DVCon 2012

Design & Verification Conference & Exhibition

February 28 – March 1, 2012

Graph-IC Verification

by

Gregory FAUX
Verification Engineer
STMicroelectronics

Dennis RAMAEKERS
Verification Engineer
ST-Ericsson

Hello!

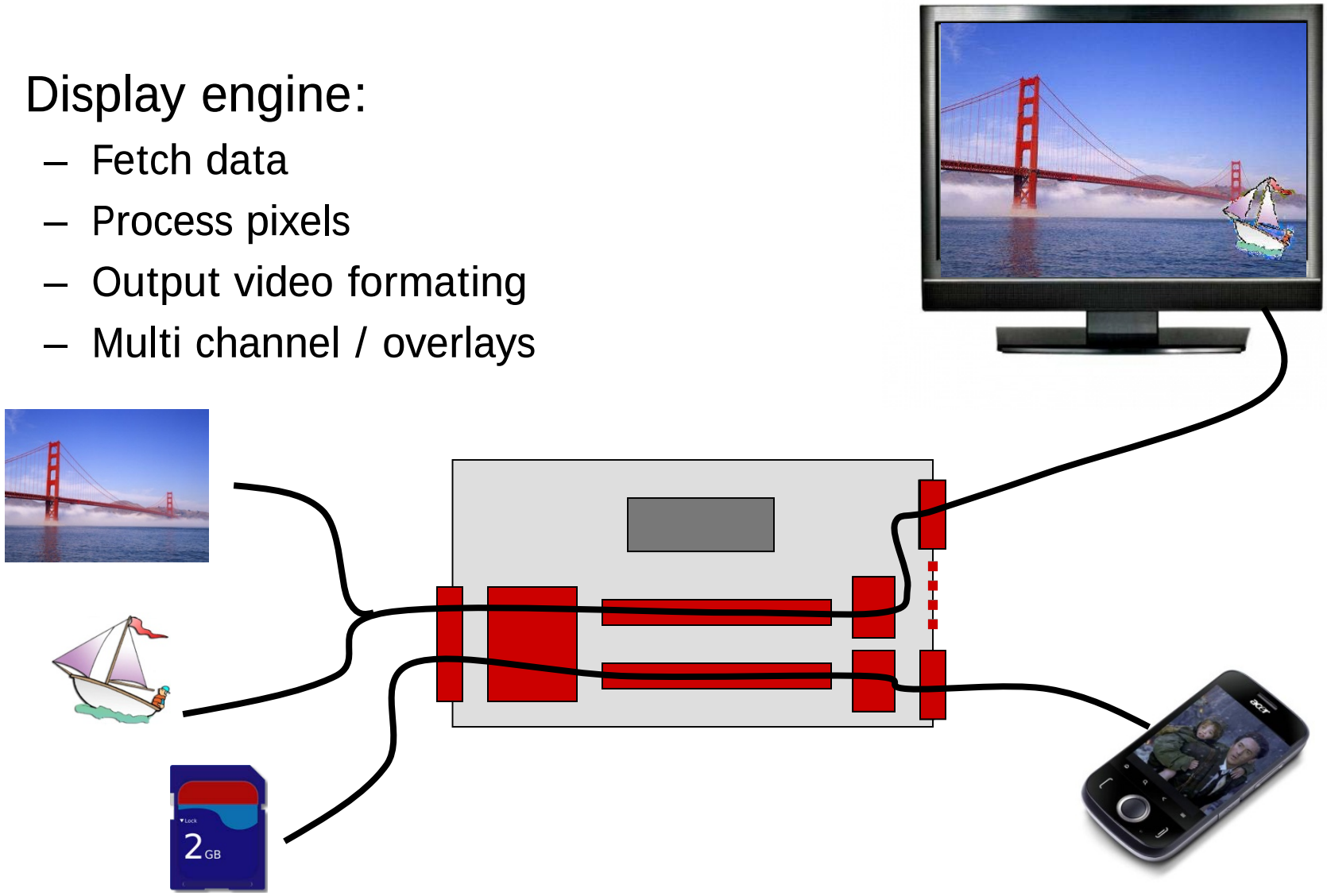
- Why are we here?
  - Report of our experience in verifying a complex and highly configurable IP

