Development of a pixel ASIC verification environment

Elia Conti, Sara Marconi

SystemVerilog and UVM mini-workshop – 14/11/2013

- Introduction
  - Requirements of VE for next generation pixel chips
- Class-based SystemVerilog VE
  - Device Under Test (DUT)
  - Verification components
- UVM VE
  - Evaluation of simulation performance
- Conclusion and future work

RD53 Collaboration (CMS/ATLAS)

Development of pixel readout integrated circuits for extreme rate and radiation

ATLAS and CMS **phase 2** pixel upgrades (~2020)

→ Peak luminosity 1-2×10^{34} cm^{-2} s^{-1}

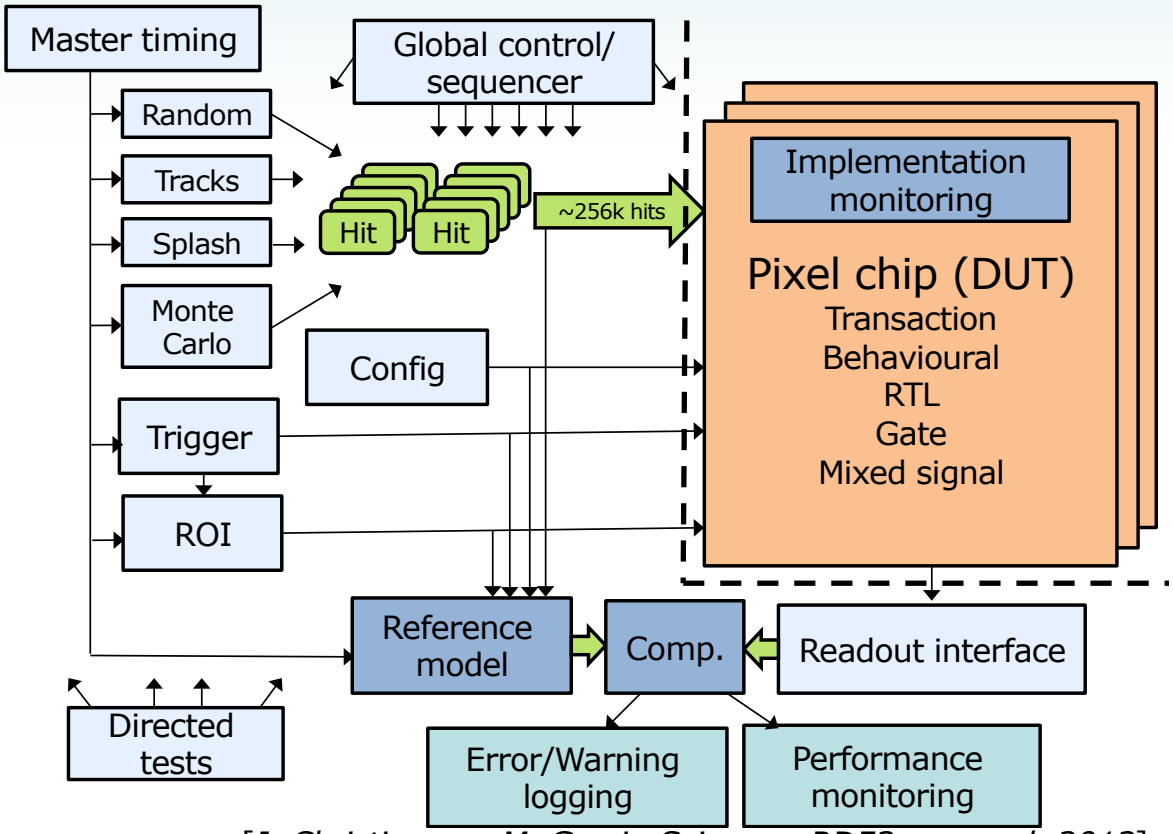
Critical challenges on IC design

- Smaller pixels to resolve tracks (*size reduction*)
- Hit rates ~1-2 GHz/cm^2 (*data rate*)
- Radiation levels ~1 Grad (*radiation hardness techniques*)
- Much higher output bandwidth (*transmission*)
- Large IC format (*low power*)
- Use of CMOS 65 nm (*technology*)

Goal of the verification working group:

Development of dedicated verification and simulation framework for optimization of next generation pixel chips

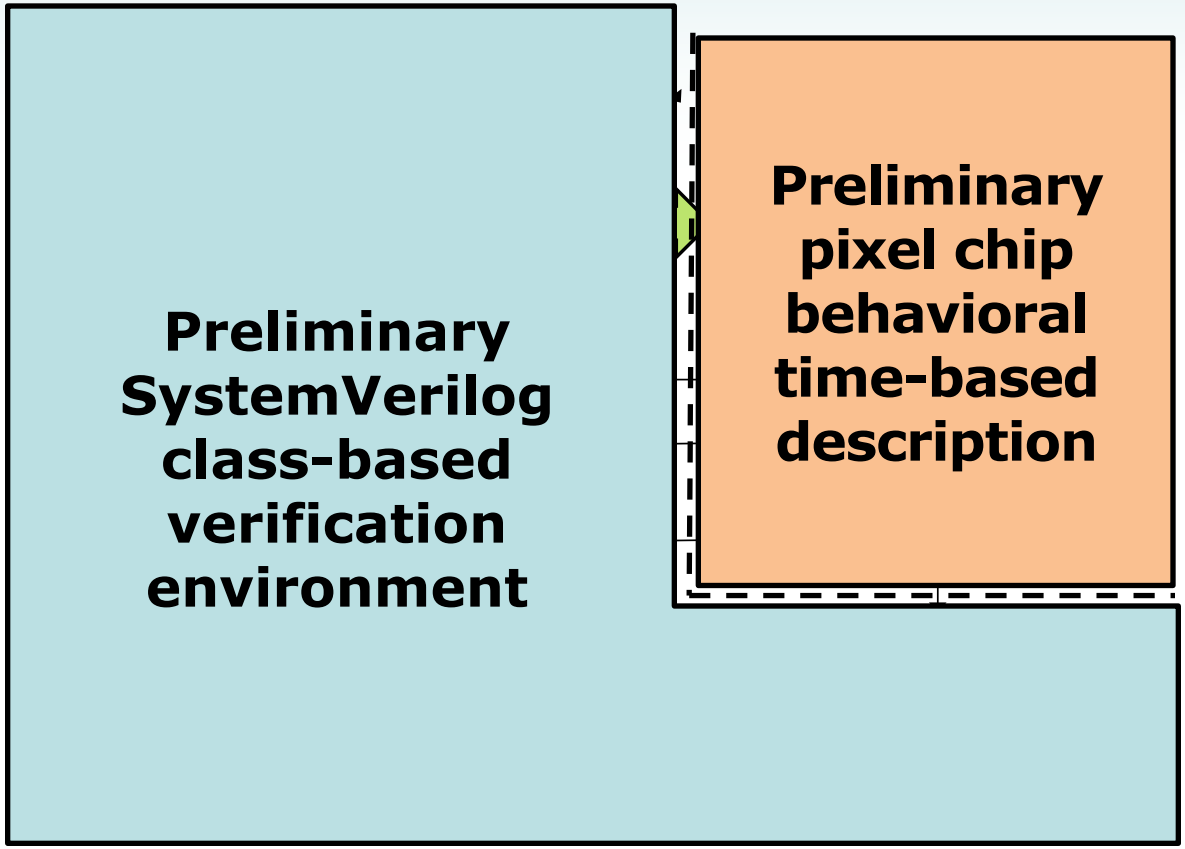
- Simulation of alternative pixel chip architectures at increasingly refined level as design progresses
- Automated verification functions
- Different kinds of data sets
  - realistic (extreme) pixel hits and triggers with constrained random distributions
  - particle hits from external Monte Carlo simulations and sensor simulations
  - mixed data



[*J. Christiansen, M. Garcia-Sciveres, RD53 proposal, 2013*]

- Hardware Verification and Description Language (HVDL): **SystemVerilog**
  - module-based Device Under Test (DUT) / class-based verification components
  - standardization, reusability → **Universal Verification Methodology (UVM)** library

