

TRANSACTION GENERATOR 2 TECHNICAL

Updated: July 8, 2011
Lasse Lehtonen, Esko Pekkarinen
Department of Computer Systems
Tampere University of Technology

Contents

| | | |
|----------|--|-----------|
| 1 | MODELING FOR TRANSACTION GENERATOR 2 | 1 |
| 1.1 | XML document root | 3 |
| 1.2 | Application | 4 |
| 1.2.1 | Memory areas | 5 |
| 1.2.2 | Tasks | 6 |
| 1.2.3 | Triggers | 7 |
| 1.2.4 | Execution counts | 8 |
| 1.2.5 | Communication with memories | 9 |
| 1.2.6 | Polynomial and distributional amounts | 10 |
| 1.2.7 | Events | 11 |
| 1.3 | Mapping | 12 |
| 1.4 | Platform | 13 |
| 1.4.1 | Resources | 14 |
| 1.4.2 | NoC | 15 |
| 1.5 | Constraints | 16 |
| 2 | NETWORKS | 17 |
| 2.1 | Network interface | 17 |
| 2.2 | Adapters | 17 |
| 2.2.1 | OCF TL3 adapter | 18 |
| 2.2.2 | OSCI TLM 2.0 adapter | 18 |
| 3 | EXAMPLE NETWORKS | 19 |
| 3.1 | Simple bus | 19 |
| 3.2 | 2-D mesh | 19 |
| 4 | XML TAGS LISTING | 21 |
| 4.1 | <system> | 21 |
| 4.2 | <application> | 22 |
| 4.3 | <service> | 23 |
| 4.3.1 | service's <task> | 24 |
| 4.4 | <task_graph> | 25 |
| 4.5 | <task> | 26 |
| 4.6 | <mem_area> | 27 |
| 4.6.1 | <in_port> | 28 |
| 4.6.2 | <out_port> | 29 |
| 4.6.3 | <trigger> | 30 |
| 4.6.4 | trigger's <in_port> | 31 |
| 4.6.5 | <exec_count> | 32 |
| 4.6.6 | <op_count> | 33 |
| 4.6.7 | <int_ops>, <float_ops>, <mem_ops>, <byte_amount>, <burst_amount> | 34 |
| 4.6.8 | <polynomial> | 35 |
| 4.6.9 | <param> | 36 |
| 4.6.10 | <distribution> | 37 |
| 4.6.11 | <uniform> | 38 |
| 4.6.12 | <normal> | 39 |
| 4.6.13 | <send> | 40 |
| 4.6.14 | <read> | 41 |
| 4.6.15 | <next_state> | 42 |
| 4.7 | <port_connection> | 43 |
| 4.8 | <event_list> | 44 |
| 4.8.1 | <event> | 45 |

| | | |
|---------|-------------------------|----|
| 4.9 | <mapping> | 46 |
| 4.9.1 | <resource> | 47 |
| 4.9.2 | <sw_platform> | 48 |
| 4.9.3 | <group> | 49 |
| 4.9.4 | group's <task> | 50 |
| 4.9.5 | resource's <mem_area> | 51 |
| 4.10 | <platform> | 52 |
| 4.10.1 | <resource_list> | 53 |
| 4.10.2 | <resource> | 54 |
| 4.10.3 | resource's <port> | 55 |
| 4.10.4 | <cache> | 56 |
| 4.10.5 | <i_cache> and <d_cache> | 57 |
| 4.10.6 | <parameter> | 58 |
| 4.10.7 | <noc> | 59 |
| 4.10.8 | <router_list> | 60 |
| 4.10.9 | <router> | 61 |
| 4.10.10 | router's <port> | 62 |
| 4.10.11 | <link_list> | 63 |
| 4.10.12 | <link> | 64 |
| 4.10.13 | <terminal_list> | 65 |
| 4.10.14 | <connection> | 66 |
| 4.10.15 | <network_interface> | 67 |
| 4.11 | <constraints> | 68 |
| 4.11.1 | <rng_seed> | 69 |
| 4.11.2 | <sim_resolution> | 70 |
| 4.11.3 | <sim_length> | 71 |
| 4.11.4 | <measurements> | 72 |
| 4.11.5 | <path_measurement> | 73 |
| 4.11.6 | <exec_mon> | 74 |
| 4.11.7 | <pe_lib> | 75 |
| 4.11.8 | <log_*> | 76 |
| 4.11.9 | <cost_function> | 77 |

1 MODELING FOR TRANSACTION GENERATOR 2

Workload model for Transaction Generator is described in eXtensible Markup Language (XML) which can be hand written or generated by a program. Transaction Generator 2 parses automatically the XML source file before simulation and modifying XML source doesn't need a recompilation. Figure 1 presents the major tags used in modeling and figure 2 expands the `<task>` tag from figure 1.