Our problems might be yours

GRAPH-IC

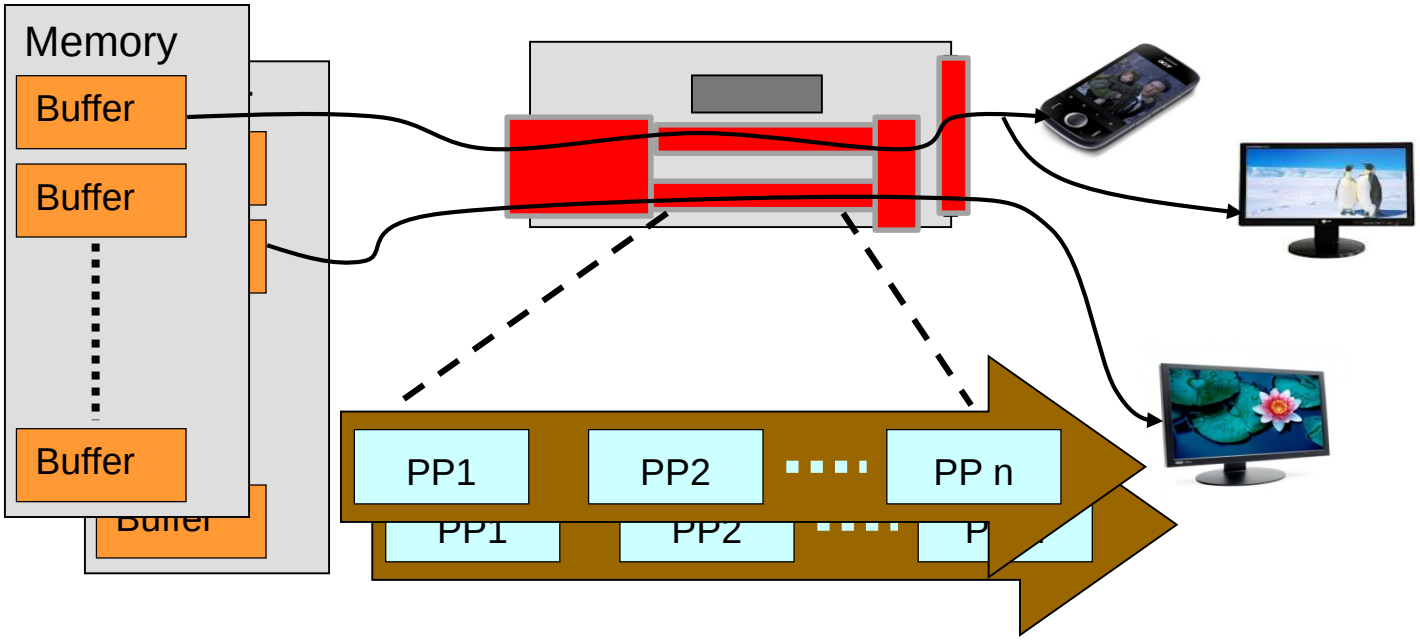
DUT Overview

- Display engine:
  - Fetch data
  - Process pixels
  - Output video formating
  - Multi channel / overlays



Where Complexity Explodes…

- Each stage presents high configurability
- High flexibility achieved by decoupling each stage