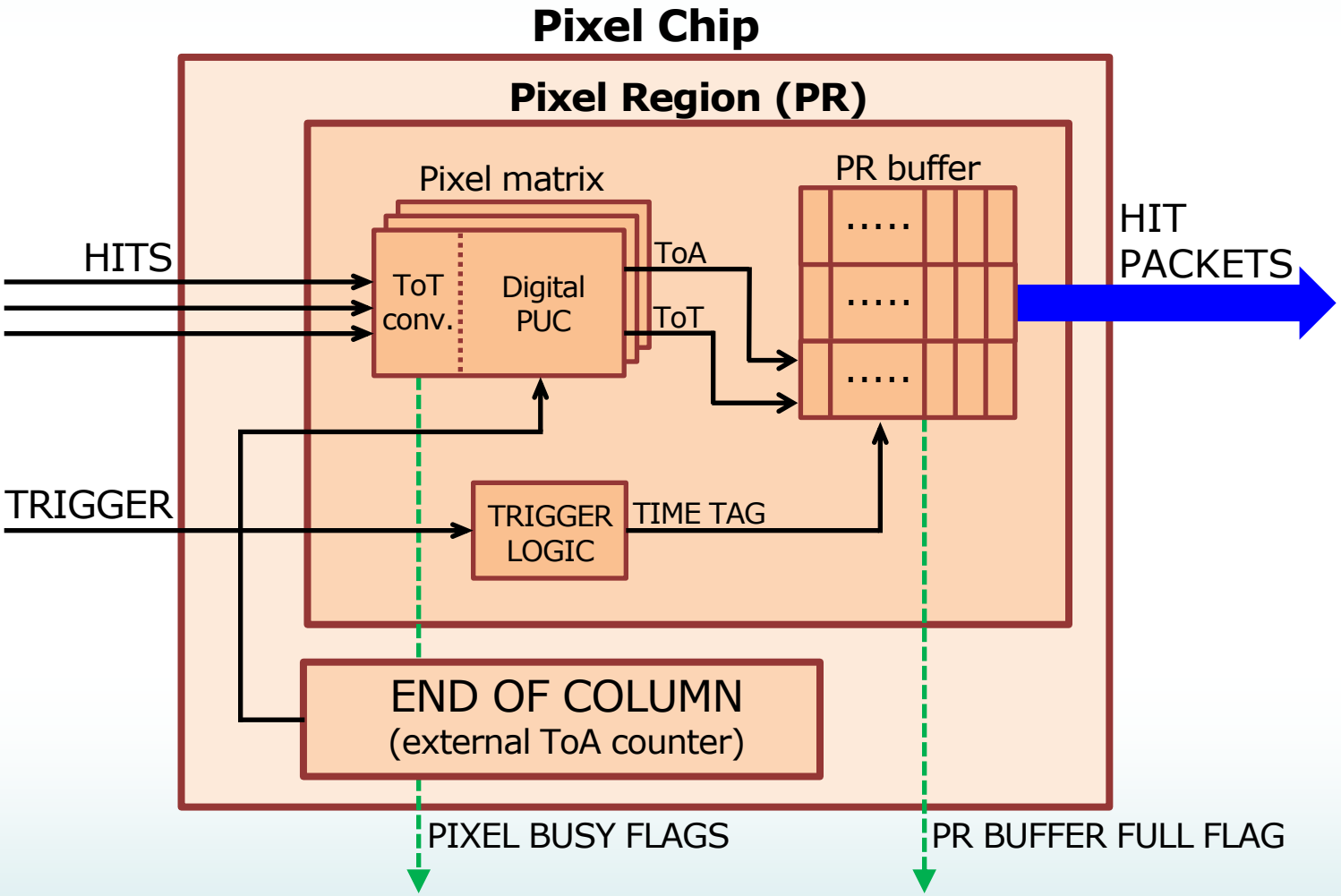
- Simulation of alternative pixel chip architectures at increasingly refined level as design progresses
- Automated verification functions
- Different kinds of data sets
  - realistic (extreme) pixel hits and triggers with constrained random distributions
  - particle hits from external Monte Carlo simulations and sensor simulations
  - mixed data



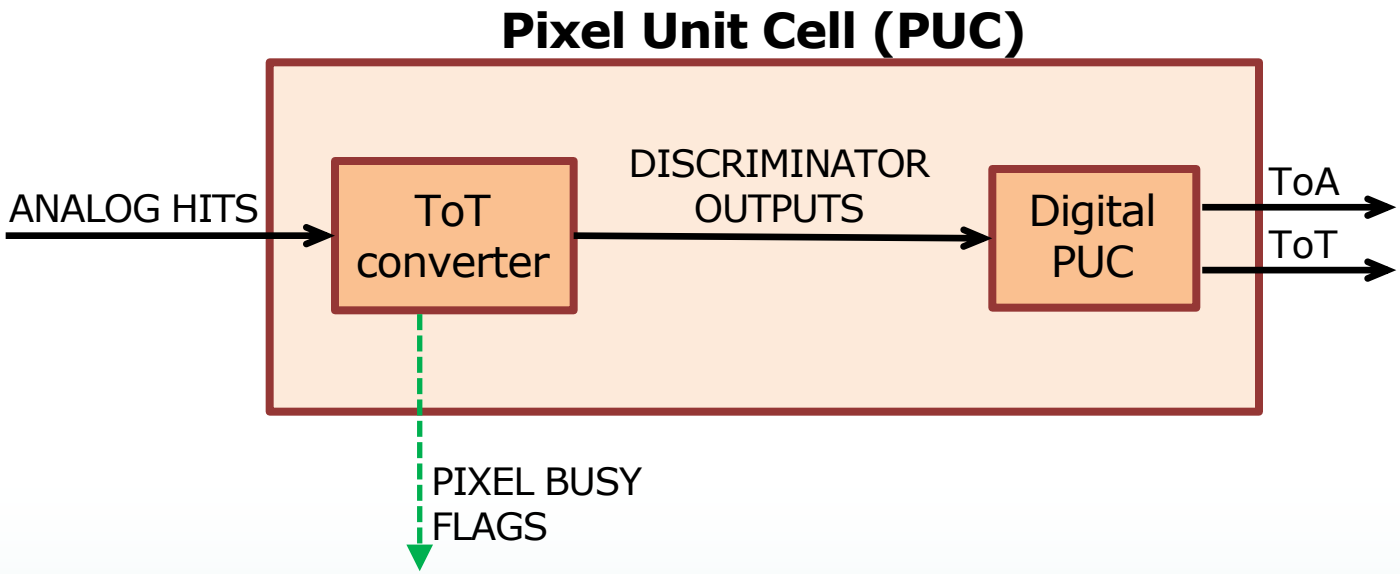
- Hardware Veri… Verilog
  - module-based … nts
  - standardizatio… M) library

