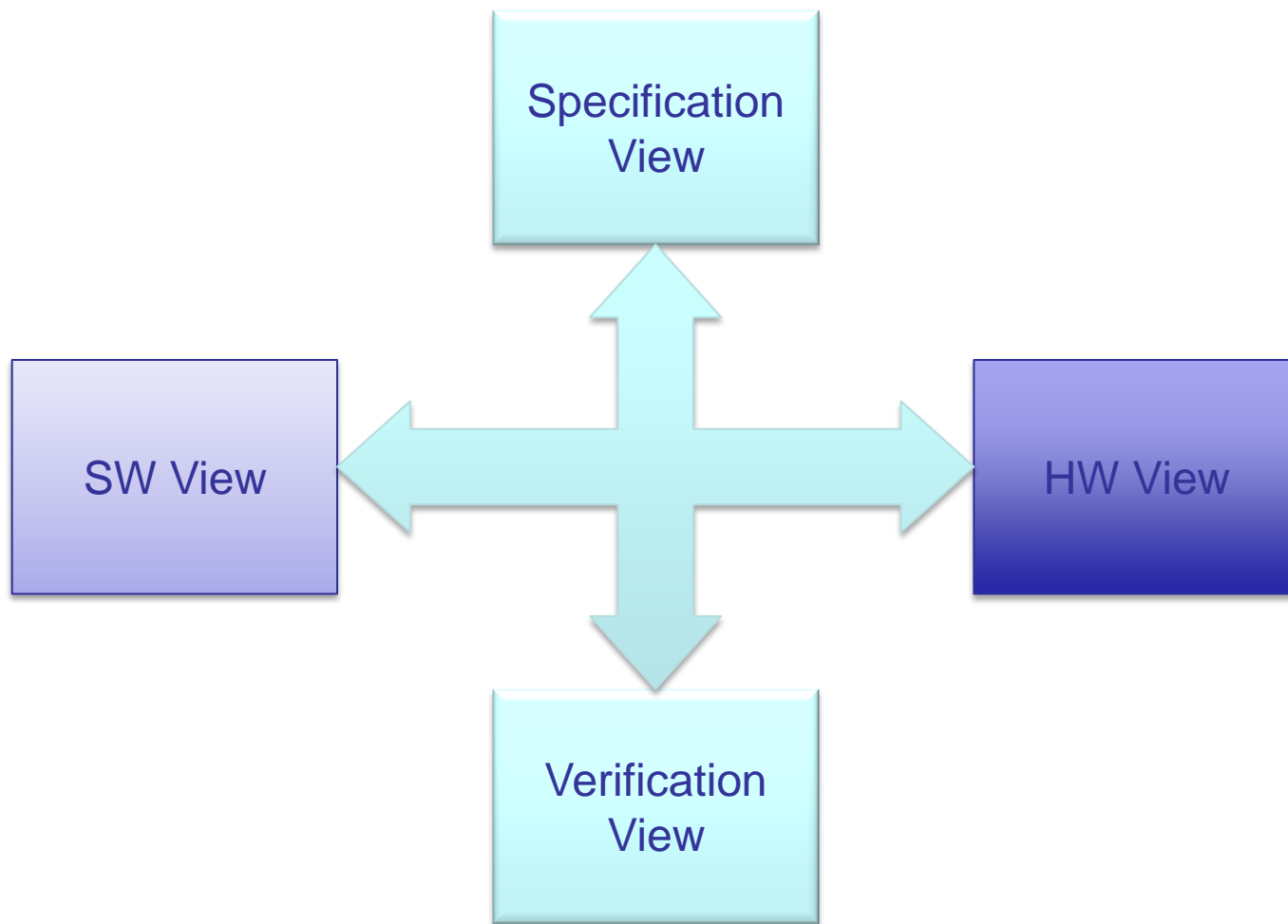




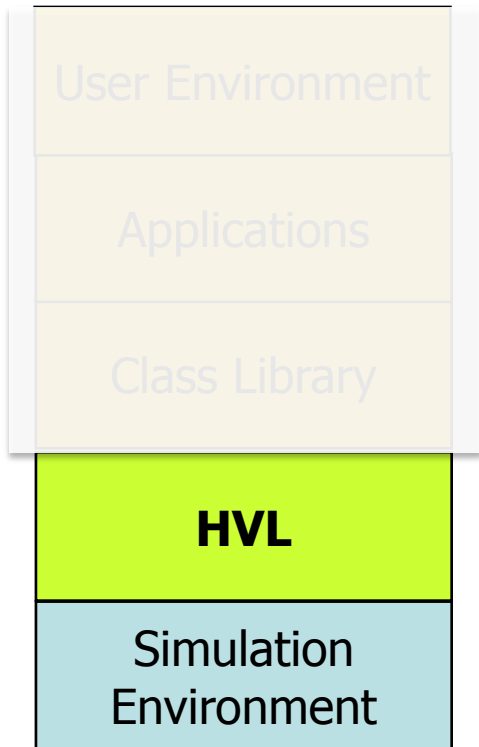
# Compressed Design Cycle



# HW/SW Interface (Verification)



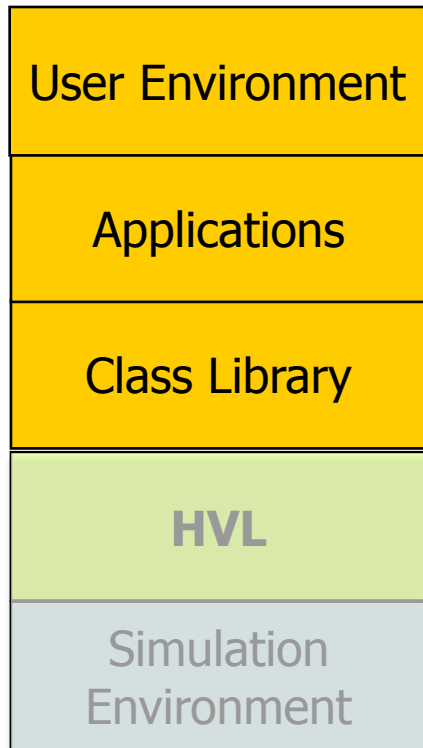
# Verification Evolution



## ■ New Verification thinking

- New verification environment
  - OOV
  - Scoreboard/Checkers
  - Constrained random
  - Coverage driven
  - Monitors/Drivers – TLM
- Emergence of HW verification languages
  - Specman 'e'
  - Vera
  - SystemC/C++