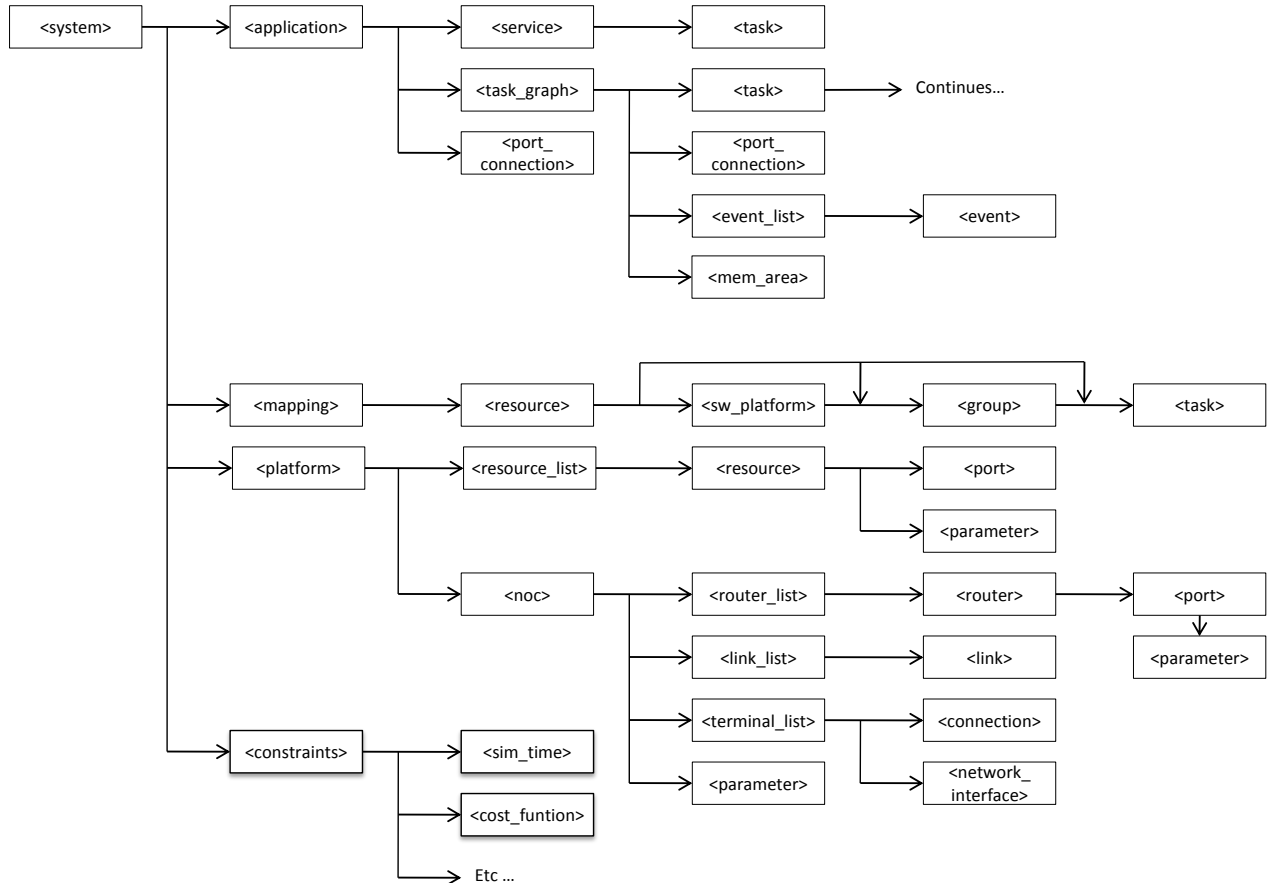


Figure 1: XML tags used in modeling, part 1

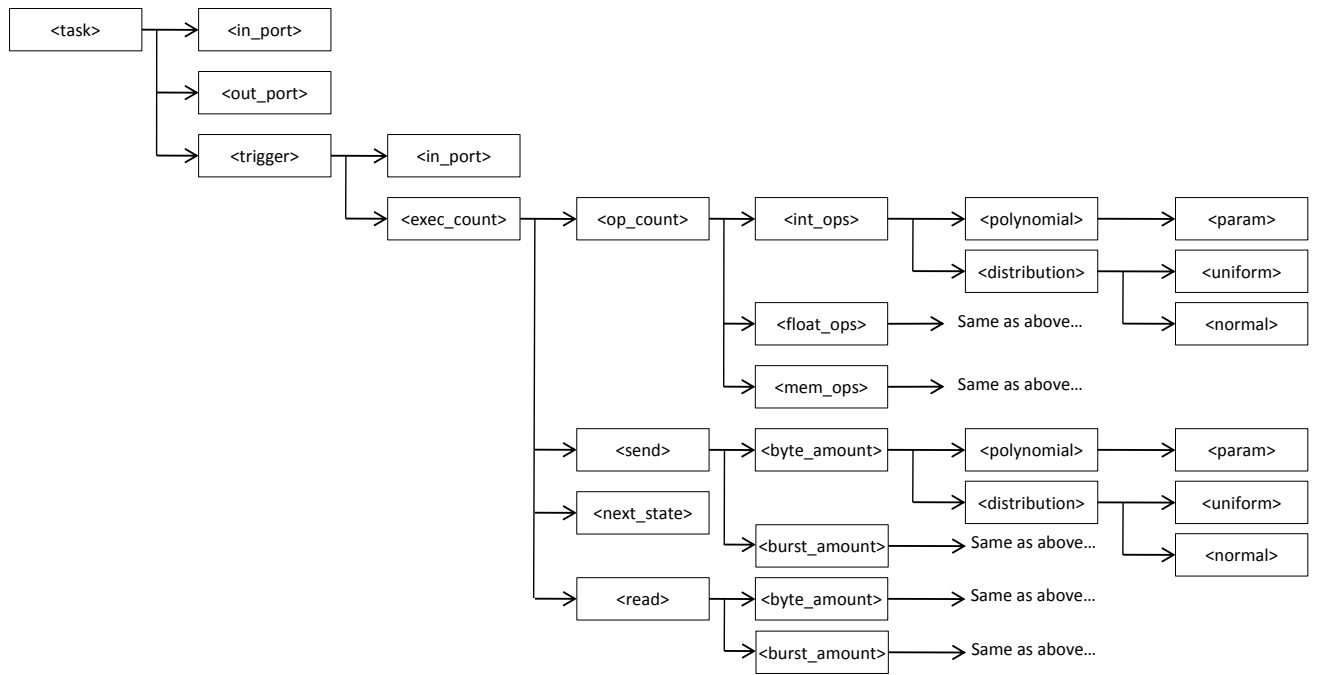


Figure 2: XML tags used in modeling, part 2

1.1 XML document root

Example code below shows the fundamental elements that must be present in the source file. Following sections describe the tag structure for <application>, <mapping>, <platform> and <constraints> tags in detail. Order of these tags inside <system> is free but they must be defined only once.

Listing 1: *Root*

```
<?xml version='1.0'?>
<!DOCTYPE system>

<system xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:noNamespaceSchemaLocation="...">
  <xsm_version value="4"/>

  <application>
    .
    .
    .
  </application>

  <mapping>
    .
    .
    .
  </mapping>

  <platform>
    .
    .
    .
  </platform>

  <constraints>
    .
    .
    .
  </constraints>

</system>
```

1.2 Application

Application model must have at least one `<task_graph>` which contains task models, memory area definitions, the connections between those and events to start the application.

`<port_connection>` tags are used to connect events to tasks and tasks and memory areas together to form the graph that represents application's communicational dependencies.

Listing 2: *Application*

```
<application>
  <task_graph>

    <task name="Task_0" id="0" class="general">
      <in_port id="7"/>
      <out_port id="0"/>
      .
    </task>
    .
    .
    <mem_area name="Memory_n" id="9" size="12030" class="general">
      .
    </mem_area>

    <port_connection src="0" dst="1"/>
    .
    .
    <port_connection src="8" dst="9"/>

    <event_list>
      <event out_port_id="6" amount="1" name="Event_0"
        id="0" period="0.1" prob="1"/>
      .
      .
      <event out_port_id="8" amount="1" name="Event_5"
        id="5" period="0.05" prob="1"/>
    </event_list>

  </task_graph>
</application>
```

1.2.1 Memory areas

Memory areas define a region of contiguous memory to model for example executable instructions or a shared data memory area. Memory areas have ID numbers which must be unique with tasks. Areas size is defined in bytes. For every task which writes data to the memory area an in port is needed and for every reading task also a out port.

Listing 3: *Memory area*

```
<mem_area name="mem_area1" id="42" size="1024" class="general">
  <in_port id="13"/>
  <in_port id="14"/>
  <out_port id="3"/>
</mem_area>
```


1.2.2 Tasks

<task> tags inside <task_graph> are used to model the behavior of tasks. Tasks may contain any number of input or output ports and triggers which defines how to react to incoming data tokens. Tasks communicate through unidirectional ports with each other.

Id numbers of input and output ports must be unique in the model.

Listing 4: *Task*

```
<task name="Task0" id="0" class="general">
  <in_port id="5"/>
  <in_port id="7"/>
  <out_port id="0"/>

  <trigger>
    <in_port id="5"/>
    <exec_count>
      <op_count>
        <int_ops>
          <polynomial>
            <param value="10000" exp="0"/>
          </polynomial>
        </int_ops>
      </op_count>
      <send out_id="0" prob="1">
        <byte_amount>
          <polynomial>
            <param value="1024" exp="0"/>
          </polynomial>
        </byte_amount>
      </send>
      <next_state value="READY"/>
    </exec_count>
  </trigger>

  <trigger>
    <in_port id="7"/>
    <exec_count>
      <op_count>
        <int_ops>
          <polynomial>
            <param value="2000" exp="0"/>
          </polynomial>
        </int_ops>
      </op_count>
      <next_state value="READY"/>
    </exec_count>
  </trigger>
</task>
```

Listing 4 models a simple task that has two input ports and one output port. When task receives a data token to port 5 it first executes 10,000 integer operations and then sends a 1 KB data token to port 0. Tokens received to port 7 triggers task to execute 2,000 integer operations but not to send anything.

1.2.3 Triggers

Triggers are evaluated when all or one of the input ports have received a full data token. If trigger's `dependence_type` attribute is "and" it's evaluated after all ports have received data and if it's "or" it's evaluated when a data token is received to any of the trigger's input ports.