Software tool: *Cadence Incisive*

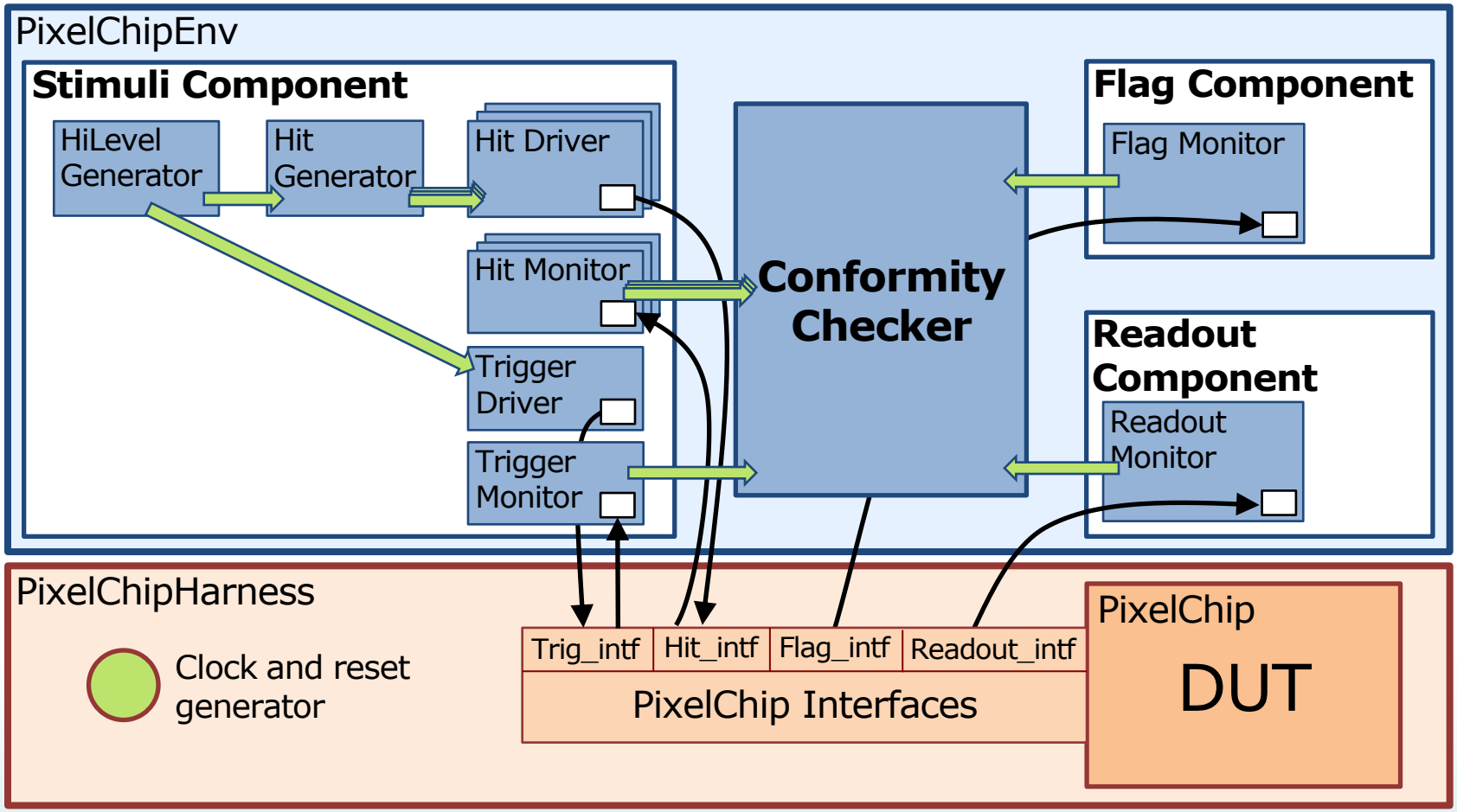
- Single pixel region with custom number of pixels
- PR buffer is an array of SystemVerilog queues

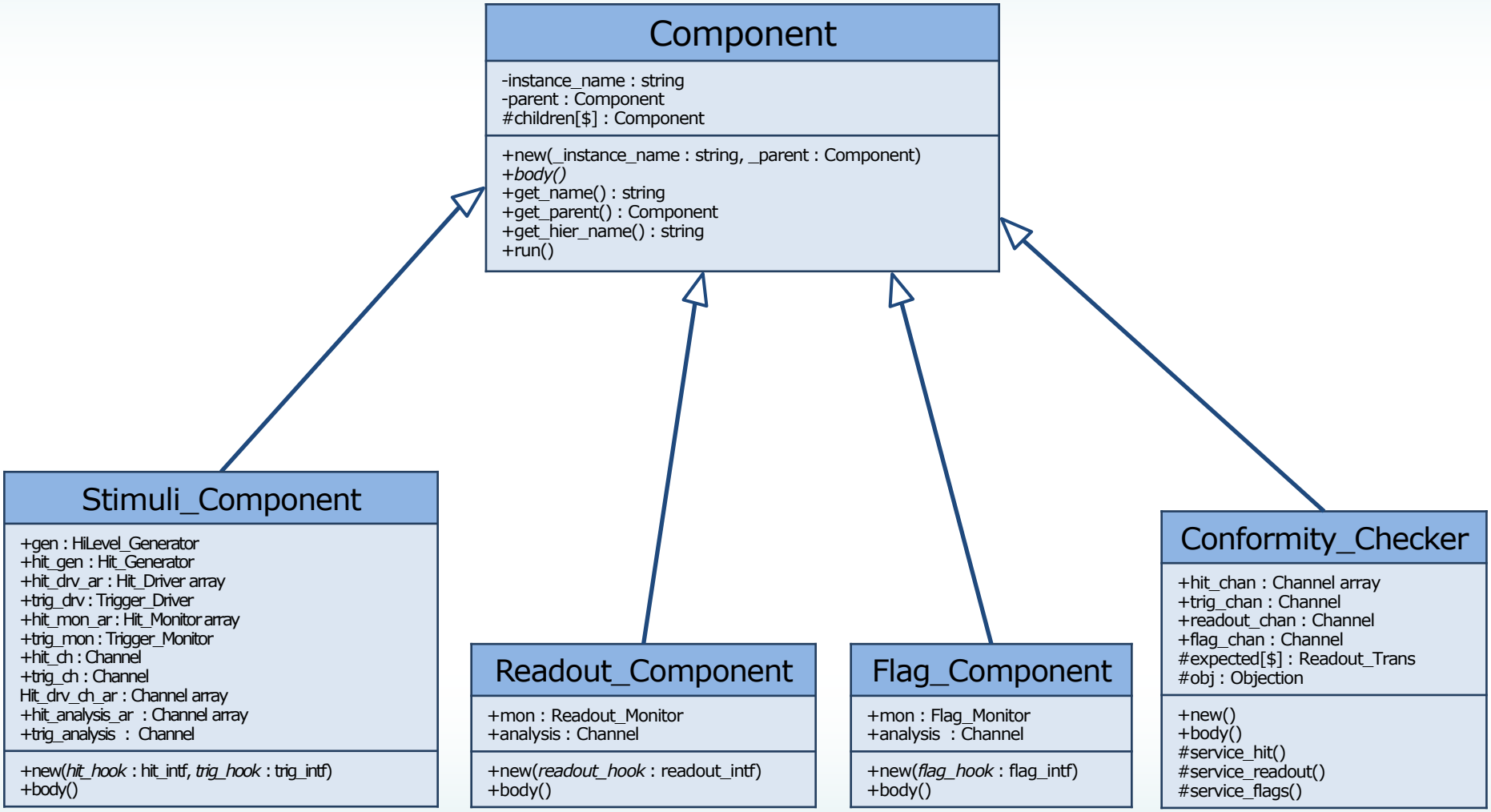


- In order to keep hit generation as general as possible a simple ToT converter module has been defined which abstracts the behavior of the analog front-end
- Current ToT converter does not produce time-walked hits



- Automated stimuli generation: randomized/external file
- Conformity checks between pixel chip inputs and outputs
- Monitoring and reporting on lost hits (pixel dead time/PR buffer full)




Component
-instance_name : string
-parent : Component
#children[$] : Component
+new(_instance_name : string, _parent : Component)
+body()
+get_name() : string
+get_parent() : Component
+get_hier_name() : string
+run()
Stimuli_Component
+gen : HiLevel_Generator
+hit_gen : Hit_Generator
+hit_drv_ar : Hit_Driver array
+trig_drv : Trigger_Driver
+hit_mon_ar : Hit_Monitor array
+trig_mon : Trigger_Monitor
+hit_ch : Channel
+trig_ch : Channel
Hit_drv_ch_ar : Channel array
+hit_analysis_ar : Channel array
+trig_analysis : Channel
+new(hit_hook : hit_intf, trig_hook : trig_intf)
+body()
Readout_Component
+mon : Readout_Monitor
+analysis : Channel
+new(readout_hook : readout_intf)
+body()
Flag_Component
+mon : Flag_Monitor
+analysis : Channel
+new(flag_hook : flag_intf)
+body()
Conformity_Checker
+hit_chan : Channel array
+trig_chan : Channel
+readout_chan : Channel
+flag_chan : Channel
#expected[$] : Readout_Trans
#obj : Objection
+new()
+body()
#service_hit()
#service_readout()
#service_flags()