  - SystemVerilog

# High-level Verification Methodology

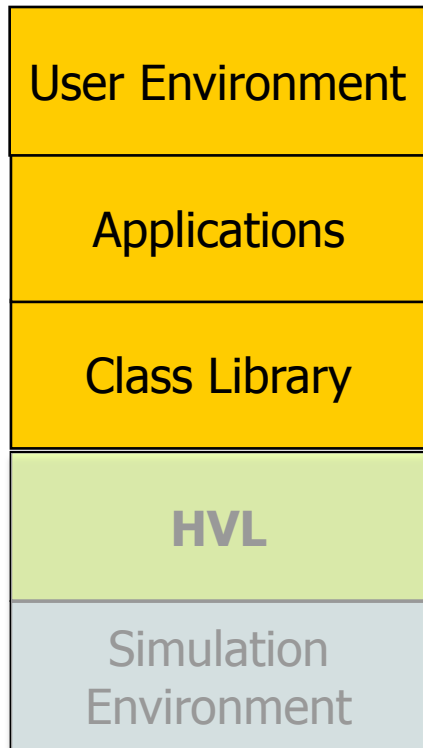


A high-level verification methodology consists of an HVL running in a simulation environment which can interact with the RTL. The HVL's object-oriented infrastructure can be organized and extended to deliver high levels of productivity for specific user applications.

The key mechanisms are **stimulus generation**, **monitoring** and **coverage**.