Triggers must have at least one input port and one `exec_count` tag.

Listing 5: *Trigger*

```
<trigger>
  <in_port id="7"/>
  <in_port id="8"/>
  <exec_count>
    <op_count>
      <int_ops>
        <polynomial>
          <param value="400" exp="0"/>
        </polynomial>
      </int_ops>
    </op_count>
    <send out_id="2" prob="1">
      <byte_amount>
        <polynomial>
          <param value="1024" exp="0"/>
        </polynomial>
      </byte_amount>
    </send>
    <send out_id="1" prob="0.5">
      <byte_amount>
        <polynomial>
          <param value="256" exp="0"/>
        </polynomial>
      </byte_amount>
    </send>
    <next_state value="READY"/>
  </exec_count>
</trigger>
<trigger dependence_type="and">
  <in_port id="10"/>
  <in_port id="11"/>
  <in_port id="12"/>
  <exec_count>
    .
    .
  </exec_count>
</trigger>
```

First trigger is evaluated whenever there is a data token in port 7 or 8. It executes 400 integer operations and after that sends 1024 bytes to port 2 and 256 bytes to port 1 with 50% probability.

Second trigger is evaluated only after all ports 10, 11 and 12 have received data tokens. If `dependence_type` attribute is not defined "or" is assumed.

1.2.4 Execution counts

Amount of operations trigger executes and bytes it sends can be different depending how many times it has been executed (triggered). Conditions of `exec_count`s are evaluated in the order they appear in the source file and all that have a condition which evaluates as true are executed.

Attribute `mod_period` is used to model periodical behavior e.g. for tasks that does every tenth time it's executed something differently. Attributes `min` and `max` are used to select a range of this period and using `mod_phase` is identical to using same value for both `min` and `max`.

Listing 6: *Execution counts (1)*

```
<trigger>
  <exec_count mod_phase="0" mod_period="3">
    ...
  </exec_count>
  <exec_count mod_phase="1" mod_period="3">
    ...
  </exec_count>
  <exec_count <!-- always executed -->
    ...
  </exec_count>
</trigger>
```

Trigger in the first example executes the first `exec_count` when task's execution count modulo three is zero. Second `exec_count` is executed when the modulo returns 1 and the third `exec_count` will be executed every time.

Listing 7: *Execution counts (2)*

```
<trigger>
  <exec_count min="1" max="4" mod_period="5">
    ...
  </exec_count>
  <exec_count min="0" max="0" mod_period="5">
    ...
  </exec_count>
</trigger>
```

Second trigger example demonstrates the use of `min` and `max`. First `exec_count` is executed when task's execution count modulo five is greater than zero and the second `exec_count` when the result is zero.

Listing 8: *Execution counts (3)*

```
<trigger>
  <exec_count mod_phase="0"> ... </exec_count>
  <exec_count mod_phase="1"> ... </exec_count>
  <exec_count mod_phase="3"> ... </exec_count>

  <exec_count min="10"> ... </exec_count>
  <exec_count max="42"> ... </exec_count>
</trigger>
```

It's possible also to define a different behavior for every time the task is executed by not defining attribute `mod_period`. Using only `min` or `max` is also possible.

1.2.5 Communication with memories

Communication with memories utilizes two tags to write and read from the memory, <send> and <read>. Send to memory differs from sending to another task directly by allowing to define a write burst length. Burst length defines how many bytes are sent to contiguous addresses or in other words how often the memory needs to change the address which may affect greatly to the performance, for example by triggering the row change in DRAM.

Listing 9: *Writing to and reading from memory*

```
<send out_id="4" prob="1">
  <byte_amount>
    <polynomial>
      <param value="320000" exp="0"/>
    </polynomial>
  </byte_amount>

  <burst_length>
    <distribution>
      <uniform min="16" max="23"/>
    </distribution>
  </burst_length>
</send>

<read out_id="2" resp_id="12" prob="1">
  <burst_length>
    <distribution>
      <uniform min="4" max="23"/>
    </distribution>
  </burst_length>

  <byte_amount>
    <polynomial>
      <param value="100" exp="0"/>
    </polynomial>
  </byte_amount>
</read>
```

For reading burst length defines how often read address must be changed to other than the next. Size of the request packet varies by the memory or its controller and is defined where the memory model is defined.

1.2.6 Polynomial and distributional amounts

Listing 10 shows few different ways to determine the amount of operations to execute on HW resource or the amount of bytes to send when a certain trigger is being evaluated. In the example code amount of integer operations is $10x^2 + 100x + 400$ where x is the amount of bytes received. Number of floating point operations is a random number in range of 30 to 90 with uniform distribution. Amount of memory operations is randomly chosen using normal distribution. If the mean attribute is not defined for a normal distribution as in the second <send>, the amount of bytes received is used as a mean.

Listing 10: *Execution and send amounts*

```
<op_count>
  <int_ops>
    <polynomial>
      <param value="400" exp="0"/>
      <param value="100" exp="1"/>
      <param value="10" exp="2"/>
    </polynomial>
  </int_ops>
  <float_ops>
    <distribution>
      <uniform min="30" max="90"/>
    </distribution>
  </float_ops>
  <mem_ops>
    <distribution>
      <normal mean="50" standard_deviation="5"/>
    </distribution>
  </mem_ops>
</op_count>

<send out_id="2" prob="1">
  <byte_amount>
    <polynomial>
      <param value="1024" exp="0"/>
    </polynomial>
  </byte_amount>
</send>

<send out_id="2" prob="1">
  <byte_amount>
    <distribution>
      <normal standard_deviation="100"/>
    </distribution>
  </byte_amount>
</send>
```

1.2.7 Events

Events are used to trigger the execution of task models either n times or periodically. At least one event must exist to start the workload model to execute as all tasks are initially waiting for tokens to trigger them and thus the model wouldn't do anything without an event to trigger at least one of the tasks.

Listing 11: *Events*

```
<event_list>

  <event id="7"
    out_port_id="1"
    amount="1"
    name="Event0"
    count="7"
    offset="0.3"
    period="0.1"
    prob="0.3"/>

  <event id="8"
    out_port_id="2"
    amount="20"
    name="Event1"
    count="1"
    offset="1.0"
    prob="1"/>

  <event id="9"
    out_port_id="3"
    amount="2"
    name="Event2"
    period="0.2"
    prob="1"/>

</event_list>
```

In the example “Event0” sends 1 byte data token to port 1 seven times every 0.1 seconds with 30% probability. Attribute offset defines when the first byte is sent (default value is zero). “Event1” sends only once 20 bytes to port 2 when simulation has been run for 1.0 seconds with 100% probability. “Event2” sends two bytes to port 3 every 0.2 seconds. Its first token is sent when simulation starts.

1.3 Mapping

Mapping section binds software platforms to HW resources, group of tasks to either software platforms or directly to the resources and finally tasks to groups. Attributes “contents” and “position” are used in automatic design space exploration to determine whether or not the optimization algorithm can e.g. move groups to different resources or alter the contents of groups by moving tasks.