Stimuli_Component

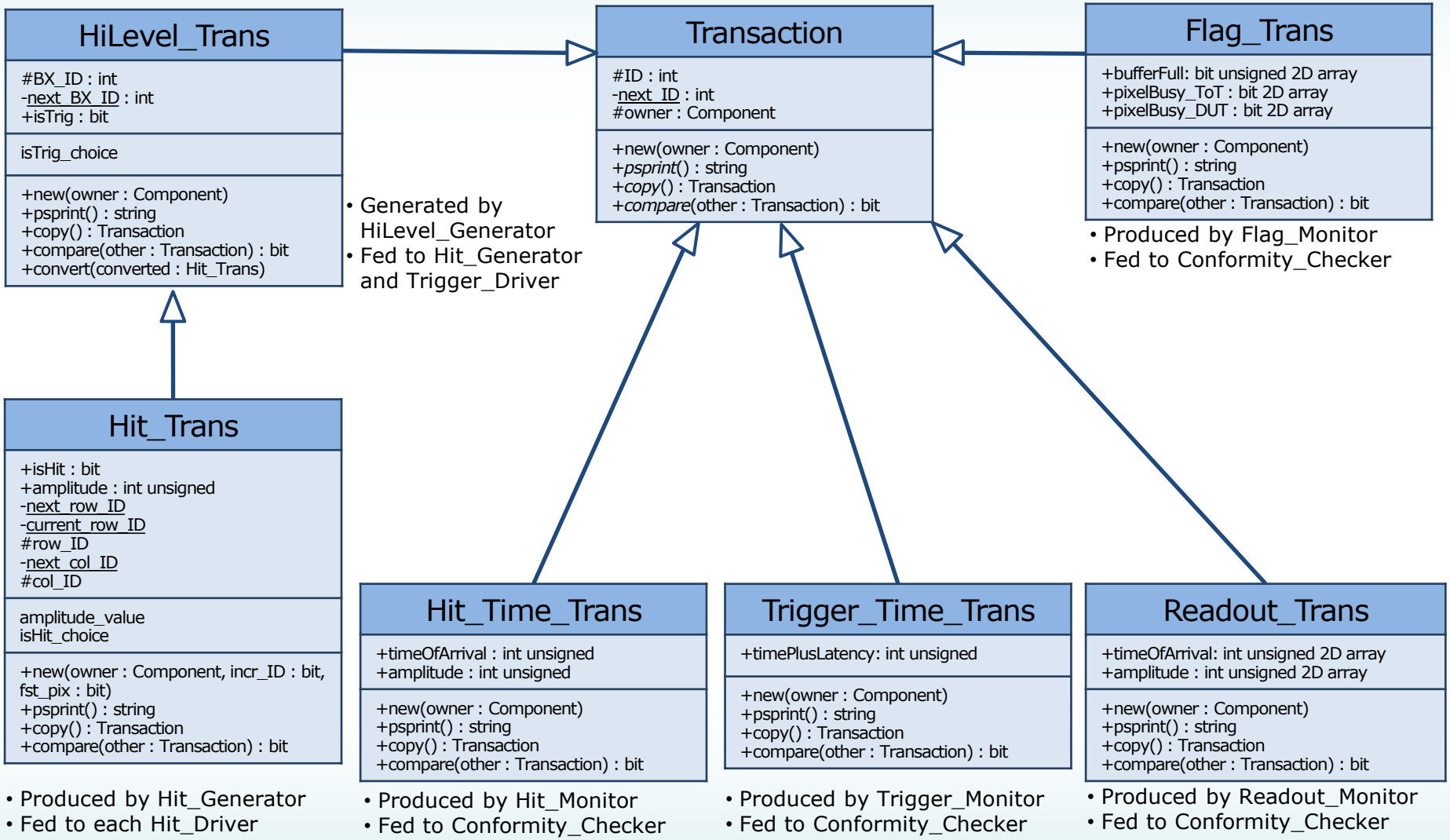
- HiLevel_Generator: generation of high level transactions (randomized or from external file)
- High level transactions are fed to Trigger_Driver (translation into trigger signal) and to Hit_Generator
- Hit_Generator: translation + randomization of high level transactions into a matrix of hit transactions (one Hit_Driver per pixel)
- Hit_Driver: translation into "analog hit" signal (**no delays modeled so far**)
- Hit_Monitor and Trigger_Monitor: generation of hit transactions and trigger transactions (adding time information) from pixel chip inputs

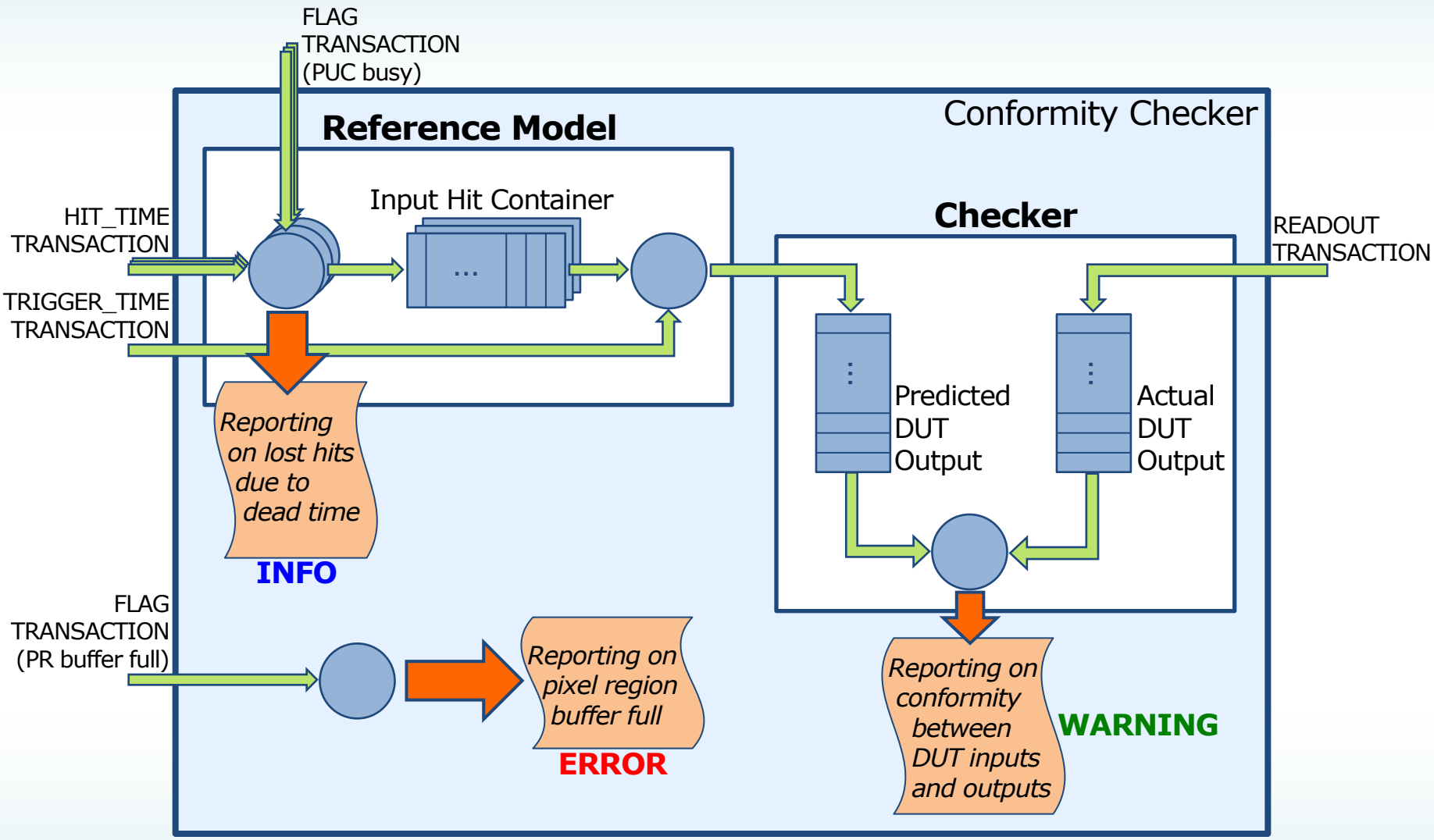
Readout_Component

- Readout_Monitor: generation of readout transactions from pixel chip outputs

Flag_Component

- Flag_Monitor: generation of flag transactions from pixel chip flags (pixel busy and PR buffer full)