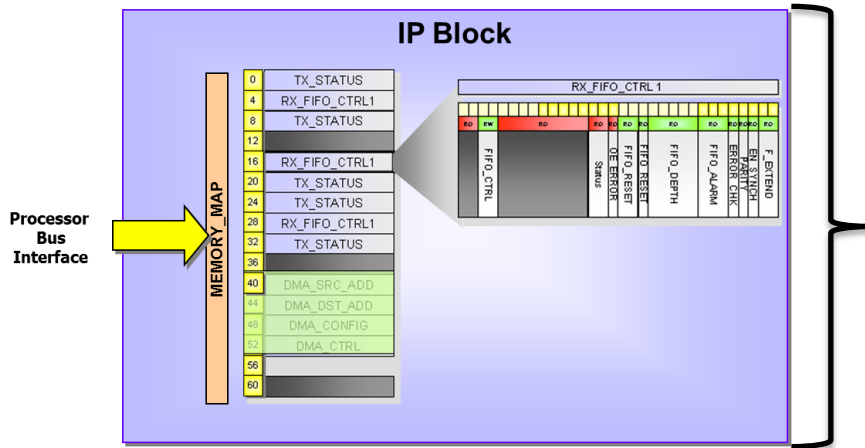
Bailey/Martin 2007

# High-level Verification Methodology



- Evolution of the Verification methodologies (**VM**)
  - A-E-I-O-U(**VM**) where  $I = V$
- Verification IP Reuse
- Application oriented
- User Capability

# HW/SW VM Applications



eRM ↔ VR\_AD

VMM ↔ RAL

OVM ↔ Register Package

UVM ↔ Register Package

- ☺ : Methodologies improve
- ☹ : How to keep aligned ...
  - To new methodologies?
  - Across different flows?



# Standardize-Centralize-Synchronise

## STANDARDIZE

HW Interface : interfaces & ports

SW Interface : registers, memories, memory maps



## CENTRALIZE

Correct Data  
Coherent specification  
Central repository

## SYNCRONIZE

Auto-generated design source

HW/SW ⇔ Design/Verification ⇔ Specification/Implementation

KEEPING TEAMS ALIGNED



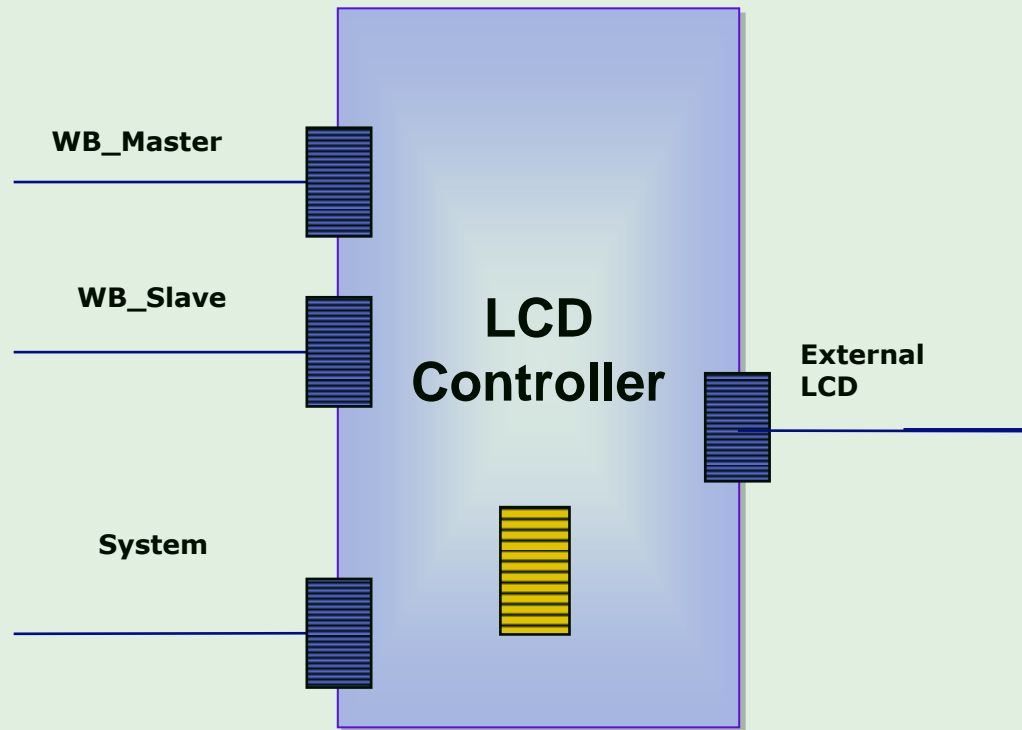
# DEMO

(Using Socrates to Auto-generate any  
OVM/UVM Implementations)