Listing 12: *Mapping*

```
<mapping>

  <resource      name="cpu0"          id="0"  contents="mutable">
    <sw_platform position="movable"  id="0"  contents="mutable">
      <group      position="movable"  id="0"  contents="mutable"
        name="group0">

        <task position="movable"  id="0"  name="Task0"/>
        <task position="movable"  id="1"  name="Task1"/>
        <task position="movable"  id="2"  name="Task2"/>

      </group>
      <group      position="movable"  id="1"  contents="immutable"
        name="group1">

        <task position="immovable" id="3"  name="Task3"/>

      </group>
    </sw_platform>
  </resource>

  <resource name="Memory1"          id="1"  contents="immutable">

    <mem_area position="immovable" id="4"  name="MemoryArea4"/>

  </resource>

  .
  .
  .

</mapping>
```

In the example “cpu0” models a general purpose processor with a software platform and two groups of tasks. First group could be altered in any way in the exploration but the second could only be moved to another resource but not changed. Resource “Memory1” has only one memory area mapped and it can’t be altered in any way.

1.4 Platform

Platform section defines the type of hardware resources, the type of NoC and how resources are connected to the NoC.

Listing 13: *Platform*

```
<platform>
  <resource_list>

    <resource id="0" name="cpu0" frequency="100" type="Generic_CPU">
      <port terminal="0"/>
    </resource>

    .
    .

    <resource id="9" name="acc9" frequency="100" type="Accelerator_x">
      <port terminal="9"/>
    </resource>

  </resource_list>

  <noc type="hibi">

    <router_list>
      <router width="32" id="0" name="Hibi_segment" frequency="80"
        type="Hibi_segment">
        <port name="hibi_p1" id="0" type="Hibi_if" address="0x100000"/>
        .
        .
        <port name="hibi_p9" id="9" type="Hibi_if" address="0x900000"/>
      </router>
    </router_list>

    <terminal_list>
      <connection port="0" router="0" name="hibi_p" id="0"/>
      .
      .
      <connection port="9" router="0" name="hibi_p" id="9"/>
      <network_interface type="Hibi_if"/>
    </terminal_list>

  </noc>
</platform>
```


1.4.1 Resources

For PEs resource's type attribute maps it to an external HW library which defines its characteristics e.g. how many integer and floating point operations it can execute in one clock cycle. Port tags determines the network terminal which the resource is connected to.

Optional buffer size attributes can be used to specify the amount of memory resource has to limit its capability to generate or receive tokens. Receive buffer is also used for inter PE traffic. Transaction Generator 2 can automatically split large tokens into smaller packets if packet_size attribute is defined.

For memory resources also request size must be defined. It defines how many bytes PEs must sent for every read access. Also two additional memory model configuration files must be set.

Optional cache section defines how often the processing element misses for instructions and data. Line size defines how many bytes are fetched per cache miss from the memory resource identified by the mem_area_id attribute.

Listing 14: *Resources*

```
<resource_list>

  <resource id="0" name="cpu0" frequency="80" type="Generic_CPU"
    rx_buffer_size="262144" tx_buffer_size="1024"
    packet_size="16">
    <port terminal="0"/>

    <cache>
      <i_miss line_size="64" mem_area_id="2">
        <distribution>
          <uniform min="1000" max="2000"/>
        </distribution>
      </i_miss>
      <d_miss line_size="64" mem_area_id="3">
        <distribution>
          <normal mean="100000" standard_deviation="20000"/>
        </distribution>
      </d_miss>
    </cache>
  </resource>

  <resource id="1" name="mem1" frequency="200" class="memory"
    rx_buffer_size="262144" tx_buffer_size="1024"
    packet_size="16" request_size="8"
    ocp_param="examples/ocpParams" adm_param="examples/admParams">
    <port terminal="1"/>
  </resource>

</resource_list>
```

1.4.2 NoC

This tag describes how network on chip models should construct themselves or how they are constructed if the used NoC model can't use this information. Attributes class, type and subtype are used to determine which NoC TG uses during simulation. Transaction Generator 2 parses and uses only router widths and port addresses from this information. All other information can be used in NoC model or ignored altogether.

Example code models a HIBI bus with two segments bridged together. Router list defines all the routers or bus segments as in this case and the ports they have. Terminal list connects routers' ports to resources' terminals and link list determines the connections between routers.

Listing 15: *NoC*

```
<noc class="hibi" type="hibi_simple" subtype="">

  <router_list>
    <router width="32" id="0" name="Hibi_segment0" frequency="25"
      type="Hibi_segment">

      <port name="hibi0_p0" id="0" type="Hibi_if" address="0x1000000"/>
      <port name="hibi0_p1" id="1" type="Hibi_if" address="0x2000000"/>
      <port name="hibi0_p2" id="2" type="Hibi_if" address="0x3000000"/>

    </router>
    <router width="32" id="1" name="Hibi_segment1" frequency="25"
      type="Hibi_segment">

      <port name="hibi1_p0" id="0" type="Hibi_if" address="0x4000000"/>
      <port name="hibi1_p1" id="1" type="Hibi_if" address="0x5000000"/>
      <port name="hibi1_p2" id="2" type="Hibi_if" address="0x6000000"/>

    </router>
  </router_list>

  <terminal_list>
    <connection port="0" router="0" name="hibi_p" id="0"/>
    <connection port="1" router="0" name="hibi_p" id="1"/>
    <connection port="2" router="0" name="hibi_p" id="2"/>
    <connection port="0" router="1" name="hibi_p" id="3"/>
    <connection port="1" router="1" name="hibi_p" id="4"/>
    <connection port="2" router="1" name="hibi_p" id="5"/>
    <network_interface type="Hibi_if"/>
  </terminal_list>

  <link_list>
    <link id="0" src_router="0" dst_router="1" src_port="2" dst_port="2"/>
  </link_list>

</noc>
```

1.5 Constraints

Constraints define parameters for Transaction Generator 2, such as simulation resolution and length and various file names for logging measurements gathered during simulation.

Listing 16: *Constraints*

```
<constraints>
  <!-- Seed for random number generator, comment out for random seed -->
  <rng_seed      value="42"/>

  <!-- Used in standalone mode -->
  <sim_resolution time="1.0"    unit="fs"/>
  <sim_length     time="1.0"    unit="ms"/>

  <!-- interval between measurements
        (both averages and snapshots) -->
  <measurements  time="2.0"    unit="ms"/>

  <!-- Path to PE lib file -->
  <pe_lib        file="examples/pe_lib.xml"/>

  <!-- Paths to log files, comment out to disable logging -->
  <log_packet    file="log_packet.txt"/>
  <log_token     file="log_token.txt"/>
  <log_summary   file="log_summary.txt"/>
  <log_pe        file="log_pe.txt"/>
  <log_app       file="log_app.txt"/>
</constraints>
```

2 NETWORKS

2.1 Network interface

TG uses a hierarchy of factories (Figure 3) to dynamically instantiate the actual NoC model. Multiple hierarchy levels speed up the compilation when some parameters are changed that requires the recompilation of only some of the NoC models.

