Follow Up GP Extension Standard Clock and Reset SC Standard

Joachim Geishauser, NXP Semiconductors

© Accellera Systems Initiative

Presentation Copyright Permission

– A non-exclusive, irrevocable, royalty-free copyright permission is granted by **NXP** to use this material in developing all future revisions and editions of the resulting draft and approved Accellera Systems Initiative **SystemC** standard, and in derivative works based on this standard.

Agenda

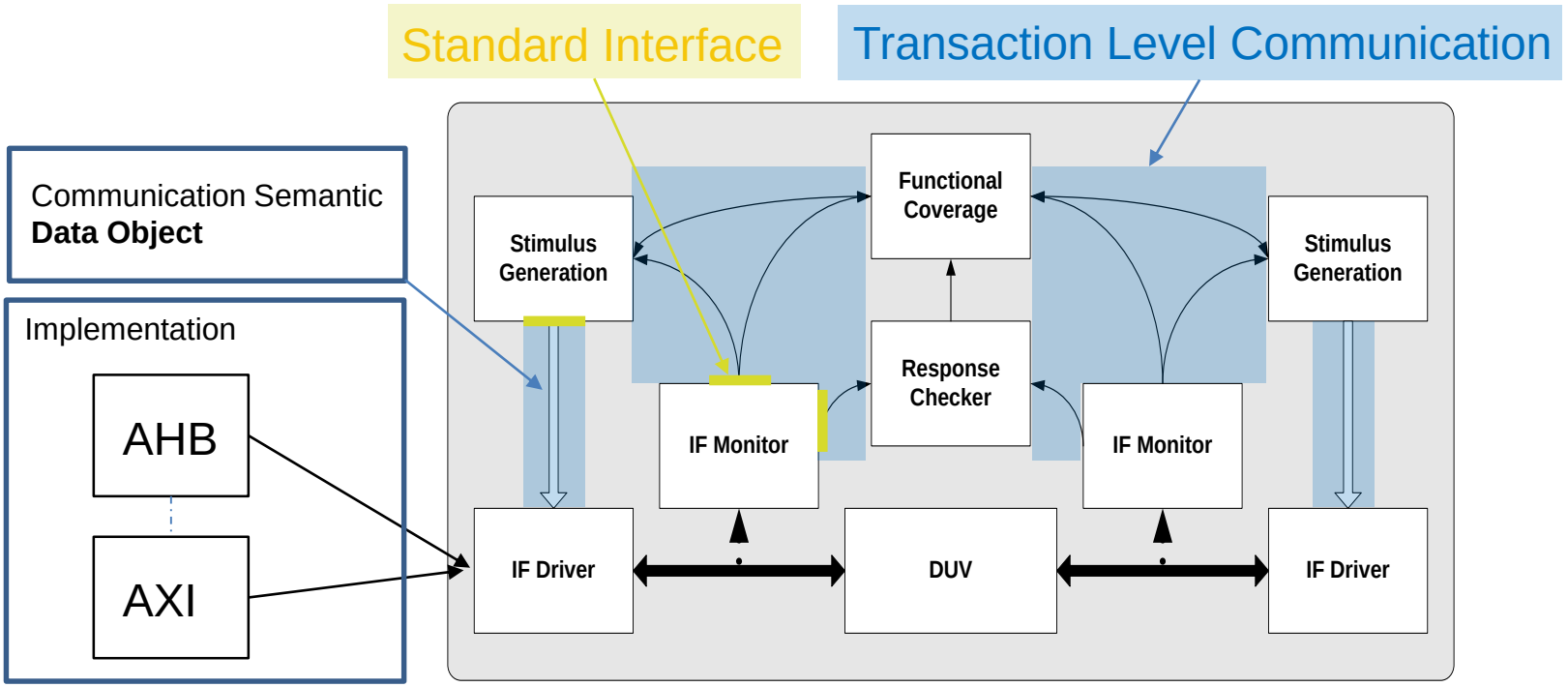
- Introduction
- GP extension follow up
- Proposals
  - Standard testbench clock interface
  - Standard testbench reset interface
- Discussion