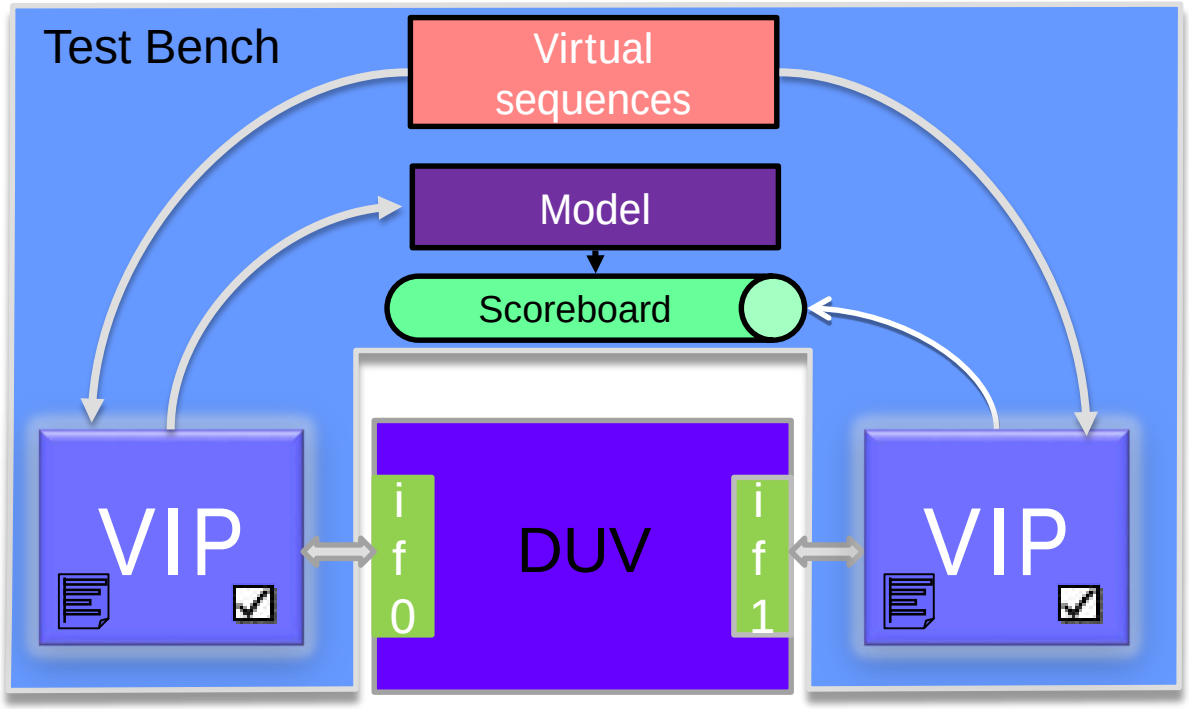


- Legal configurations are hard to generate, with complex resources management
- … recurrent problem raised @each verification level

CRV LIMITATIONS

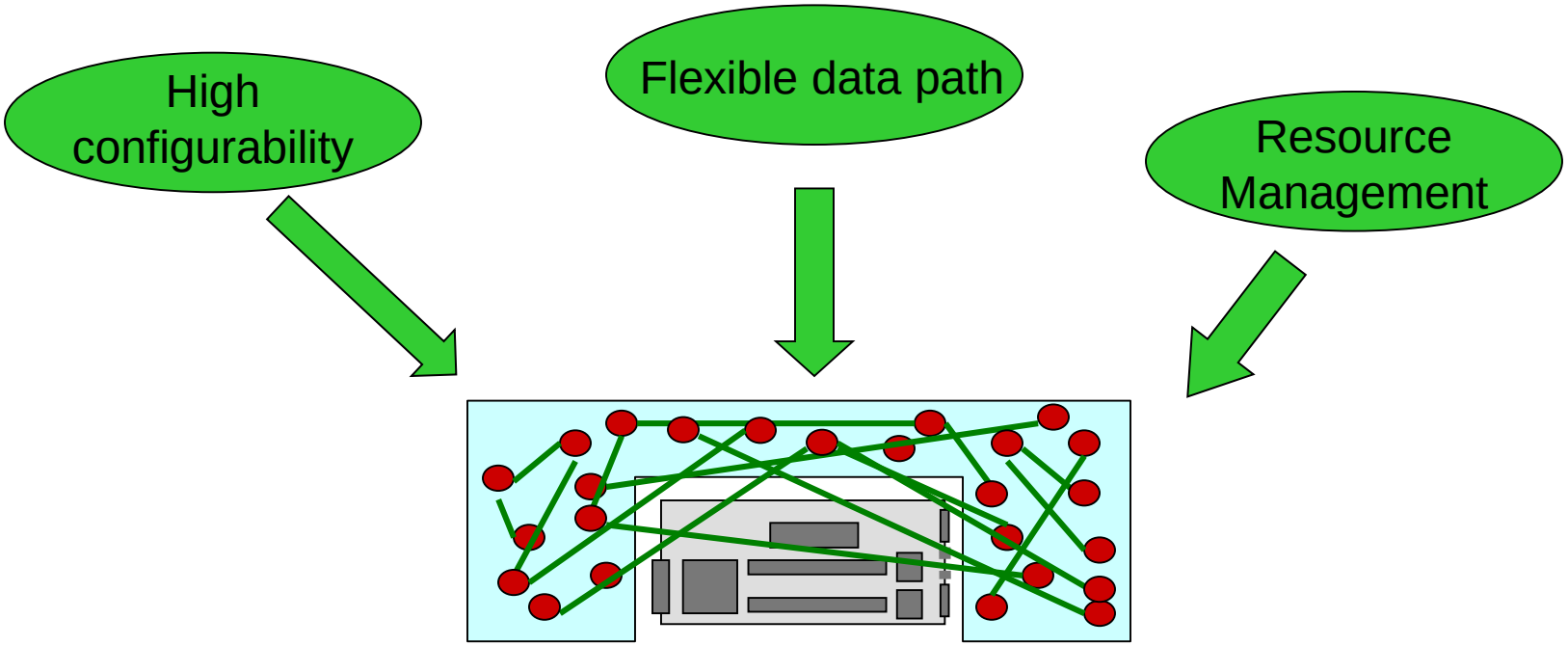
Original Approach

- Constraint Random Verification (VIPs, scbd, sequences...)
- C reference model



Constraints Based Modeling

- Multiple stages, each requiring a large constraints set
- Additional constraints required to ensure valid data path
- Data paths can be multiple and re-configurable



Tedious generation of legal configurations !