HiLevel_Trans
#BX_ID : int
-next_BX_ID : int
+isTrig : bit
isTrig_choice
+new(owner : Component)
+psprint() : string
+copy() : Transaction
+compare(other : Transaction) : bit
+convert(converted : Hit_Trans)
• Generated by HiLevel_Generator
• Fed to Hit_Generator and Trigger_Driver
Transaction
#ID : int
-next_ID : int
#owner : Component
+new(owner : Component)
+psprint() : string
+copy() : Transaction
+compare(other : Transaction) : bit
Flag_Trans
+bufferFull: bit unsigned 2D array
+pixelBusy_ToT : bit 2D array
+pixelBusy_DUT : bit 2D array
+new(owner : Component)
+psprint() : string
+copy() : Transaction
+compare(other : Transaction) : bit
• Produced by Flag_Monitor
• Fed to Conformity_Checker
Hit_Trans
+isHit : bit
+amplitude : int unsigned
-next_row_ID
-current_row_ID
#row_ID
-next_col_ID
#col_ID
amplitude_value
isHit_choice
+new(owner : Component, incr_ID : bit, fst_pix : bit)
+psprint() : string
+copy() : Transaction
+compare(other : Transaction) : bit
• Produced by Hit_Generator
• Fed to each Hit_Driver
Hit_Time_Trans
+timeOfArrival : int unsigned
+amplitude : int unsigned
+new(owner : Component)
+psprint() : string
+copy() : Transaction
+compare(other : Transaction) : bit
• Produced by Hit_Monitor
• Fed to Conformity_Checker
Trigger_Time_Trans
+timePlusLatency: int unsigned
+new(owner : Component)
+psprint() : string
+copy() : Transaction
+compare(other : Transaction) : bit
• Produced by Trigger_Monitor
• Fed to Conformity_Checker
Readout_Trans
+timeOfArrival: int unsigned 2D array
+amplitude : int unsigned 2D array
+new(owner : Component)
+psprint() : string
+copy() : Transaction
+compare(other : Transaction) : bit
• Produced by Readout_Monitor
• Fed to Conformity_Checker


FLAG
TRANSACTION
(PUC busy)
Conformity Checker
Reference Model
Input Hit Container
...
HIT_TIME
TRANSACTION
TRIGGER_TIME
TRANSACTION
Checker
READOUT
TRANSACTION
Predicted
DUT
Output
Actual
DUT
Output
Reporting
on lost hits
due to
dead time
INFO
FLAG
TRANSACTION
(PR buffer full)
Reporting on
pixel region
buffer full
ERROR
Reporting on
conformity
between
DUT inputs
and outputs
WARNING

Motivation

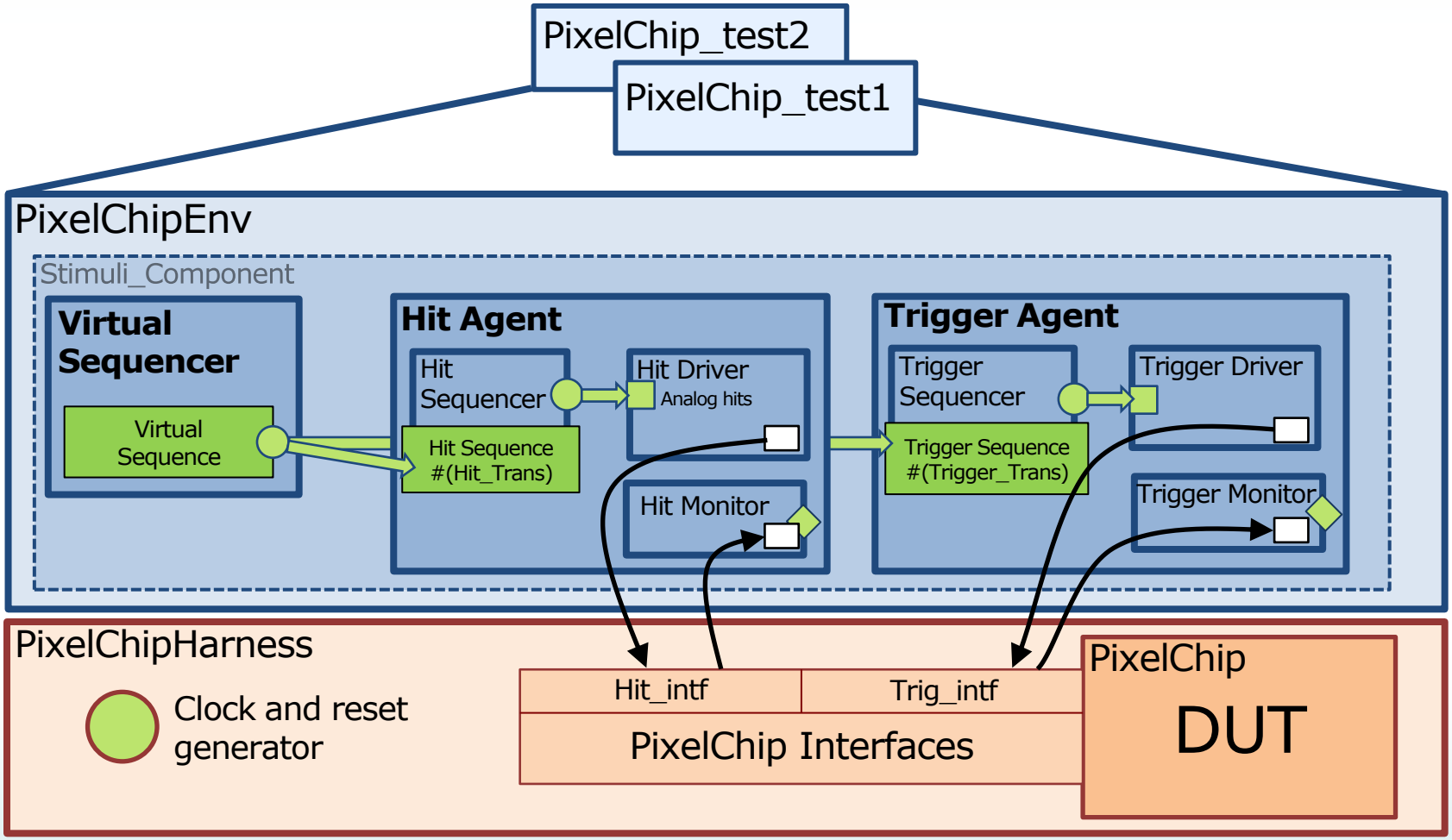
Standardize into UVM for:

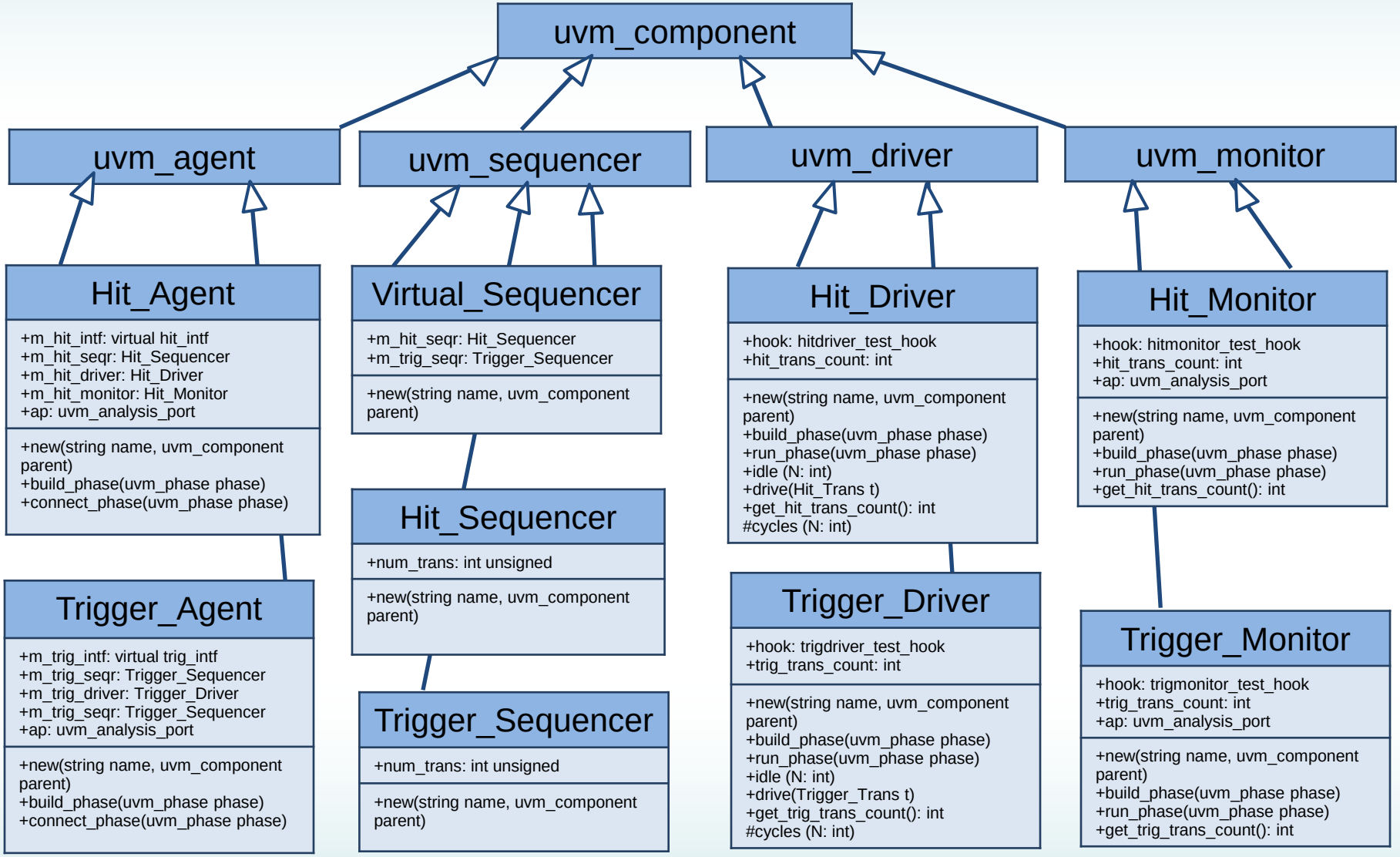
- More solid and documented base classes
- Mature standard and growing community
- Highly customizable reporting features
- Highly configurable and reusable verification environment