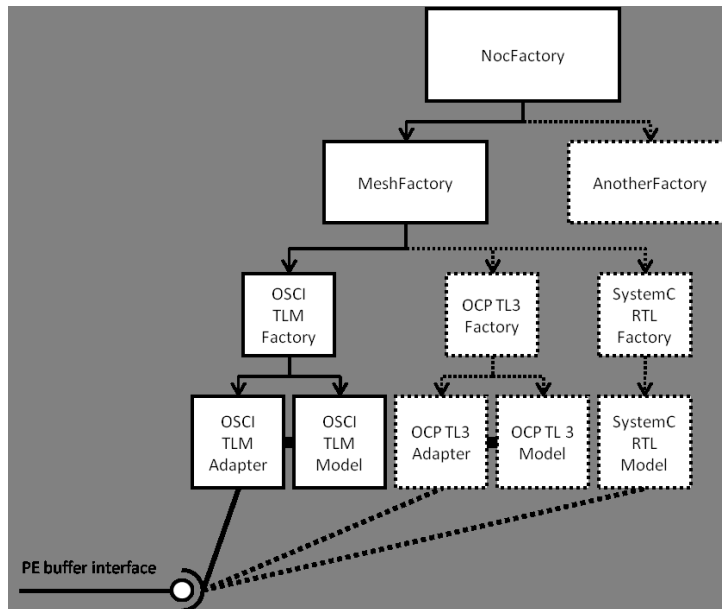


Figure 3: NoC factory pattern

The actual factories to construct depends on the attributes given to `<noc>` tag. Class `NocFactory` is always constructed and will dynamically construct some other factory depending on the class attribute. `NocFactory` class implementation (`tg_root/hw_lib/noc_factory/noc_factory.cc`) needs to be modified if one wants to attach other NoC models to TG.

```
<system>
...
<platform>
...
  <noc class="mesh_2d" type="sc_rtl_1" subtype="2x2"> ... </noc>
</platform>
...
</system>
```

2.2 Adapters

TG offers a simple custom TL interface for NoCs and adapters for OCP TLM2 Kit TL3 sockets and OSCI TLM 2.0 sockets. These adapters will not serve all NoC models and are intended to be more of an examples of how to adapt TG for various models. Both adapters annotate no time for communication and will only stall if the PE they are connected to does not have enough space in its receive buffer. Adapters support only writing as TG is designed for NoCs with split transactions where the path of read response can be different from the read request path.

2.2.1 OCP TL3 adapter

Adapter class has template parameters for the amount of agents and the bus width and uses two multi-sockets (master and slave). Sockets should be bound in the order of PE id numbers (first bind PE with id 0, then 1, etc).

```
template <unsigned int N_AGENTS, unsigned int DATA_WIDTH>
class SctgToOcpTl3 : public sc_core::sc_module
{
    public:

    ocpip::ocp_master_socket_tl3<DATA_WIDTH, 0>* masterSocket;
    ocpip::ocp_slave_socket_tl3<DATA_WIDTH, 0>* slaveSocket;

};
```

2.2.2 OSCI TLM 2.0 adapter

This adapter uses tagged sockets and has a pair of initiator and target sockets per PE. Again PEs should be bound in order of their id numbers (Socket[0] is connected to PE with id 0, etc). Adapter uses two timing points for transactions.

```
template <unsigned int N_AGENTS, unsigned int DATA_WIDTH>
class SctgToOsciTlm : public sc_core::sc_module, public tlm::tlm_mm_interface
{
    public:

    tlm_utils::simple_initiator_socket_tagged
    <SctgToOsciTlm, DATA_WIDTH> initSockets[N_AGENTS];

    tlm_utils::simple_target_socket_tagged
    <SctgToOsciTlm, DATA_WIDTH> targetSockets[N_AGENTS];

};
```

3 EXAMPLE NETWORKS

The TG package includes currently two example networks: simple bus and 2-D mesh. Their properties are introduced briefly in this section.

3.1 Simple bus

An untimed shared bus model. Port address must match the PE id. By default supports 4, 9, 16, 25, 36 or 64 agents.

```
<noc class="simple_bus" type="sc_tlm_1" subtype="4_agents">

  <router_list>
    <router width="32" id="0" name="r1" frequency="50" type="">
      <port name="p1" id="0" type="ptk_if" address="0x00000000"/>
      <port name="p2" id="1" type="ptk_if" address="0x00000001"/>
      <port name="p3" id="2" type="ptk_if" address="0x00000002"/>
      <port name="p4" id="3" type="ptk_if" address="0x00000003"/>
    </router>
  </router_list>

  <terminal_list>
    ...
  </terminal_list>

</noc>
```

| Class | Type | Subtype | Notes |
|------------|----------|----------|---|
| simple_bus | sc_tlm_1 | n_agents | OSCI TLM 2.0 nearly untimed shared bus model with two timing points |

3.2 2-D mesh

The 2-D mesh included in TG package is depicted in figure 4. Every IP is connected to a single router which in turn connects up to four adjacent routers. The IP's network interface has a packet codec that encodes the sent data for the network and decodes the incoming data for the IP.

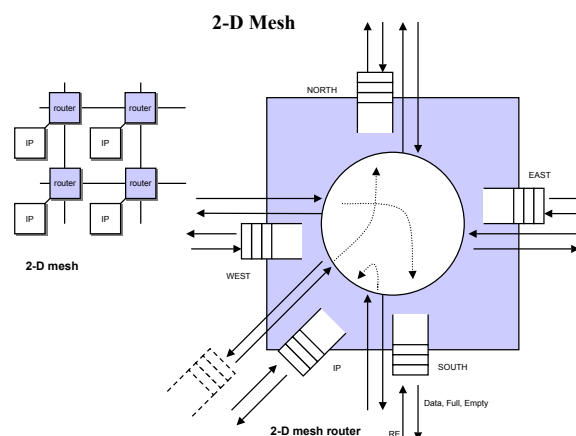


Figure 4: 2-D mesh network and router overview

The first four hexes of the address define the row number and the last four hexes the column number.

```
<noc class="mesh_2d" type="sc_rtl1_1" subtype="2x2">

  <router_list>
    <router width="32" id="0" name="mesh_r1" frequency="50" type="mesh_router">
      <port name="mesh_p1" id="0" type="ptk_if" address="0x00000000"/>
    </router>
    <router width="32" id="1" name="mesh_r2" frequency="50" type="mesh_router">
      <port name="mesh_p2" id="1" type="ptk_if" address="0x00000001"/>
    </router>
    <router width="32" id="2" name="mesh_r3" frequency="50" type="mesh_router">
      <port name="mesh_p3" id="2" type="ptk_if" address="0x00010000"/>
    </router>
    <router width="32" id="3" name="mesh_r4" frequency="50" type="mesh_router">
      <port name="mesh_p4" id="3" type="ptk_if" address="0x00010001"/>
    </router>
  </router_list>

  <terminal_list>
    ...
  </terminal_list>

</noc>
```

| Class | Type | Subtype | Notes |
|---------|---------------|--------------|---|
| mesh_2d | vhd | $N \times M$ | VHDL model, usage requires a simulator capable of mixed language simulation |
| | sc_rtl1 | $N \times M$ | SystemC RTL model with 4-state variables |
| | sc_rtl2 | $N \times M$ | SystemC RTL model with 2-state variables |
| | sc_ocp_tlm3_1 | $N \times M$ | OCPTLM2 TL3 model |
| | sc_tlm1 | $N \times M$ | OSCI TLM 2.0 AT model using two timing points for transactions |

The following table lists the RTL model template parameters one can change in the source code.

| Name | Meaning | Notes |
|-----------------|---------------------------------------|--|
| n_ag_g | Number of agents | By default 4,9,16,25,36 and 64 supported |
| rows_g | Number of rows | By default 2,3,4,5,6 and 8 supported |
| cols_g | Number of columns | By default 2,3,4,5,6 and 8 supported |
| stfw_en_g | Enable store-and-forward | 0 = disable, 1 = enable |
| data_width_g | Data bus width | |
| addr_width_g | Address bus width | |
| packet_length_g | Packet length in words | Must be greater or equal to resource's packet_size |
| tx_len_width_g | Number of bits to code tx length | |
| timeout_g | Waiting time for packet completion | |
| fill_packet_g | Enable packet fill with dummy data | 0 = disable, 1 = enable |
| lut_en_g | Enable the use of lut in packet codec | 0 = disable, 1 = enable |
| net_type_g | Network type selection | 0 = mesh |
| len_flit_en_g | Enable length flit | 0 = disable, 1 = enable |
| oaddr_flit_en_g | Enable original address flit | 0 = disable, 1 = enable |
| status_en_g | Enable status | 0 = disable, 1 = enable |
| fifo_depth_g | Router fifo depth in words | |
| mesh_freq_g | Network frequency (Hz) | |
| ip_freq_g | IP frequency (Hz) | |