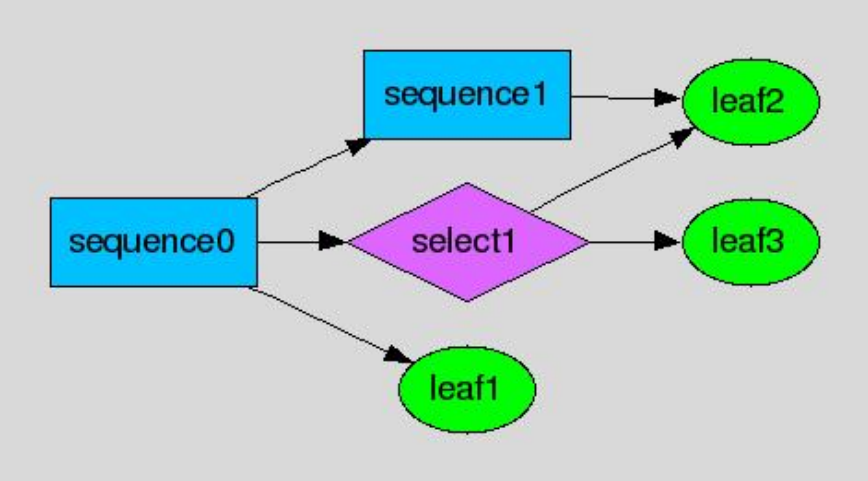
Challenges

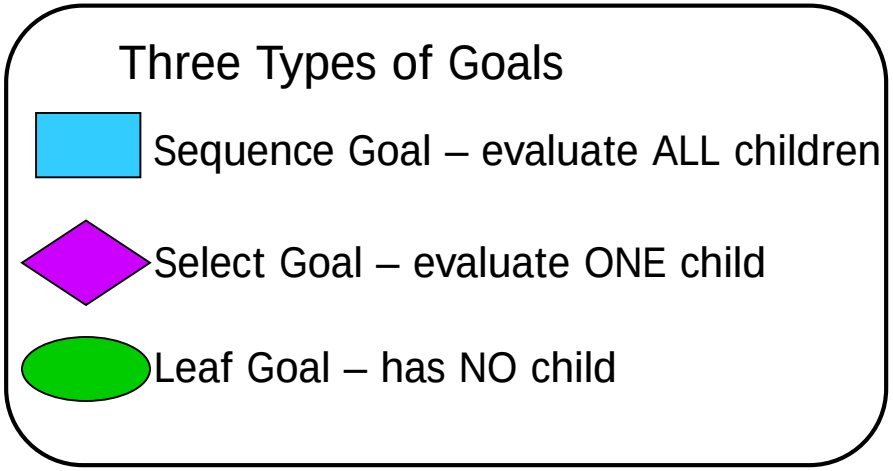
- Coverage closure
  - Constraints set was hard to develop and maintain
  - Huge test suite
  - Some scenario still missing
- Horizontal reuse
  - Important constraint subset tightly linked to IP version
- Vertical reuse
  - Different verification approaches and languages

GRAPH BASED VERIFICATION MAIN CONCEPTS

Overview

- User describes sequential steps required to generate verification scenario using graph

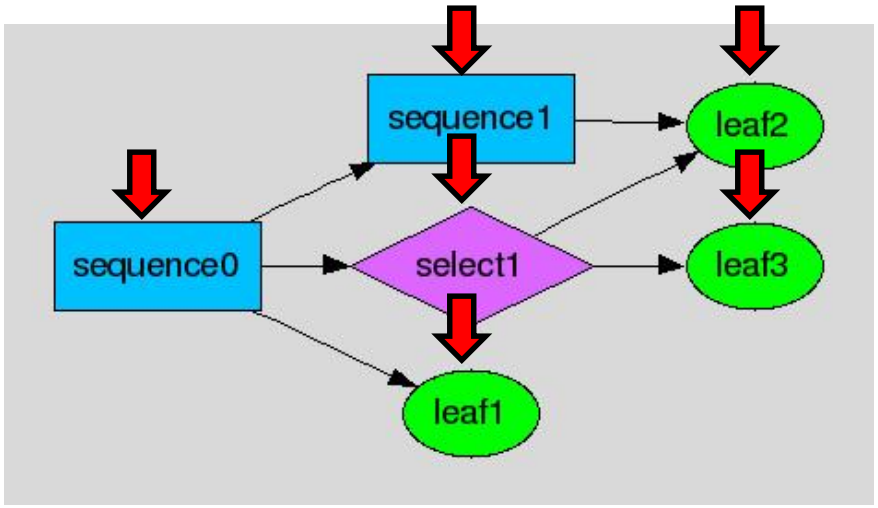




- Nodes can be used to perform
  - Configuration generation
  - TB interaction: data injection/grabbing, synchronization on events
  - File generation

