TLM – AMS Interaction
First Concepts

Karsten Einwich

Fraunhofer Institut for Integrated Circuits

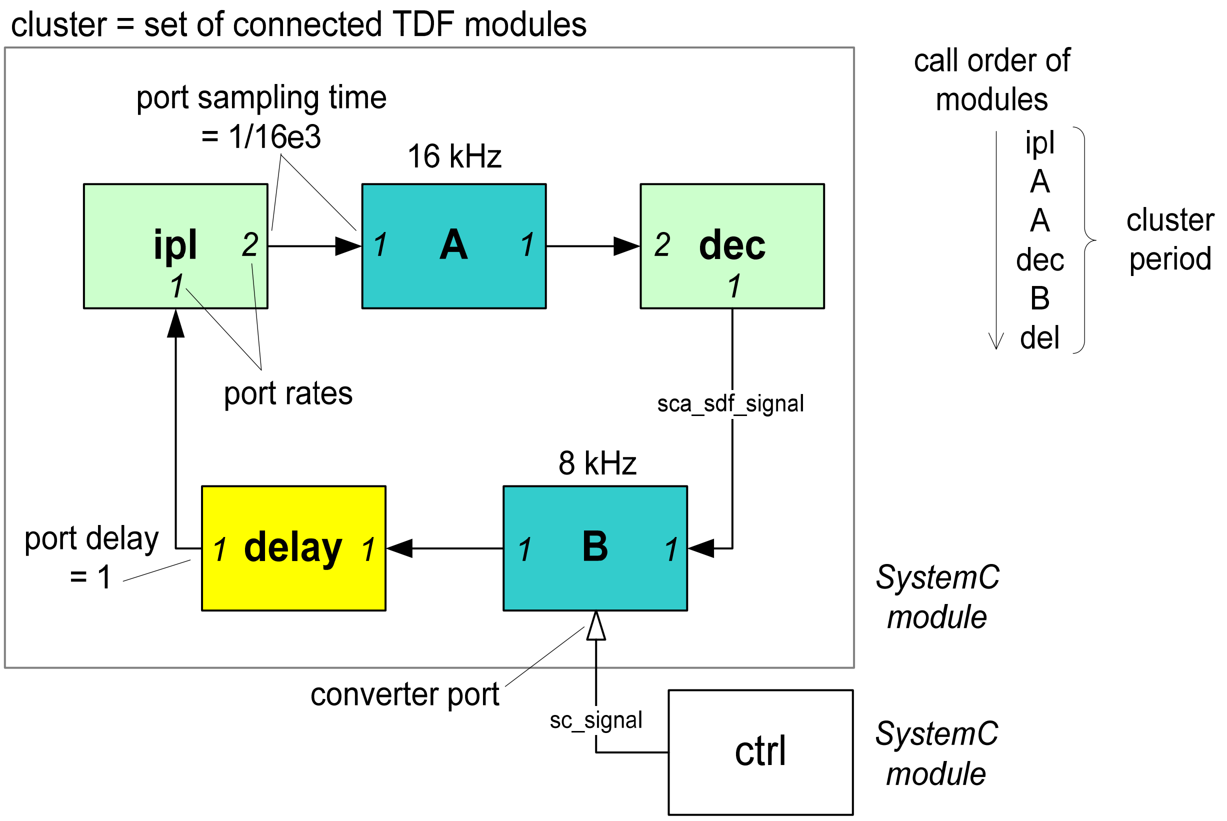
Design Automation Division

Fraunhofer Institut
Integrierte Schaltungen

Outline

- Current DE TDF synchronization
- TLM – AMS Interaction Semantic ??
- Challenges General TLM – AMS Interaction
- Proposal for simplification
- First proposal for language elements

Implementation of Multi-Rate TDF in SystemC-AMS



- The **first** sample is mapped to time **0s**
- Following samples mapped to **n * sampling_time**
- Converter ports read / write samples from/to SystemC always to the sdf-sample time points in the **first delta cycle** of the sample time point