© Accellera Systems Initiative

Introduction - Background

Problem: Interoperability between components and used generic payload

Testbench Infrastructure Overview



© Accellera Systems Initiative

Introduction – Data Object

Data object content was defined by

- Register Properties
  - Register Address
  - Register Size
  - Register Mode Properties

Mode examples

- User
- Supervisor
- Debug
- Test

Table 2.1. ARM processor modes

Processor mode	Architectures	Mode number
User	All	0b10000
FIQ - Fast Interrupt Request	All	0b10001
IRQ - Interrupt Request	All	0b10010
Supervisor	All	0b10011
Abort	All	0b10111
Undefined	All	0b11011
System	ARMv4 and above	0b11111
Monitor	Security Extensions only	0b10110

http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.dui0204j/Chdddhea.html

© Accellera Systems Initiative

Proposal “Mode” Support

Example of custom extension that should be added to the standard.

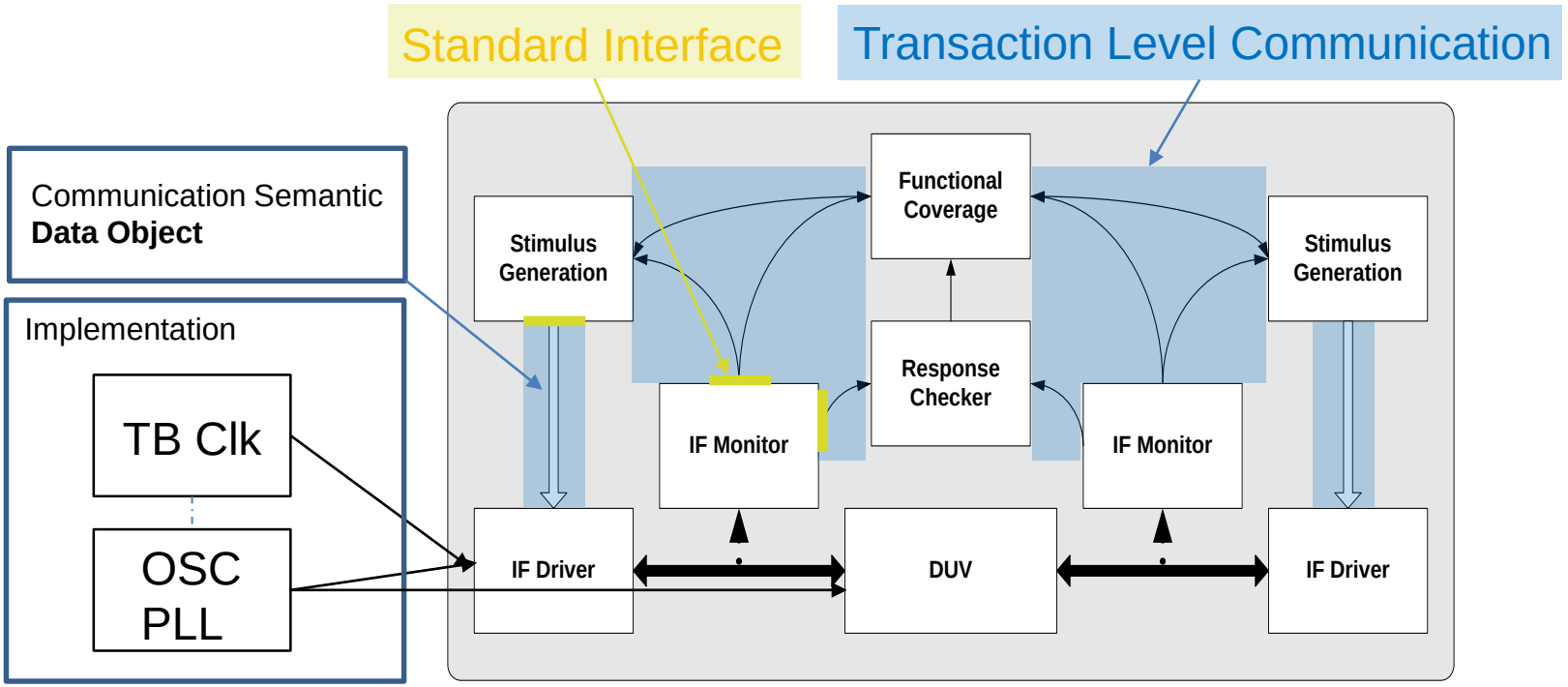
```
class nxp_acc_extension extends uvm_tlm_extension #(nxp_acc_extension);
    typedef enum { USER = 1, SUPERVISOR = 2, TEST = 4, DEBUG = 8 } access_mode_type_e;

    rand bit [31:0] access_mode = USER;
    rand bit [15:0] master_id = 'h0;

…

    function new(string name="nxp_acc_extension");
        super.new(name);
    endfunction: new

   constraint mode_constraint {
     access_mode inside {USER , SUPERVISOR , USER | TEST , USER | DEBUG ,
                  USER | TEST | DEBUG , SUPERVISOR | TEST ,
                  SUPERVISOR | DEBUG , SUPERVISOR | TEST | DEBUG } ;
  }

endclass : nxp_acc_extension
```