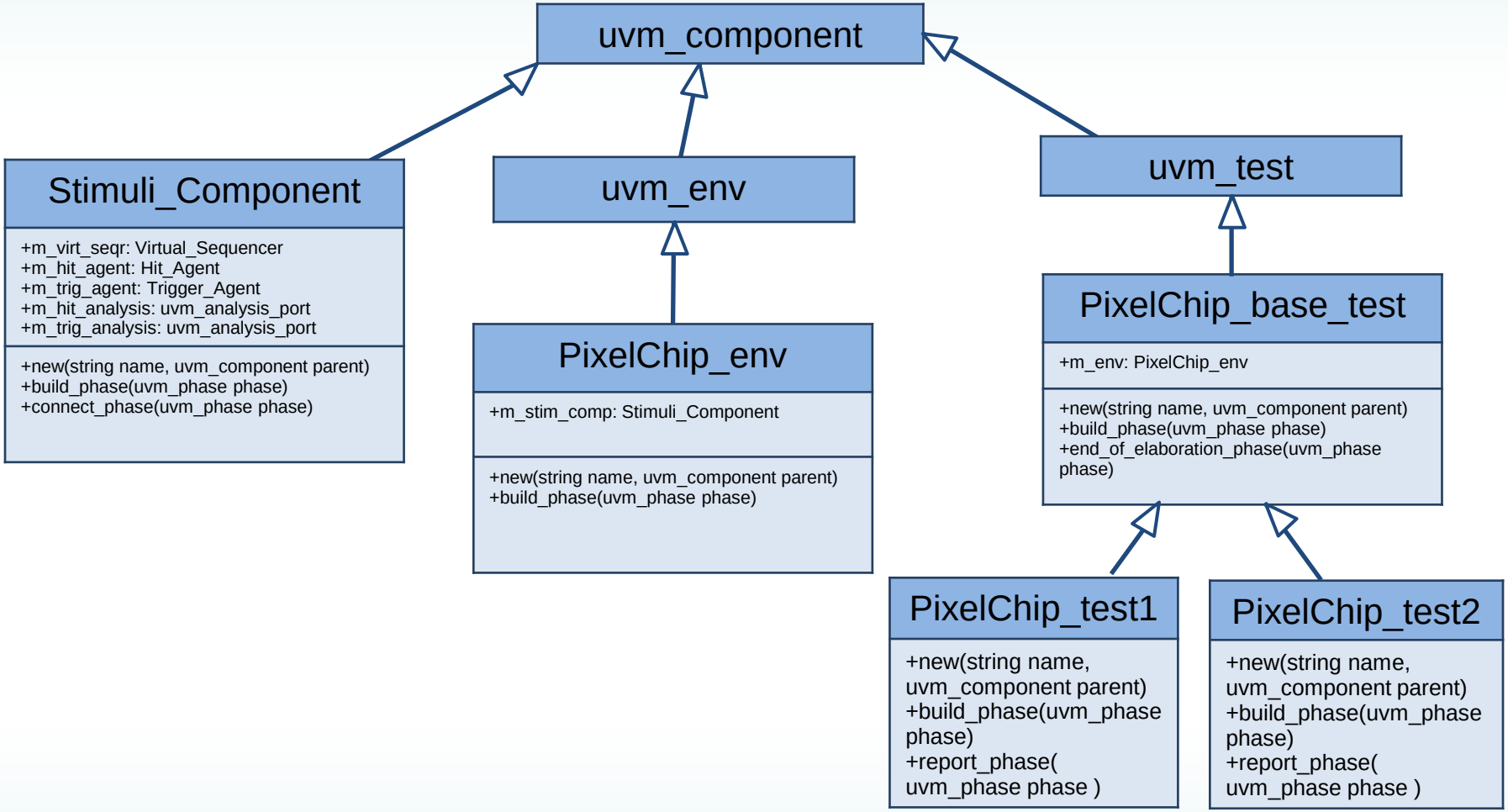
At the moment, focus on stimuli generation only

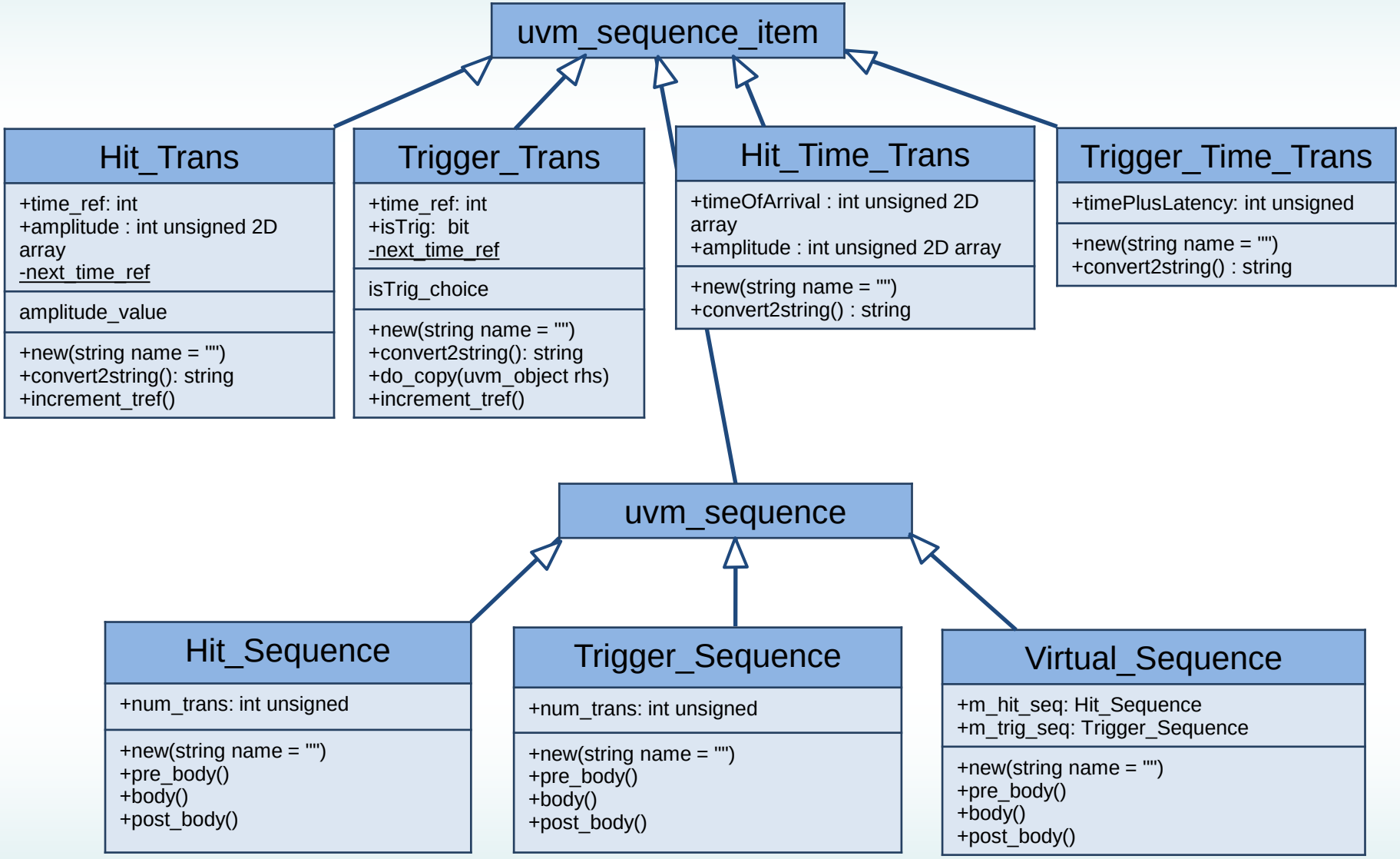
- No Readout/Flag components
- No Conformity Checker
- Study on scalability