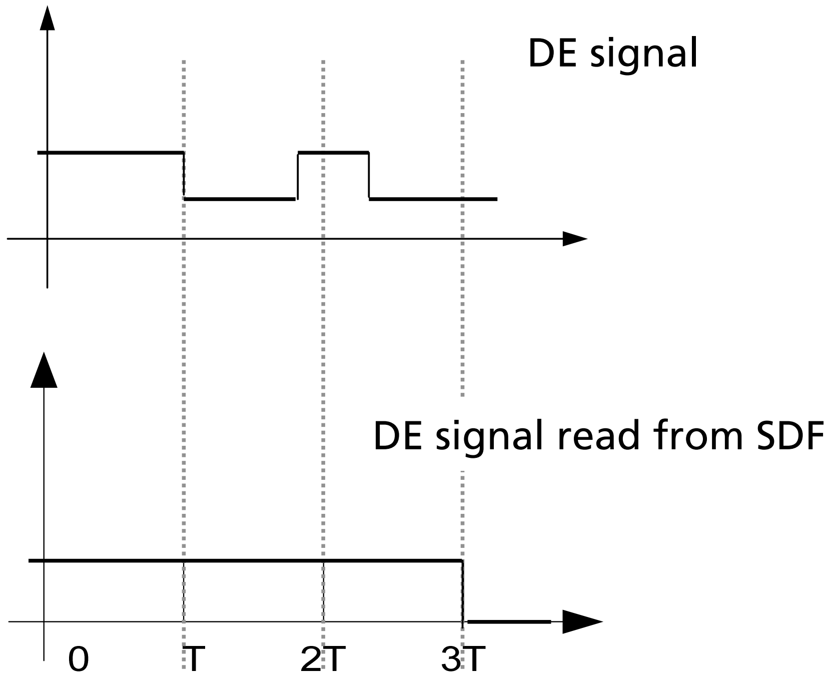
Synchronization between SDF and DE

- TDF samples are mapped to sc_time – the **first sample** has the time stamp **zero**.

- SystemC (DE) signals are **sampled** at **Δ=0** of the specified sampling period. TDF samples are **scheduled** at **Δ=0** as well (and thus valid at least at Δ=1).

- The sampling period T is specified as port attribute and propagated along the TDF signals of the cluster.

- That is why the sampling period must be specified at least for one port of a module in every TDF cluster – are ≥ 2 sampling periods given, the simulator performs a consistency check.



Principles for Synchronization

- Such loosely as possible
- Reducing context switching
- Synchronizing on data dependencies
- Synchronization only at discrete time points (currently in constant time steps)
- -> decoupled time scales
- Analogue runs before digital (however we have no backtracking, and may the next analogue time step not yet calculated)

What's the TLM-AMS Interaction Semantic

- Makes a timeless AMS sense ???
- -> TLM-AMS interaction is only possible with TLM/T ??
- -> or the AMS part annotates the time

- A TLM master accesses an analogue module and returns the current analogue value

- An analogue master accesses a TLM slave due an analogue event like threshold crossing

Synchronization Challenges for AMS-TLM

- A TLM master calls an module method to an arbitrary time point and accesses member variables
- Such an access can’t be detected by the AMS synchronization
- AMS synchronization means, that the equation system, which was setup by connected modules has to be synchronized
- In the general case a very tight synchronization will be required
- Thus we will loose the “abstraction” and especially the performance

Is the general approach really required?

- There is always a delay between analogue and digital
  - There are fifos
  - An A/D, D/A converter has a conversion time
  - …

- If we can use such delays we are able to decouple the time scales

- Curiosity: if we introduce implementation details we enable a fast simulation at high abstraction level

- The synchronization should be deterministic

First Proposal

- Introducing an synchronization variables (e.g. sca_tdf::ams2tlm, sca_tdf::tlm2ams)
- It is like an synchronization port – however no signal must be bound
- Has an attribute delay – must be at least one time step

TDF Module – Example with LTF

```
SCA_TDF_MODULE(prefi_ad) :  public virtual
          basic_slave_base< long , long >
{
   sca_tdf::sca_in<double>    in;    // signal inport

    typedef  tlm_transport_if … if_type;
   sc_export<if_type> slave_port;

   sca_tdf::ams2tlm<double>  a2d;
   sca_tdf::tlm2ams<bool>      fc_high;

                          :

   void attributes()
   {
     a2d.set_delay(1.0,SC_US);
     fc_high.set_delay(2.0,SC_US);
   }

  SCA_TDF_CTOR(prefi_ad)
  {
    slave_port(*this);
  }
```

```
basic_status write( const long& adr , const long& data)
{
  fc_high.write(bool(data)); //fc_high.read() not allowed
  return basic_protocol::SUCCESS;
}

basic_status read( const long& adr , const long& data)
{
  data = long(1024.0 * a2d.read()); //a2d.write not allowed
  return basic_protocol::SUCCESS;
}

void sig_proc() {
    double tmp;    // high or low cut-off freq.
    //fc_high.write not allowed
    if(fc_high.read())    tmp = ltf_1(b, a1, s, in.read());
    else                       tmp = ltf_0(b, a0, s, in.read());
     a2d.write(tmp);  //a2d.read not allowed
  }
};
```