Graph Evaluation

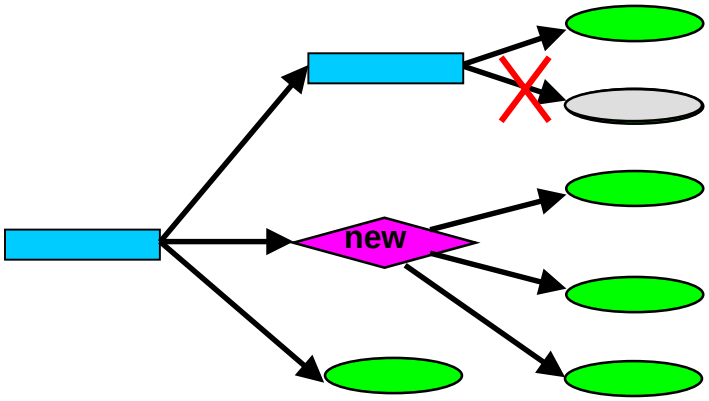
- Ordered graph walk
- Multiple and concurrent evaluations are possible
- Executing a node means evaluating the corresponding C++ function




```
goal sequence0 {…}
goal sequence1 {…}
goal leaf2 {…}
goal select1 {…}
goal leaf3 {…}
goal leaf1 {…}
```

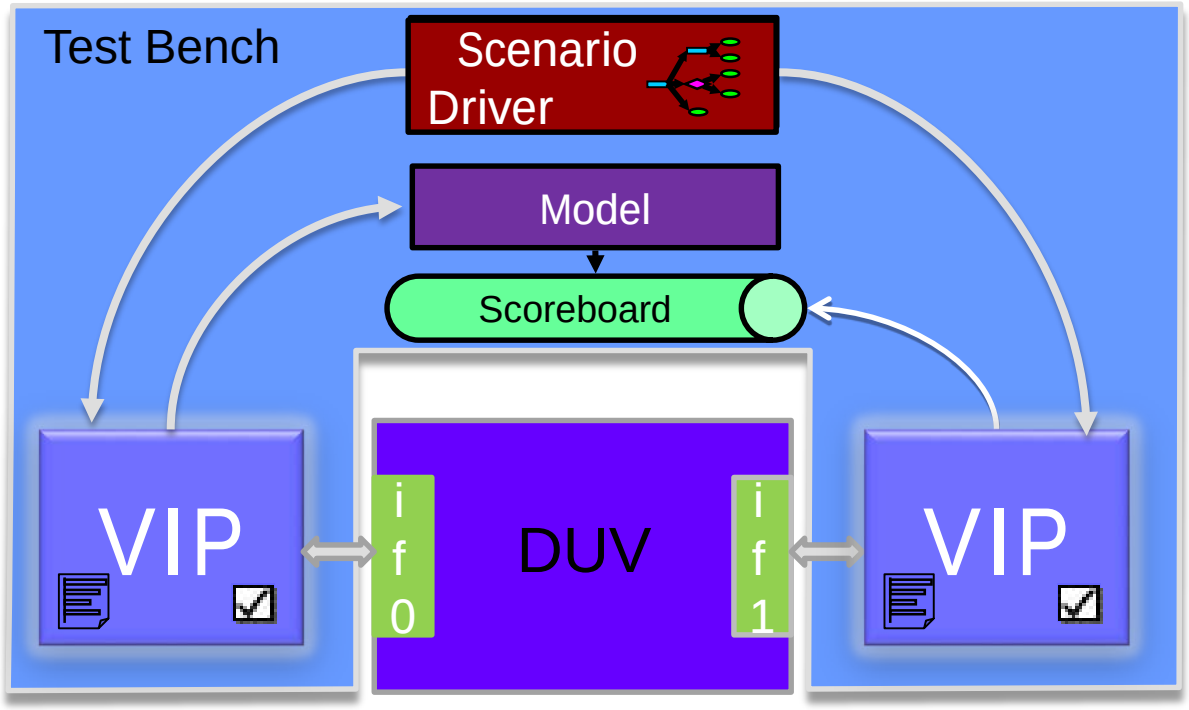
Graph Evaluation (cont'd)

- Constraints can be applied to any ***select*** child
  - If ***masked***, a child will ***never*** be selected
  - If ***forced***, a child will ***always*** be selected
- Constraints can be static and/or dynamic
- Goal may be redefined (body and subgraph)