© Accellera Systems Initiative

Introduction - Background

Problem: Interoperability between components using clock

Testbench Infrastructure Overview



© Accellera Systems Initiative

Proposal “Clock”

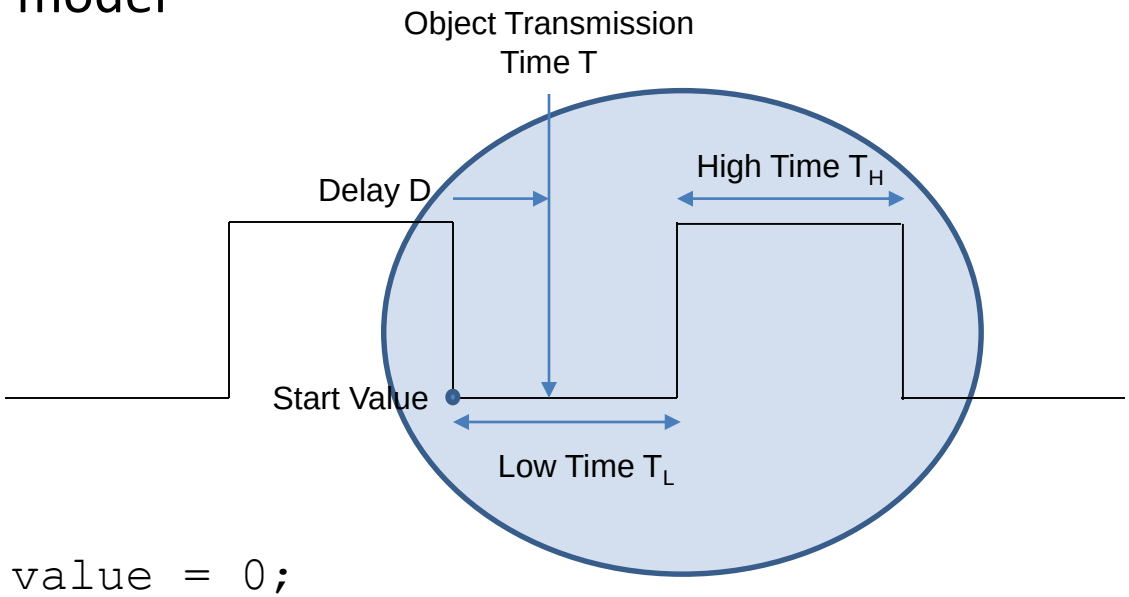
Example definition of clock transaction in the testbench.

```
class abs_clk_seq_item extends uvm_sequence_item;
    `uvm_object_utils(abs_clk_seq_item)

  typedef enum {NS=-9,US=-6,PS=-12,UNDEF=10} Timeunit_T;
  Timeunit_T  m_time_unit;

  // Value the clk started with
  rand bit m_start_value;

  // Delay of cmd transmition to ongoing clock
  rand bit[31:0] m_initial_delay;

  // The desired high time
  rand bit[31:0] m_high_time;

  // The desired low time
  rand bit[31:0] m_low_time;

  // Jitter of the clock signal
  rand bit[31:0] m_jitter;

…
endclass : abs_clk_seq_item
```