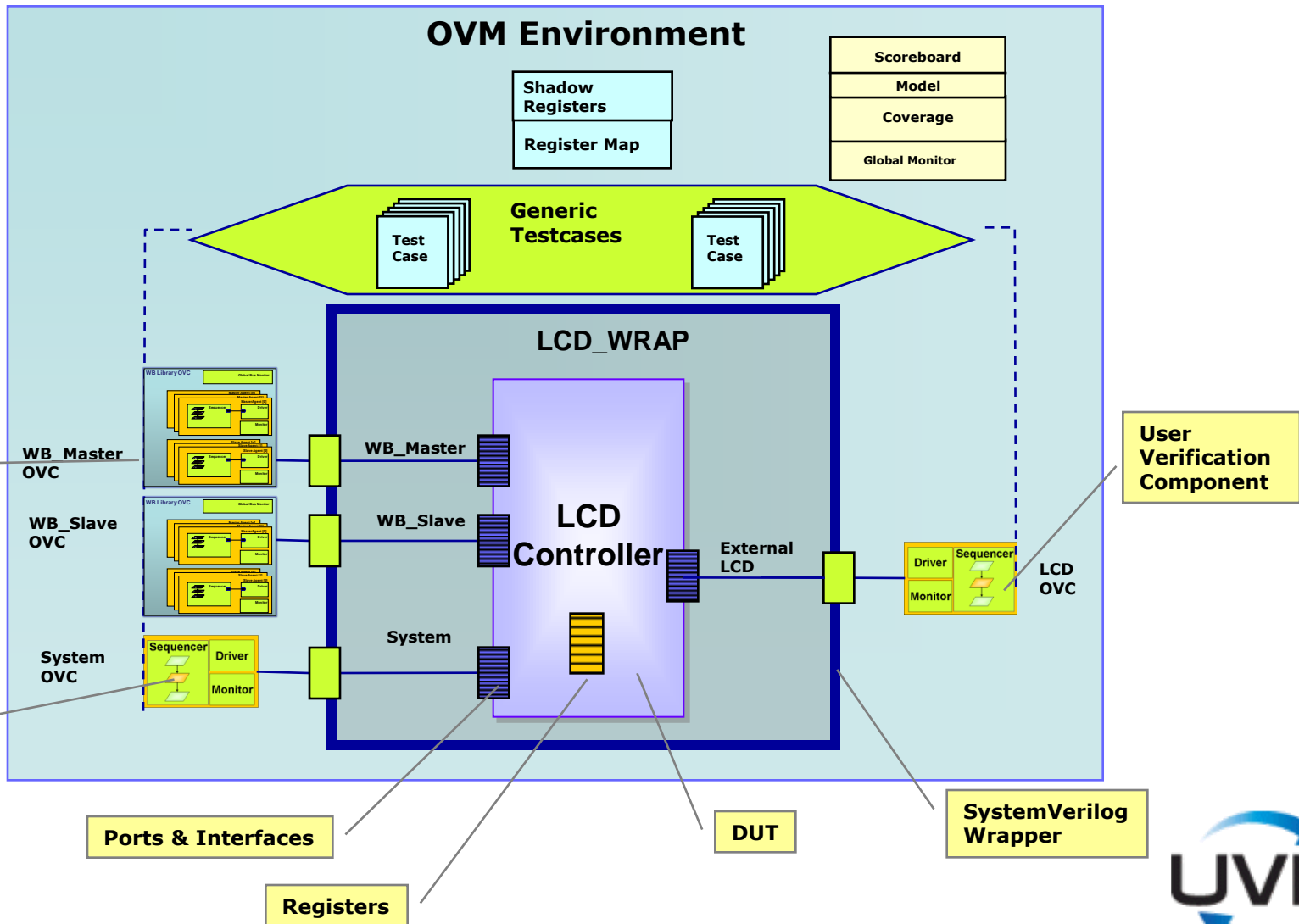




# LCD Controller



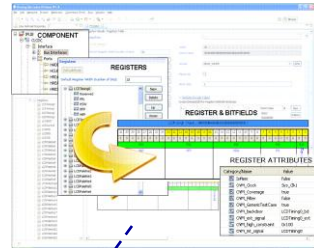
# IP Verification Environment based on OVM