- Single agent with transaction containing array of hits
- Different tests can be written




uvm_component
uvm_agent
uvm_sequencer
uvm_driver
uvm_monitor
Hit_Agent
+m_hit_intf: virtual hit_intf
+m_hit_seqr: Hit_Sequencer
+m_hit_driver: Hit_Driver
+m_hit_monitor: Hit_Monitor
+ap: uvm_analysis_port
+new(string name, uvm_component parent)
+build_phase(uvm_phase phase)
+connect_phase(uvm_phase phase)
Trigger_Agent
+m_trig_intf: virtual trig_intf
+m_trig_seqr: Trigger_Sequencer
+m_trig_driver: Trigger_Driver
+m_trig_seqr: Trigger_Sequencer
+ap: uvm_analysis_port
+new(string name, uvm_component parent)
+build_phase(uvm_phase phase)
+connect_phase(uvm_phase phase)
Virtual_Sequencer
+m_hit_seqr: Hit_Sequencer
+m_trig_seqr: Trigger_Sequencer
+new(string name, uvm_component parent)
Hit_Sequencer
+num_trans: int unsigned
+new(string name, uvm_component parent)
Trigger_Sequencer
+num_trans: int unsigned
+new(string name, uvm_component parent)
Hit_Driver
+hook: hitdriver_test_hook
+hit_trans_count: int
+new(string name, uvm_component parent)
+build_phase(uvm_phase phase)
+run_phase(uvm_phase phase)
+idle (N: int)
+drive(Hit_Trans t)
+get_hit_trans_count(): int
#cycles (N: int)
Trigger_Driver
+hook: trigdriver_test_hook
+trig_trans_count: int
+new(string name, uvm_component parent)
+build_phase(uvm_phase phase)
+run_phase(uvm_phase phase)
+idle (N: int)
+drive(Trigger_Trans t)
+get_trig_trans_count(): int
#cycles (N: int)
Hit_Monitor
+hook: hitmonitor_test_hook
+hit_trans_count: int
+ap: uvm_analysis_port
+new(string name, uvm_component parent)
+build_phase(uvm_phase phase)
+run_phase(uvm_phase phase)
+get_hit_trans_count(): int
Trigger_Monitor
+hook: trigmonitor_test_hook
+trig_trans_count: int
+ap: uvm_analysis_port
+new(string name, uvm_component parent)
+build_phase(uvm_phase phase)
+run_phase(uvm_phase phase)
+get_trig_trans_count(): int


uvm_component
Stimuli_Component
+m_virt_seqr: Virtual_Sequencer
+m_hit_agent: Hit_Agent
+m_trig_agent: Trigger_Agent
+m_hit_analysis: uvm_analysis_port
+m_trig_analysis: uvm_analysis_port
+new(string name, uvm_component parent)
+build_phase(uvm_phase phase)
+connect_phase(uvm_phase phase)
uvm_env
PixelChip_env
+m_stim_comp: Stimuli_Component
+new(string name, uvm_component parent)
+build_phase(uvm_phase phase)
uvm_test
PixelChip_base_test
+m_env: PixelChip_env
+new(string name, uvm_component parent)
+build_phase(uvm_phase phase)
+end_of_elaboration_phase(uvm_phase phase)
PixelChip_test1
+new(string name, uvm_component parent)
+build_phase(uvm_phase phase)
+report_phase(uvm_phase phase)
PixelChip_test2
+new(string name, uvm_component parent)
+build_phase(uvm_phase phase)
+report_phase(uvm_phase phase)


uvm_sequence_item
Hit_Trans
+time_ref: int
+amplitude : int unsigned 2D array
-next_time_ref
amplitude_value
+new(string name = "")
+convert2string(): string
+increment_tref()
Trigger_Trans
+time_ref: int
+isTrig: bit
-next_time_ref
isTrig_choice
+new(string name = "")
+convert2string(): string
+do_copy(uvm_object rhs)
+increment_tref()
Hit_Time_Trans
+timeOfArrival : int unsigned 2D array
+amplitude : int unsigned 2D array
+new(string name = "")
+convert2string() : string
Trigger_Time_Trans
+timePlusLatency: int unsigned
+new(string name = "")
+convert2string() : string
uvm_sequence
Hit_Sequence
+num_trans: int unsigned
+new(string name = "")
+pre_body()
+body()
+post_body()
Trigger_Sequence
+num_trans: int unsigned
+new(string name = "")
+pre_body()
+body()
+post_body()
Virtual_Sequence
+m_hit_seq: Hit_Sequence
+m_trig_seq: Trigger_Sequence
+new(string name = "")
+pre_body()
+body()
+post_body()

Agents

- One agent is used for each interface of the DUT

Sequences and sequencers

- Hit_Sequence: generates a user-defined number of Hit_Trans
- Trigger_Sequence: generates a user-defined number of Trigger_Trans
- Virtual_Sequence: higher level sequence that coordinates lower level ones
- One sequencer per each sequence

Test

- Automatically instantiated when launched by top-level module
- Specified through command line
- Configures the environment (override classes, active/passive agents, number of transactions, default sequences to be run…) before building it

  → VE source code does not need to be altered, only accessing the test file one can customize the test

- Different tests are being defined

- Simulations performed with only Stimuli_Component in the PixelChip_Env
- Matrix size: **1024x256 pixels (256K)**
- **NO DUT** actually present (*further development needed*)
- Debug messages NOT evaluated (they cause huge overhead)
- Performance analyzed through **Incisive Advanced Profiler** (***Iprof***):
  - **100 transactions** sent in sequence (**to each pixel):**
    - Memory Usage: 20.3M program + 1113.1M data + 1.0M profile = 1134.4M total
    - CPU Usage: 0.8s system + 142.2s user = **143.0s** total (99.9% CPU)
  - **10k transactions** sent in sequence (**to each pixel):**
    - Memory Usage: 20.3M program + 1113.1M data + 1.0M profile = 1134.4M total
    - CPU Usage: 1.1s system + 12216.0s user = **12217.1s** total (100.0% CPU)

➔ Resource usage mainly for randomization (of 256K values to be sent to each pixel)

➔ A high level strategy for generating clusters should be defined

- ✓ Preliminary version of SystemVerilog verification environment
  - Class-based SystemVerilog
  - Standardized into UVM library (focus on stimuli generation only)
- ✓ Study on scalability and simulation performance
  - Resource usage mainly for randomization of transactions

➔ Further developments on the DUT:

- Implement more complex features (i.e. PR columns)
- Event-based TLM description of pixel chip

➔ Further developments on the Verification environment:

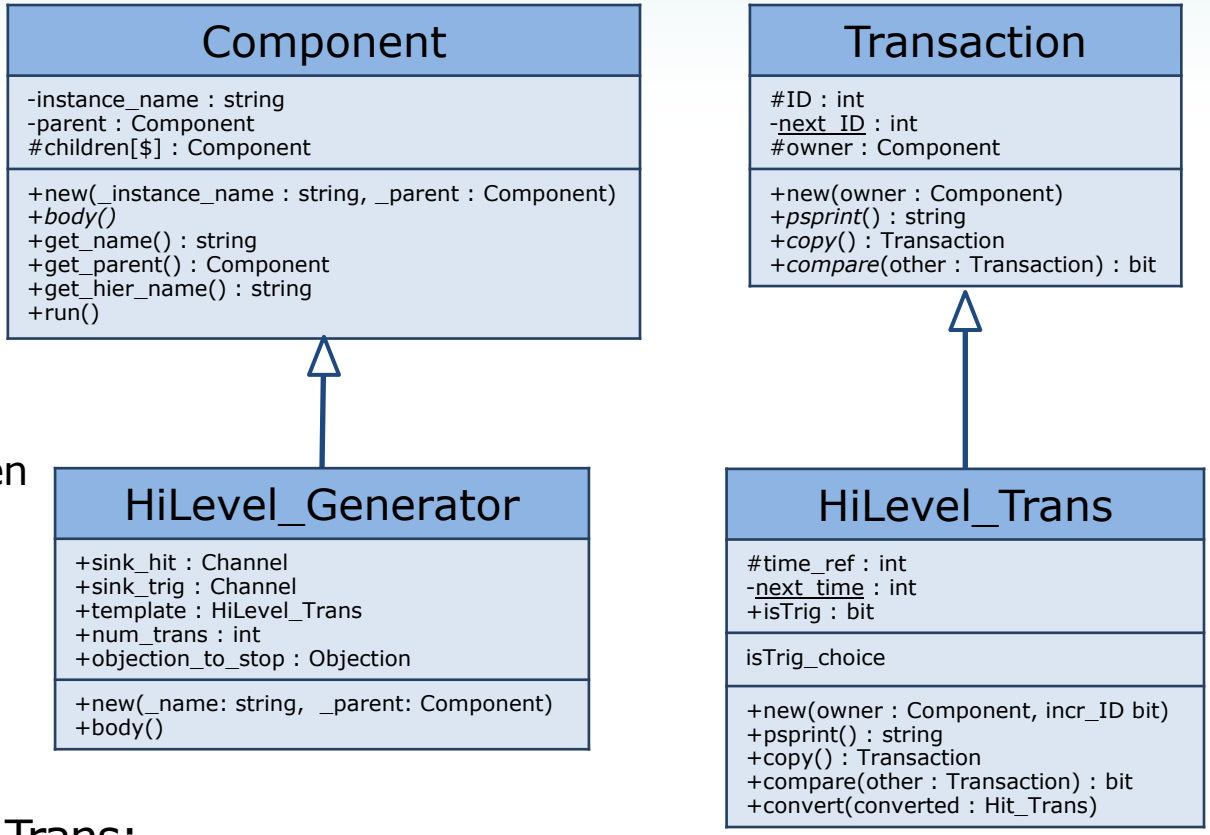
- Standardization of other components into UVM
- Implementation of new components
- Definition of high level transaction generation strategy

BACKUP

Class-based SystemVerilog VE – HiLevel Gen.

HiLevel_Generator abstracts high level interaction generator

- For each collision cycle it generates a HiLevel_Trans transaction: any number of transactions can be chosen
- This information is sent to both Hit_Driver and Trigger_Driver on 2 separate channels



Data members of HiLevel_Trans:

- time_ref associated to each transaction (data type: int)
- isTrig (yes/no signal: do we want Trigger_Driver to associate a trigger to current BX?)

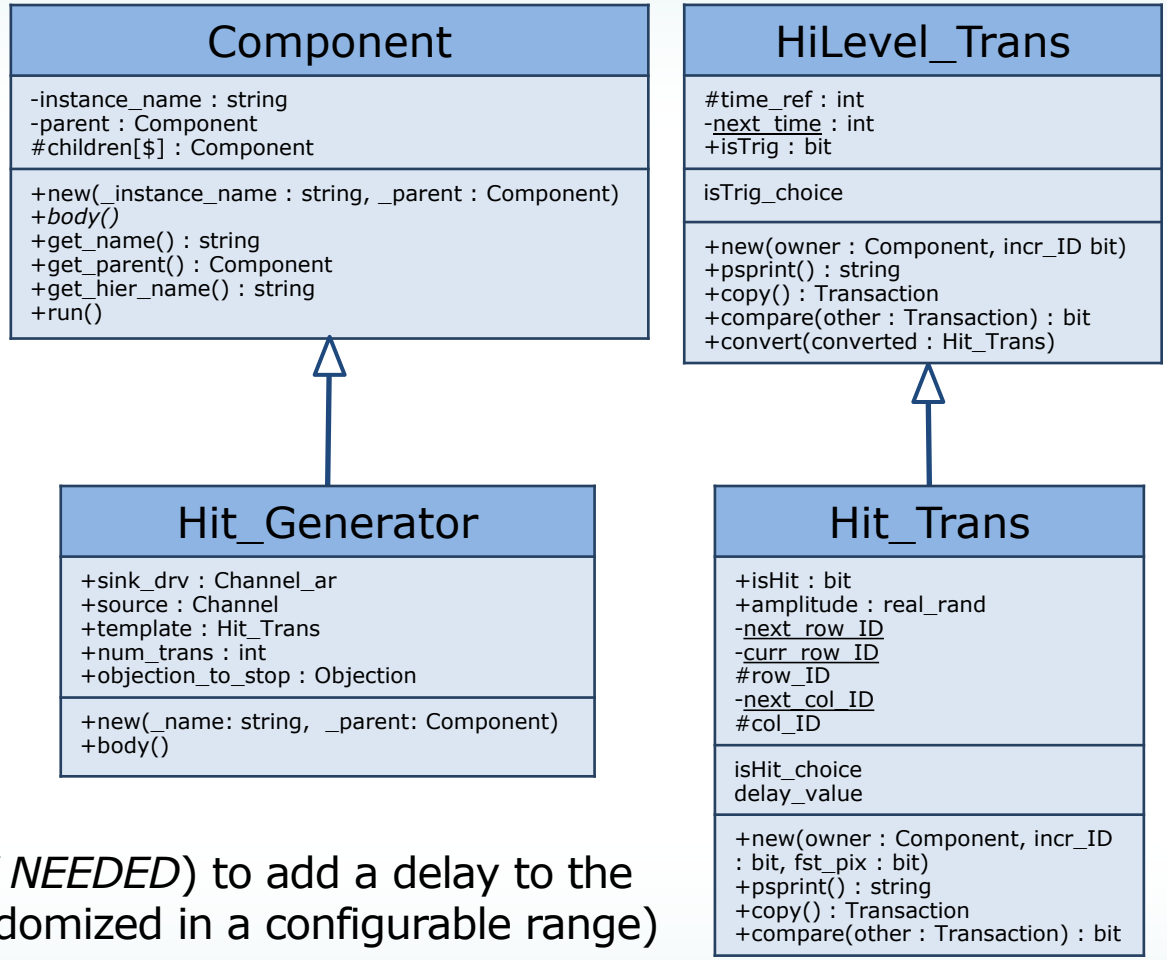
isTrig is randomized at each collision cycle with constraints (50% yes-no probability)

Class-based SystemVerilog VE – Hit Generator

Hit_Generator takes HiLevel_Trans transactions from input channel and translates them in MxN Hit_Trans transactions to be sent through MxN channels to the same number of Hit_Drivers

Hit_Trans adds

- 2 position indexes according to destination Hit_Driver's
- isHit (yes/no signal: do we want that pixel to be hit during the current BX)?
- Hit amplitude
- Hit_delay (*FURTHER STUDY NEEDED*) to add a delay to the hit generation (could be randomized in a configurable range)



So far, amplitude, isHit are randomized at each collision cycle for each channel with constraints (amplitude in a given range, isHit with 50% yes-no probability)

- Simulation performance is one of the main issues of the framework
- A study has been carried out in order to compare different simulators that are a reasonable option:
  - Vendor A
  - Vendor B
- In order to obtain meaningful results, a fully developed and specific OVM-Verification Environment and Pixel Chip has been used (**FEI4_B provided by ATLAS**)
- Both gate level and RTL model have been used for the DUT
- Different philosophy between tools when dealing with optimization:
  - **Vendor A**: Optimization options have to be indicated in the simulation command, otherwise no optimization is performed
  - **Vendor B**: if not differently specified, optimization is automatically performed (and access to the design is restricted)

- The comparison has been done on the time spent by the simulator in:
  1. Compilation of libraries
  2. Elaboration of the design hierarchy
  3. Actual Simulation

1. The two tools have shown that compilation of libraries is not a bottleneck
2. An important difference has instead been observed in terms of elaboration time:
   - Vendor A: much faster
   - Vendor B: an approach that separates the elaboration of the DUT and of the Verification Environment can be used to overcome this bottleneck (when dealing with verification)
3. Remarkable difference observed in terms of simulation time:
   - Vendor B: simulations are 3-6X faster