4 XML TAGS LISTING

4.1 <system>

<system>

Root tag of the model.

| Tags | Count | Notes |
|---------------|-------|-------|
| <application> | 1 | |
| <mapping> | 1 | |
| <platform> | 1 | |
| <constraints> | 1 | |

| Attributes | Type | Description |
|------------|------|-------------|
| none | | |

4.2 <application>

```
<system>  
  <application>
```

| Tags | Count | Notes |
|-------------------|-------|-------|
| <service> | 0+ | |
| <task_graph> | 1+ | |
| <task_connection> | 0+ | |

| Attributes | Type | Description |
|------------|------|-------------|
| none | | |

4.3 <service>

```
<system>
  <application>
    <service>
```

| Tags | Count | Notes |
|--------|-------|-------|
| <task> | 1+ | |

| Attributes | Type | Description |
|------------|---------|-------------|
| id | natural | Required |
| name | string | Optional |

4.3.1 service's <task>

```

<system>
  <application>
    <service>
      <task>

```

| Tags | Count | Notes |
|------|-------|-------|
| none | | |

| Attributes | Type | Description |
|------------|---------|--|
| id | natural | Required, refers to <task> in <task_graph> |

4.4 <task_graph>

```
<system>
  <application>
    <task_graph>
```

| Tags | Count | Notes |
|-------------------|-------|-------|
| <task> | 1+ | |
| <task_connection> | 1+ | |
| <event_list> | 1+ | |

| Attributes | Type | Description |
|------------|------|-------------|
| none | | |

4.5 <task>

```
<system>
  <application>
    <task_graph>
      <task>
```

| Tags | Count | Notes |
|---------------|-------|-----------------|
| <in_port> | 1+ | |
| <out_port> | 0+ | |
| <trigger> | 1+ | |
| <restriction> | 0+ | Not implemented |

| Attributes | Type | Description |
|------------|---------|--------------------------|
| id | natural | Required, must be unique |
| name | string | Optional |
| class | string | Required |

4.6 <mem_area>

```
<system>
  <application>
    <task_graph>
      <mem_area>
```

| Tags | Count | Notes |
|------------|-------|-------|
| <in_port> | 1+ | |
| <out_port> | 0+ | |

| Attributes | Type | Description |
|------------|----------|--------------------------|
| id | natural | Required, must be unique |
| size | positive | Required |
| name | string | Optional |
| class | string | Required |

4.6.1 `<in_port>`

```

<system>
  <application>
    <task_graph>
      <task>
        <in_port>
        <mem_area>
        <in_port>

```

| Tags | Count | Notes |
|------|-------|-------|
| none | | |

| Attributes | Type | Description |
|------------|---------|---|
| id | natural | Required, unique in group of task's <code><in_port></code> and <code><out_port></code> tags |

4.6.2 `<out_port>`

```

<system>
  <application>
    <task_graph>
      <task>
        <out_port>
        <mem_area>
        <out_port>

```

| Tags | Count | Notes |
|------|-------|-------|
| none | | |

| Attributes | Type | Description |
|------------|---------|---|
| id | natural | Required, unique in group of task's <code><in_port></code> and <code><out_port></code> tags |

4.6.3 `<trigger>`

```

<system>
  <application>
    <task_graph>
      <task>
        <trigger>

```

| Tags | Count | Notes |
|---------------------------------|-------|-------|
| <code><in_port></code> | 1+ | |
| <code><exec_count></code> | 1+ | |

| Attributes | Type | Description |
|------------------------------|--------|--|
| <code>dependence_type</code> | string | Optional. Either “or” or “and”, default “or” |

4.6.4 trigger's <in_port>

```

<system>
  <application>
    <task_graph>
      <task>
        <trigger>
          <in_port>

```

| Tags | Count | Notes |
|------|-------|-------|
| none | | |

| Attributes | Type | Description |
|------------|---------|--|
| id | natural | Required, id must be one of task's own in_port |

4.6.5 <exec_count>

```

<system>
  <application>
    <task_graph>
      <task>
        <trigger>
          <exec_count>

```

| Tags | Count | Notes |
|--------------|-------|-------|
| <op_count> | 1+ | |
| <send> | 0+ | |
| <next_state> | 1 | |

| Attributes | Type | Description |
|------------|---------|-------------|
| min | natural | |
| max | natural | |
| mod_period | natural | |
| mod_phase | natural | |

4.6.6 <op_count>

```

<system>
  <application>
    <task_graph>
      <task>
        <trigger>
          <exec_count>
            <op_count>

```

| Tags | Count | Notes |
|-------------|-------|--|
| <int_ops> | 0..1 | At least one of the ops tags must be present |
| <float_ops> | 0..1 | At least one of the ops tags must be present |
| <mem_ops> | 0..1 | At least one of the ops tags must be present |

| Attributes | Type | Description |
|------------|--------|---|
| prob | double | Optional, probability from zero to one, default one |

4.6.7 <int_ops>, <float_ops>, <mem_ops>, <byte_amount>, <burst_amount>

```

<system>
  <application>
    <task_graph>
      <task>
        <trigger>
          <exec_count>
            <op_count>
              <int_ops>
              <float_ops>
              <mem_ops>
            <send>
              <byte_amount>
              <burst_amount>
          <read>
            <byte_amount>
            <burst_amount>

```

| Tags | Count | Notes |
|----------------|-------|--|
| <polynomial> | 0..1 | Mutually exclusive with <distribution> |
| <distribution> | 0..1 | Mutually exclusive with <polynomial> |

| Attributes | Type | Description |
|------------|------|-------------|
| none | | |

4.6.8 <polynomial>

```

<system>
  <application>
    <task_graph>
      <task>
        <trigger>
          <exec_count>
            <op_count>
              <int_ops>
                <polynomial>
              <float_ops>
                <polynomial>
              <mem_ops>
                <polynomial>
            <send>
              <byte_amount>
                <polynomial>

```

| Tags | Count | Notes |
|---------|-------|-------|
| <param> | 1+ | |

| Attributes | Type | Description |
|------------|------|-------------|
| none | | |

4.6.9 <param>

```

<system>
  <application>
    <task_graph>
      <task>
        <trigger>
          <exec_count>
            <op_count>
              <int_ops>
                <polynomial>
                  <param>
                <float_ops>
                  <polynomial>
                    <param>
                <mem_ops>
                  <polynomial>
                    <param>
              <send>
                <byte_amount>
                  <polynomial>
                    <param>

```

| Tags | Count | Notes |
|------|-------|-------|
| none | | |

| Attributes | Type | Description |
|------------|-----------|-------------|
| exp | natural | Required |
| value | xs:double | Required |

4.6.10 <distribution>

```

<system>
  <application>
    <task_graph>
      <task>
        <trigger>
          <exec_count>
            <op_count>
              <int_ops>
                <distribution>
              <float_ops>
                <distribution>
              <mem_ops>
                <distribution>
            <send>
              <byte_amount>
                <distribution>

```

| Tags | Count | Notes |
|-----------|-------|-----------------------------------|
| <uniform> | 0..1 | Mutually exclusive with <normal> |
| <normal> | 0..1 | Mutually exclusive with <uniform> |

| Attributes | Type | Description |
|------------|------|-------------|
| none | | |