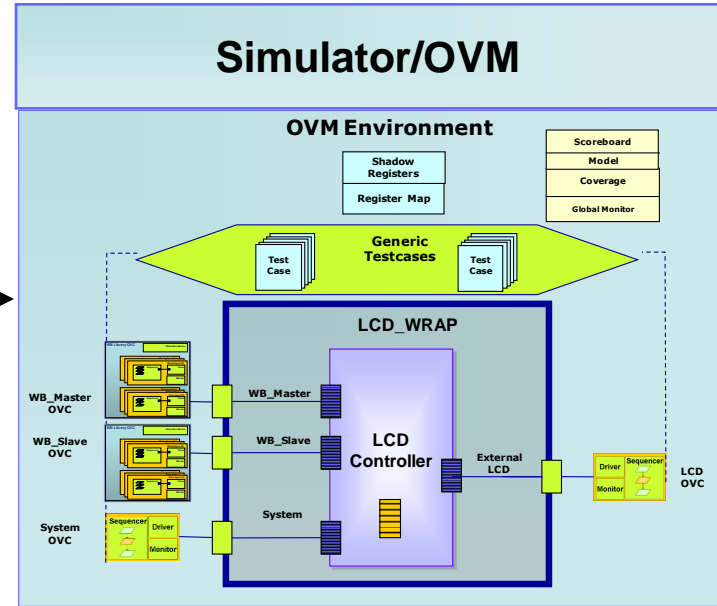
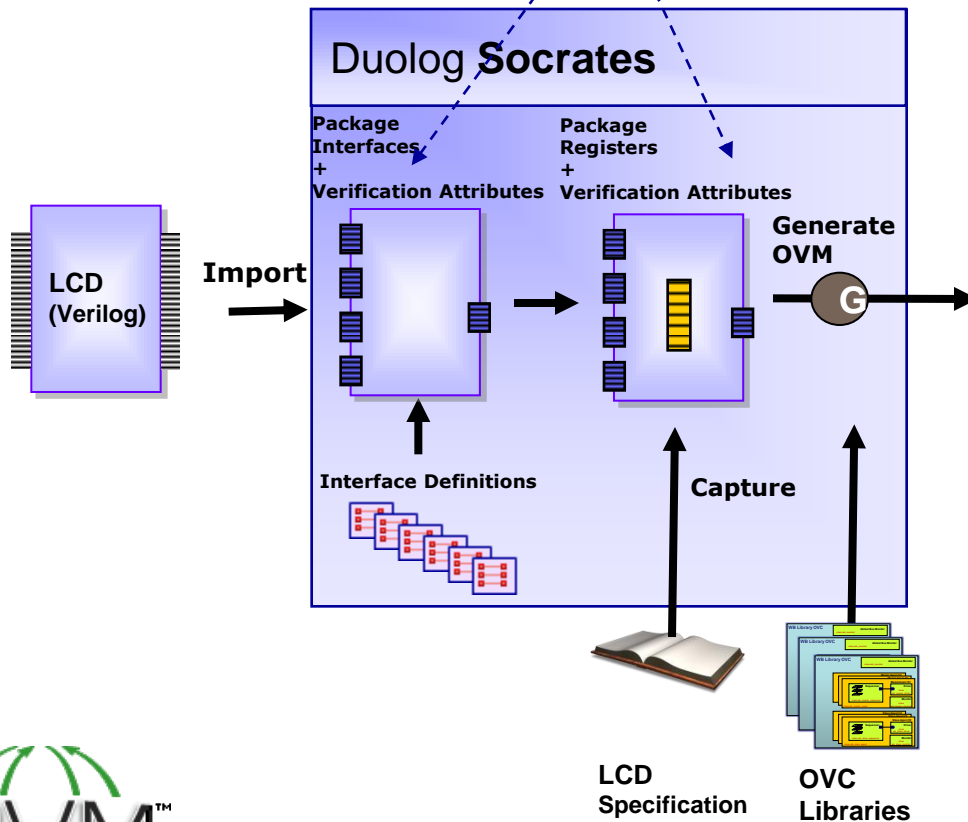
# Auto-generation of IP OVM infrastructure