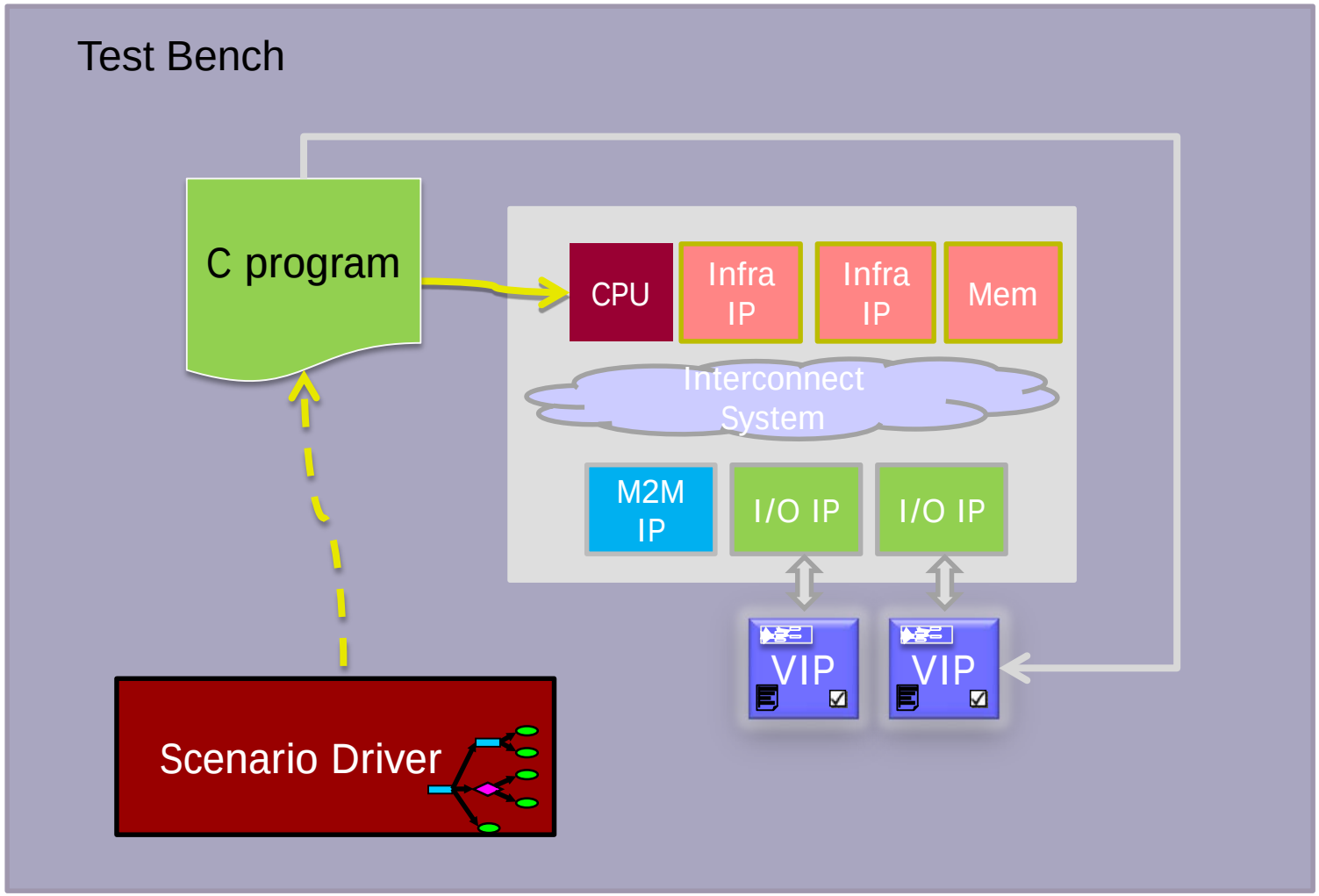


APPLICATION

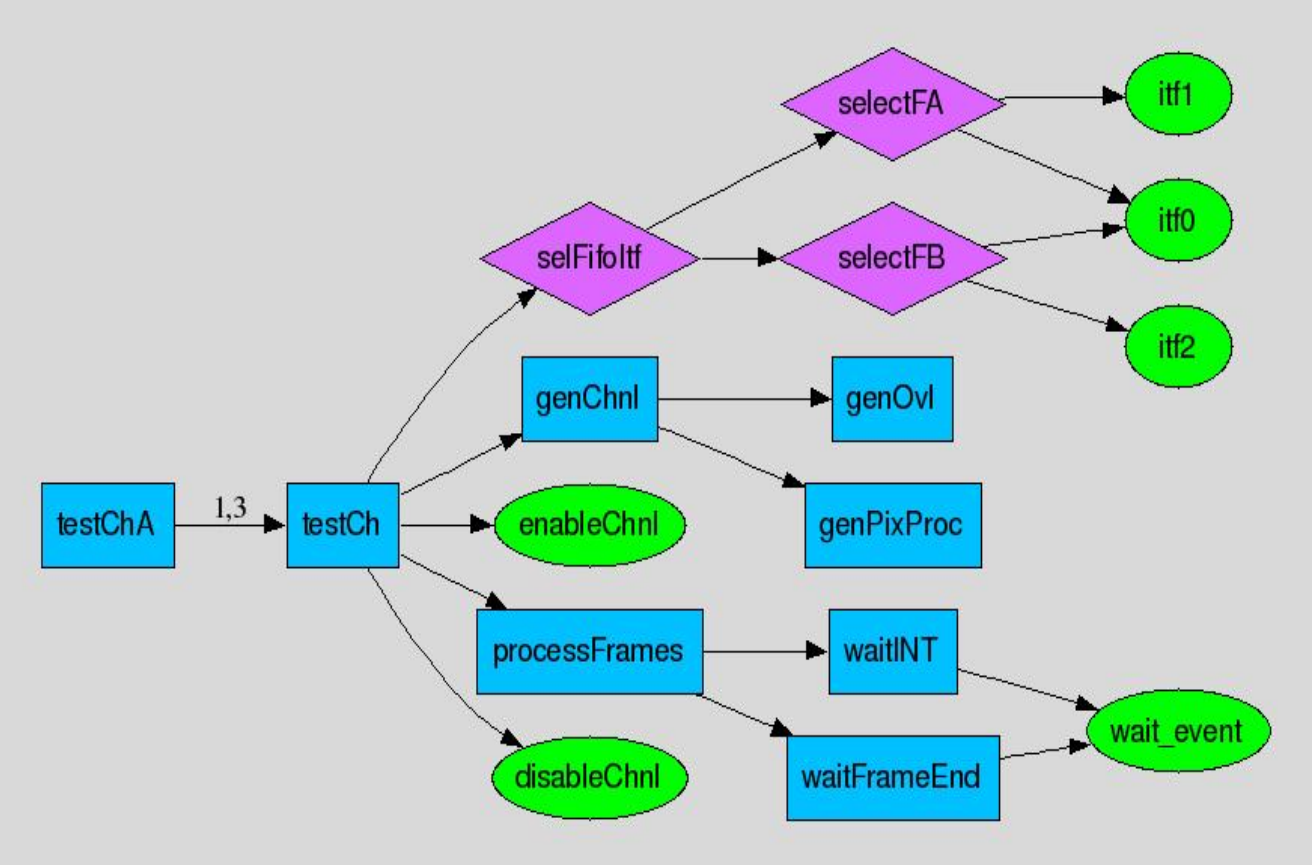
Application On Our Controller IP Testbench



Application On Our Controller System Test Generation



Graph Overview



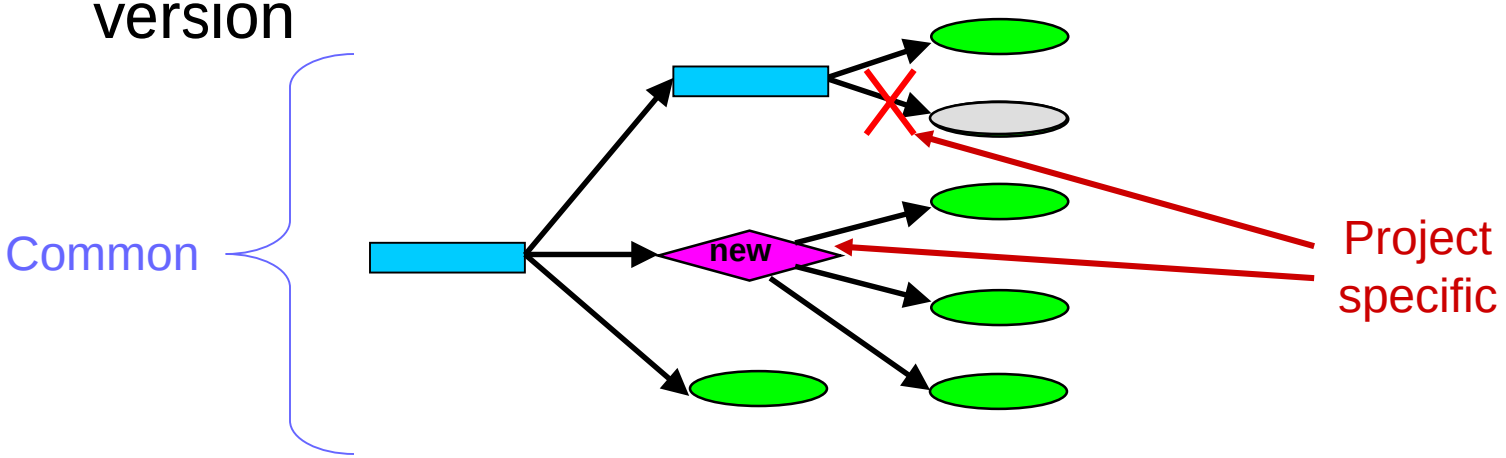
BENEFITS

Coverage Closure

- The sequential nature of graph evaluation permitted
  - An easy decomposition of the original problem
  - An easy datapath modeling
  - Smooth resources management
- Generation of unexplored scenario
  - New bugs found
- Re-writing of the test suite
  - From basic to very complex tests
  - Faster Verification closure