© Accellera Systems Initiative

Proposal “Clock”

Definition model

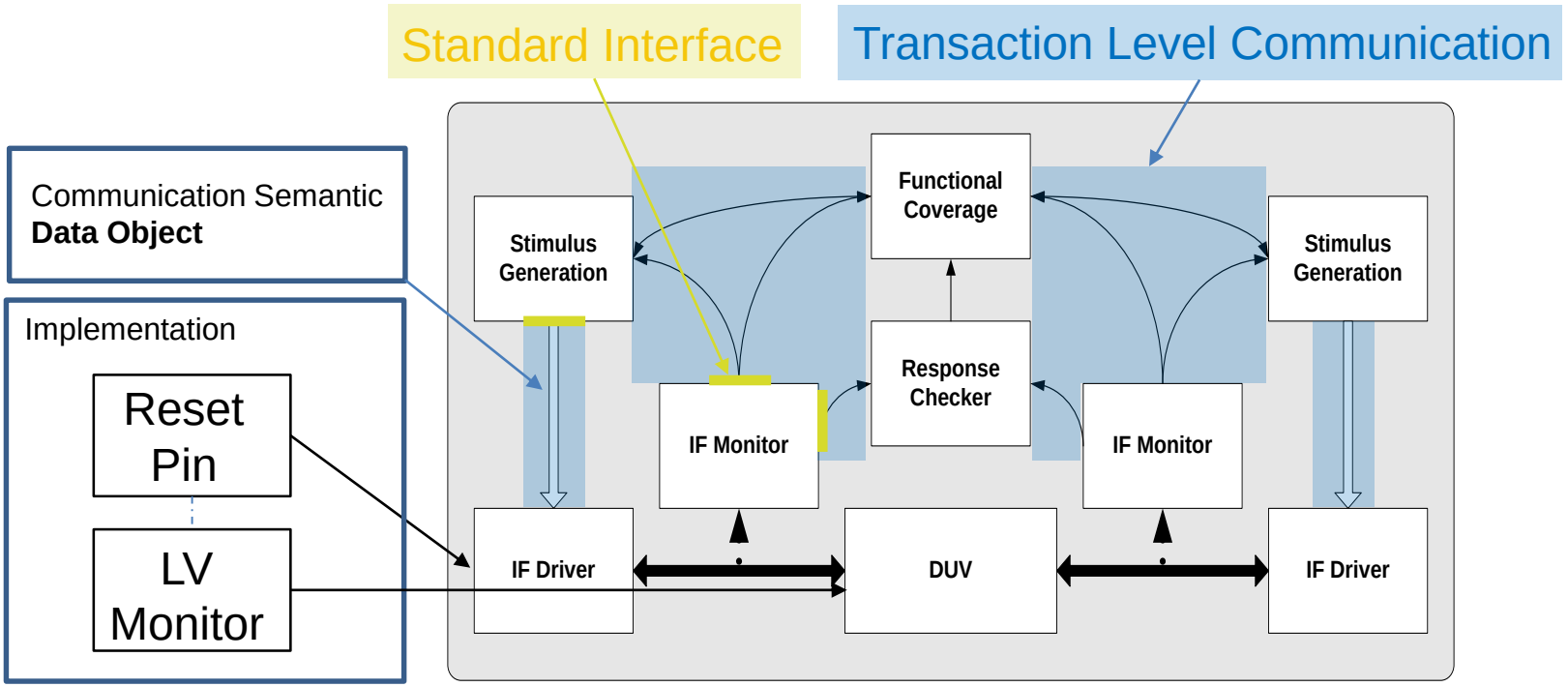



```
m_start_value = 0;
m_low_time = T_L;
m_high_time = T_H;
m_initial_delay = D;
m_jitter = 0;
```

Introduction - Background

Problem: Interoperability between components using reset

Testbench Infrastructure Overview



© Accellera Systems Initiative

Proposal “Reset”