Bitwise

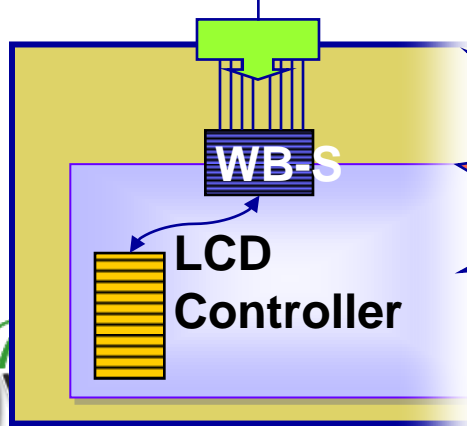
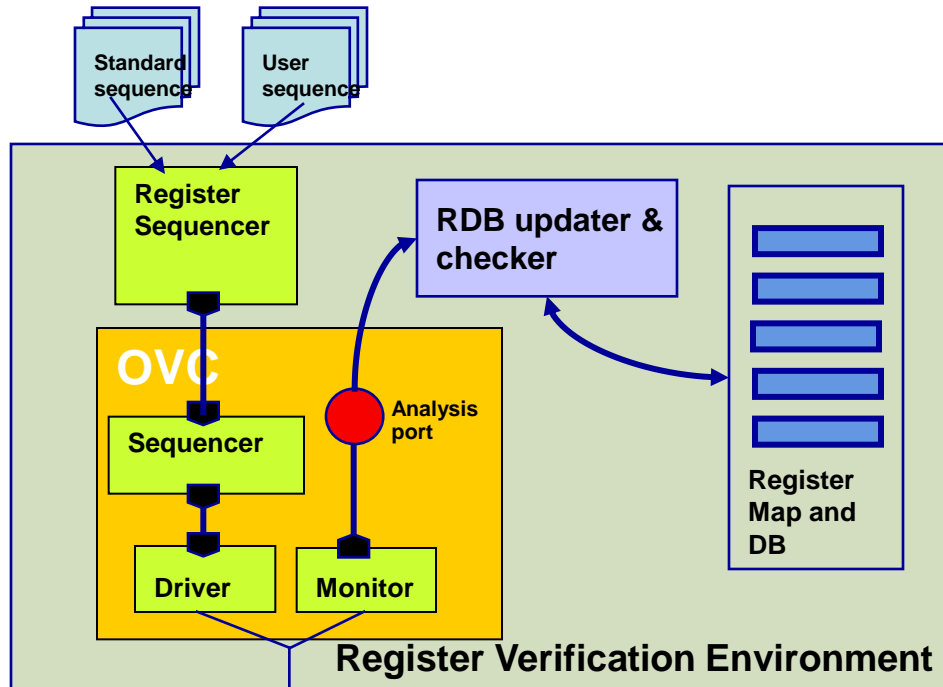


Cadence ncsim/Mentor Questasim



OVM SystemVerilog code & scripts

# Auto-generation of IP OVM infrastructure



100 % autogeneration with the possibility of user extensions

- Register verification is separated from the standard slave OVC:
  - Modular and reusable
  - Bus protocol changes (WB -> APB) doesn't require any changes in the register verification part
- OVC Master sequencer:
  - Sending 32 bit vector sequences to the driver
- Register sequencer:
  - Executing sequences from sequence library (register data)
  - TLM connection to the OVC master sequencer using adaptor sequences
- Monitor sending the read/write data to the analysis port
- RDB checker & updater :
  - Link between bus activity and RDB

# VM attributes

Duolog Socrates-Bitwise V1.8.1

File Edit Navigate Search Project Generator Coherency Check Run Window Help

o v m

- vga\_enh\_top
  - Interface
    - Bus Interface
      - WB\_Slave
        - wbs\_
          - wbs\_
            - wbs\_
              - wbs\_

**COMPONENT**

**REGISTERS**

DefaultMod

Default Register Width (number of bits): 32

- CTRL (0x0, MemoryMap)
- STAT (0x4, MemoryMap)
- HTIM (0x8, MemoryMap)
- Thgate
- Thsync
- VTIM (0xc, MemoryMap)
- HVLEN (0x10, MemoryMap)
- PARA

**Interface Details**

Set the properties of the selected interface.

Interface: WB\_Slave

Width: 32

Address Range: 0x1000

Memory Map: MemoryMap

Access Sizes: 8, 16, 32

WB Slave Bus Interface: processor interface

Memory Maps Modes

**REGISTER & BITFIELDS**

HTIM [Reset : 00000000000000000000000000000000]

29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RW											RW						RW													
Thsync											Thgdel						Thgate													

Category/Name	Value
OVM_bf_collect_coverage	cov_all
OVM_bf_constraint	valid_range {value.Thgate < 1024;}
OVM_bf_external_event	
OVM_bf_hdl_path	thgate
OVM_bf_type	



# Conclusion

- HW/SW interface requires critical verification attention
- HW/SW interface modelling is key to synchronizing teams
- HW/SW interface modelling is key to allowing smoother VM transition
- Demoed Auto-generation of OVM/UVM from a single source
- Looking forward to the next transition