4.6.11 <uniform>

```

<system>
  <application>
    <task_graph>
      <task>
        <trigger>
          <exec_count>
            <op_count>
              <int_ops>
                <distribution>
                  <uniform>
              <float_ops>
                <distribution>
                  <uniform>
            <mem_ops>
              <distribution>
                <uniform>
          <send>
            <byte_amount>
              <distribution>
                <uniform>

```

| Tags | Count | Notes |
|------|-------|-------|
| none | | |

| Attributes | Type | Description |
|------------|----------|-------------|
| min | positive | Required |
| max | positive | Required |

4.6.12 <normal>

```

<system>
  <application>
    <task_graph>
      <task>
        <trigger>
          <exec_count>
            <op_count>
              <int_ops>
                <distribution>
                  <normal>
              <float_ops>
                <distribution>
                  <normal>
            <mem_ops>
              <distribution>
                <normal>
          <send>
            <byte_amount>
              <distribution>
                <normal>

```

| Tags | Count | Notes |
|------|-------|-------|
| none | | |

| Attributes | Type | Description |
|--------------------|-----------|---|
| mean | positive | Optional, if not specified the input amount is used as mean |
| standard_deviation | xs:double | Required, value must be greater than zero |

4.6.13 <send>

```

<system>
  <application>
    <task_graph>
      <task>
        <trigger>
          <exec_count>
            <send>

```

| Tags | Count | Notes |
|----------------|-------|-------|
| <byte_amount> | 1 | |
| <burst_amount> | 1 | |

| Attributes | Type | Description |
|------------|-----------|--|
| out_id | natural | Required, must be one of task's own output ports |
| prob | xs:double | Optional, probability for sending, value from zero to one, default one |

4.6.14 <read>

```

<system>
  <application>
    <task_graph>
      <task>
        <trigger>
          <exec_count>
            <read>

```

| Tags | Count | Notes |
|----------------|-------|-------|
| <byte_amount> | 1 | |
| <burst_amount> | 1 | |

| Attributes | Type | Description |
|------------|-----------|--|
| out_id | natural | Required, must be one of task's own output port |
| resp_id | natural | Required, must be one of target memory's output port |
| prob | xs:double | Optional, probability for sending, value from zero to one, default one |

4.6.15 <next_state>

```
<system>
  <application>
    <task_graph>
      <task>
        <trigger>
          <next_state>
```

| Tags | Count | Notes |
|------|-------|-------|
| none | 0 | |

| Attributes | Type | Description |
|------------|--------|--|
| value | string | Required, possible values “FREE” and “READY” |

4.7 <port_connection>

```
<system>
  <application>
    <port_connection>
  <task_graph>
    <port_connection>
```

| Tags | Count | Notes |
|------|-------|-------|
| none | 0 | |

| Attributes | Type | Description |
|------------|---------|---------------------------------|
| src | natural | Required, source port's id |
| dst | natural | Required, destination port's id |

4.8 <event_list>

```
<system>
  <application>
    <task_graph>
      <event_list>
```

| Tags | Count | Notes |
|---------|-------|-------|
| <event> | 1+ | |

| Attributes | Type | Description |
|------------|------|-------------|
| none | | |

4.8.1 <event>

```

<system>
  <application>
    <task_graph>
      <event_list>
        <event>

```

| Tags | Count | Notes |
|------|-------|-------|
| none | | |

| Attributes | Type | Description |
|-------------|-----------|---|
| id | natural | Required, must be unique in group of events |
| period | xs:double | Required (Optional if count is 1), time between sends |
| prob | xs:double | Required, probability for sending |
| out_port_id | natural | Required |
| amount | positive | Required, bytes to send |
| offset | double | Optional, time for first send, default zero |
| count | positive | Optional, how many times to send, default unlimited |
| name | string | Optional |

4.9 <mapping>

```
<system>  
  <mapping>
```

| Tags | Count | Notes |
|------------|-------|-------|
| <resource> | 1+ | |

| Attributes | Type | Description |
|------------|------|-------------|
| none | | |

4.9.1 <resource>

```
<system>
  <mapping>
    <resource>
```

| Tags | Count | Notes |
|---------------|-------|--|
| <group> | 1+ | Mutually exclusive with <sw_platform> |
| <sw_platform> | 1+ | Mutually exclusive with <group> |
| <mem_area> | 1+ | Mutually exclusive with the above, only for memories |

| Attributes | Type | Description |
|------------|---------|---|
| id | natural | Required |
| contents | string | Required, possible values “immutable” and “mutable” |
| name | string | Optional |

4.9.2 `<sw_platform>`

```
<system>
  <mapping>
    <resource>
      <sw_platform>
```

| Tags | Count | Notes |
|---------|-------|-------|
| <group> | 1+ | |

| Attributes | Type | Description |
|------------|---------|---|
| id | natural | Required |
| position | string | Required, possible values “immovable” and “movable” |
| contents | string | Required, possible values “immutable” and “mutable” |
| priority | natural | Optional |

4.9.3 <group>

```

<system>
  <mapping>
    <resource>
      <group>
      <sw_platform>
      <group>

```

| Tags | Count | Notes |
|--------|-------|-------|
| <task> | 1+ | |

| Attributes | Type | Description |
|------------|---------|---|
| id | natural | Required, refers to <task> in <task_graph> |
| position | string | Required, possible values “movable” and “immovable” |
| contents | string | Required, possible values “mutable” and “immutable” |
| name | string | Optional |

4.9.4 group's <task>

```

<system>
  <mapping>
    <resource>
      <group>
        <task>
      <sw_platform>
        <group>
          <task>

```

| Tags | Count | Notes |
|------|-------|-------|
| none | | |

| Attributes | Type | Description |
|------------|---------|---|
| id | natural | Required, refers to <task> in <task_graph> |
| position | string | Required, possible values “movable” and “immovable” |
| name | string | Optional |
| priority | natural | Optional |

4.9.5 resource's <mem_area>

```

<system>
  <mapping>
    <resource>
      <mem_area>

```

| Tags | Count | Notes |
|------|-------|-------|
| none | | |

| Attributes | Type | Description |
|------------|---------|---|
| id | natural | Required, refers to <mem_area> in <task_graph> |
| position | string | Required, possible values “movable” and “immovable” |
| name | string | Optional |
| priority | natural | Optional |

4.10 <platform>

```
<system>  
  <platform>
```

| Tags | Count | Notes |
|-----------------|-------|-------|
| <resource_list> | 1 | |
| <noc> | 1 | |

| Attributes | Type | Description |
|------------|------|-------------|
| none | | |

4.10.1 <resource_list>

```
<system>
  <platform>
    <resource_list>
```

| Tags | Count | Notes |
|------------|-------|-------|
| <resource> | 1+ | |

| Attributes | Type | Description |
|------------|------|-------------|
| none | | |

4.10.2 <resource>

```

<system>
  <platform>
    <resource_list>
      <resource>

```

| Tags | Count | Notes |
|-------------|-------|-------|
| <port> | 1 | |
| <parameter> | 0+ | |
| <cache> | 0-1 | |

| Attributes | Type | Description |
|----------------|----------|---|
| id | natural | Required |
| name | string | Required |
| type | string | Required, type refers to a resource description in HW library |
| frequency | natural | Optional |
| packet_size | positive | Optional |
| rx_buffer_size | positive | Optional |
| tx_buffer_size | positive | Optional |
| request_size | positive | Required for memories |
| ocp_param | srting | Required for memories |
| adm_param | srting | Required for memories |

4.10.3 resource's <port>

```
<system>
  <platform>
    <resource_list>
      <resource>
        <port>
```

| Tags | Count | Notes |
|------|-------|-------|
| none | | |

| Attributes | Type | Description |
|------------|---------|--|
| terminal | natural | Required, refers to a <connection>'s id in <terminal_list> |