Horizontal reuse
Maintenance and IP derivatives

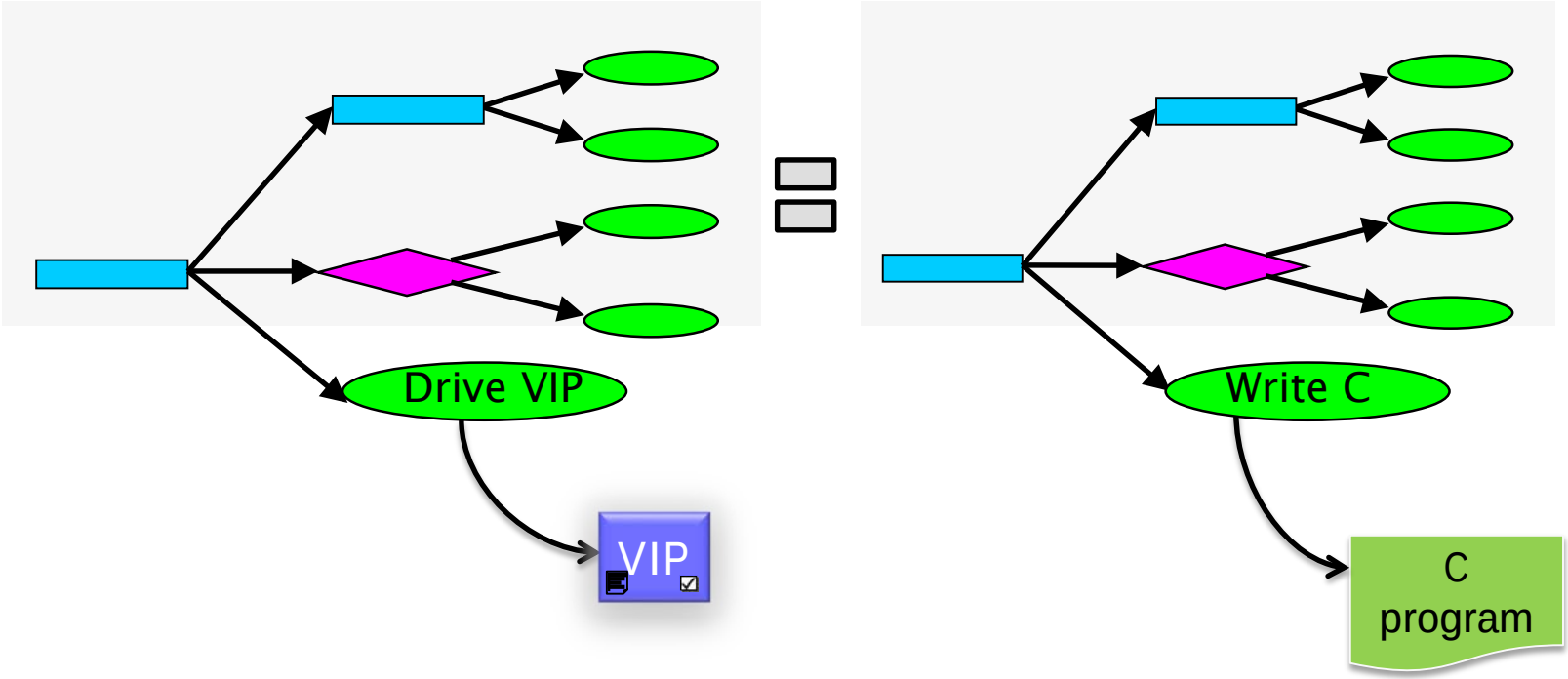
- Easy derivatives support via several select and leaf goals redefinitions
  - Project independent common pool of files for easy maintenance
  - Few configuration specific files are needed per IP version



  - Time to first valid test divided by 2

Vertical Reuse
From IP to SoC

- Most of the graph model is common to both platforms
- Natural split between nodes that are platform agnostic and those which are not



Conclusion

- CRV faces some limits on really complex IPs
- Our experience with Graph Based Verification was positive
  - Productivity and quality improved
- We applied it @IP and @SoC
  - Vertical reuse enabler
  - System tests (deployment on-going)