Example definition of reset transaction in testbench.

```
class abs_reset_seq_item extends uvm_sequence_item;
    `uvm_object_utils(abs_reset_seq_item)

   //### Public Properties
   /*! Commands */
   typedef enum { ASSERT_WITH_DELAY = 1,    /*!< Assert with delay */
                  DEASSERT_WITH_DELAY = 2   /*!< Deassert with delay */
                } abs_reset_cmd_t;

   typedef enum { POR=1, HARD_RESET=2, SOFT_RESET = 4 } reset_type_t;

   rand bit [15:0]  m_cmd;                   /*! Reset command */
   rand bit [15:0]  m_reset_type;            /*! Reset type */
   rand int         m_delay_cycles;          /*! Delay cycles */
   int              m_duration;              /*! Duration of reset */
   int              m_reset_domain           /*! Reset domain identifier */
…
endclass : abs_reset_seq_item
```

`reset_type_t` is coded one hot for bit 0-2. Reusable register property will use these base definitions.

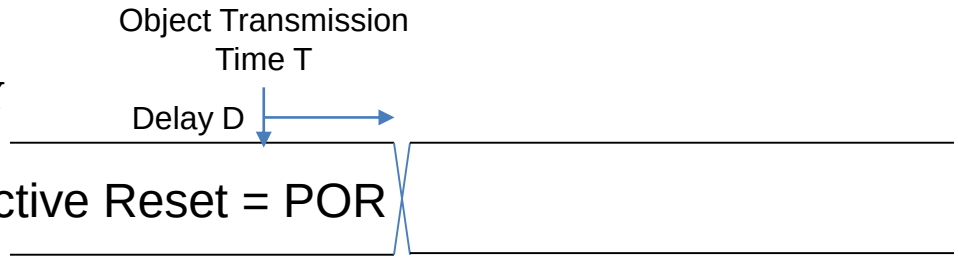
Detailed reset_types like low_power_reset, watch_dog reset can be coded in upper bits.

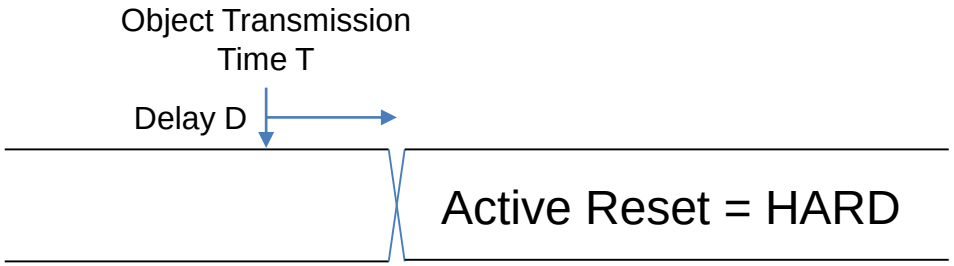
© Accellera Systems Initiative

Proposal “Reset”

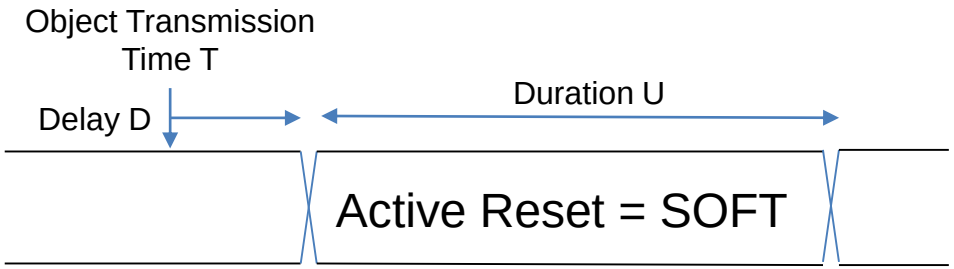
Definition model

```
m_cmd = DEASSERT_WITH_DELAY
m_reset_type = POR;
m_delay_cycles = D/clk;
m_duration = 0;
```






```
m_cmd = ASSERT_WITH_DELAY
m_reset_type = HARD;
m_delay_cycles = D/clk;
m_duration = 0;
```




```
m_cmd = ASSERT_WITH_DELAY
m_reset_type = HARD;
m_delay_cycles = D/clk;
m_duration = U/clk;
```


© Accellera Systems Initiative

Discussion