4.10.4 <cache>

```
<system>
  <platform>
    <resource_list>
      <resource>
        <cache>
```

| Tags | Count | Notes |
|---------|-------|-------|
| i.cache | 1 | |
| d.cache | 1 | |

| Attributes | Type | Description |
|------------|------|-------------|
| none | | |

4.10.5 <i.cache> and <d.cache>

```

<system>
  <platform>
    <resource_list>
      <resource>
        <cache>
          <i.cache>
          <d.cache>

```

| Tags | Count | Notes |
|--------------|-------|--------------------|
| distribution | 0+ | Mutually exclusive |
| polynomial | 0+ | Mutually exclusive |

| Attributes | Type | Description |
|-------------|----------|---------------------------------------|
| line_size | positive | How many bytes to read per cache miss |
| mem_area_id | natural | Refers to memory resource |

4.10.6 `<parameter>`

```

<system>
  <platform>
    <resource_list>
      <resource>
        <parameter>
        <port>
          <parameter>
      </resource>
    </resource_list>
    <noc>
      <parameter>
      <router_list>
        <router>
          <port>
            <parameter>
        </router>
      </router_list>
    </noc>
  </platform>
</system>

```

`<parameter>` tags are used to pass technology dependent parameters.

| Tags | Count | Notes |
|------|-------|-------|
| none | | |

| Attributes | Type | Description |
|------------|--------|-------------|
| name | string | Required |
| value | string | Required |

4.10.7 <noc>

```
<system>
  <platform>
    <noc>
```

| Tags | Count | Notes |
|-----------------|-------|-------|
| <router_list> | 0+ | |
| <link_list> | 0+ | |
| <terminal_list> | 1 | |
| <parameter> | 0+ | |

| Attributes | Type | Description |
|------------|--------|-------------|
| type | string | Required |

4.10.8 <router_list>

```
<system>
  <platform>
    <noc>
      <router_list>
```

| Tags | Count | Notes |
|----------|-------|-------|
| <router> | 0+ | |

| Attributes | Type | Description |
|------------|------|-------------|
| none | | |

4.10.9 `<router>`

```
<system>
  <platform>
    <noc>
      <router_list>
        <router>
```

| Tags | Count | Notes |
|--------|-------|-------|
| <port> | 1+ | |

| Attributes | Type | Description |
|------------|----------|-------------|
| id | natural | Required |
| type | string | Optional |
| frequency | natural | Required |
| width | positive | Required |
| name | string | Optional |

4.10.10 router's <port>

```

<system>
  <platform>
    <noc>
      <router_list>
        <router>
          <port>

```

| Tags | Count | Notes |
|-------------|-------|-------|
| <parameter> | 0+ | |

| Attributes | Type | Description |
|------------|----------|---|
| id | natural | Required |
| type | string | Optional |
| name | string | Optional |
| width | positive | Optional |
| address | string | Required, TG uses this when sending packets |

4.10.11 <link_list>

```
<system>
  <platform>
    <noc>
      <link_list>
```

| Tags | Count | Notes |
|--------|-------|-------|
| <link> | 0+ | |

| Attributes | Type | Description |
|---------------|----------|-------------|
| default_width | positive | Optional |

4.10.12 <link>

```

<system>
  <platform>
    <noc>
      <link_list>
        <link>

```

| Tags | Count | Notes |
|------|-------|-------|
| none | | |

| Attributes | Type | Description |
|------------|------------------|-------------|
| id | natural | Required |
| src_router | natural | Required |
| dst_router | natural | Required |
| src_port | natural | Required |
| dst_port | natural | Required |
| name | string | Optional |
| width | xs:potiveInteger | Optional |

4.10.13 <terminal_list>

```
<system>
  <platform>
    <noc>
      <terminal_list>
```

| Tags | Count | Notes |
|---------------------|-------|-------|
| <connection> | 1+ | |
| <network_interface> | 1 | |

| Attributes | Type | Description |
|------------|------|-------------|
| none | | |

4.10.14 <connection>

```
<system>
  <platform>
    <noc>
      <terminal_list>
        <connection>
```

| Tags | Count | Notes |
|------|-------|-------|
| none | | |

| Attributes | Type | Description |
|------------|---------|---------------------------------|
| id | natural | Required |
| router | natural | Required, refers to router's id |
| port | natural | Required, refers to port's id |
| name | string | Optional |
| address | string | Optional |

4.10.15 <network_interface>

```
<system>
  <platform>
    <noc>
      <terminal_list>
        <network_interface>
```

| Tags | Count | Notes |
|------|-------|-------|
| none | | |

| Attributes | Type | Description |
|------------|--------|-------------|
| type | string | Required |
| name | string | Optional |

4.11 <constraints>

```
<system>
  <constraints>
```

| Tags | Count | Notes |
|--------------------|-------|--|
| <rng_seed> | 0..1 | If not defined current system time is used as a seed |
| <sim_resolution> | 1 | |
| <sim_length> | 1 | |
| <measurements> | 1 | |
| <path_measurement> | 0+ | |
| <exec_mon> | 0..1 | |
| <pe_lib> | 1 | |
| <log_packet> | 0..1 | |
| <log_token> | 0..1 | |
| <log_summary> | 0..1 | |
| <log_pe> | 0..1 | |
| <log_app> | 0..1 | |
| <cost_function> | 0+ | |

| Attributes | Type | Description |
|------------|------|-------------|
| none | | |

4.11.1 <rng_seed>

```
<system>
  <constraints>
    <rng_seed>
```

| Tags | Count | Notes |
|------|-------|-------|
| none | | |

| Attributes | Type | Description |
|------------|---------|---|
| value | natural | Required, Used to seed random number generators |

4.11.2 <sim_resolution>

```
<system>
  <constraints>
    <sim_resolution>
```

| Tags | Count | Notes |
|------|-------|-------|
| none | | |

| Attributes | Type | Description |
|------------|--------|--|
| time | double | Required, simulations resolution |
| unit | string | Required, possible values are fs, ps, ns, us, ms and s |

4.11.3 <sim_length>

```
<system>
  <constraints>
    <sim_length>
```

| Tags | Count | Notes |
|------|-------|-------|
| none | | |

| Attributes | Type | Description |
|------------|--------|--|
| time | double | Required, simulations length |
| unit | string | Required, possible values are fs, ps, ns, us, ms and s |

4.11.4 <measurements>

```
<system>
  <constraints>
    <measurements>
```

| Tags | Count | Notes |
|------|-------|-------|
| none | | |

| Attributes | Type | Description |
|------------|--------|--|
| time | double | Required, time between measurements |
| unit | string | Required, possible values are fs, ps, ns, us, ms and s |

4.11.5 `<path_measurement>`

```
<system>
  <constraints>
    <path_measurement>
```

| Tags | Count | Notes |
|------|-------|-------|
| none | | |

| Attributes | Type | Description |
|------------|--------------|-------------------------------------|
| src | unsigned int | Required, starting source port id |
| dst | unsigned int | Required, final destination port id |

For every token sent through the starting source port there must be exactly one incoming token to final destination port.

4.11.6 <exec_mon>

```
<system>
  <constraints>
    <exec_mon>
```

| Tags | Count | Notes |
|------|-------|-------|
| none | | |

| Attributes | Type | Description |
|------------|--------|--|
| using | string | Required, true if using Execution Monitor, false otherwise |

4.11.7 <pe_lib>

```
<system>  
  <constraints>  
    <pe_lib>
```

| Tags | Count | Notes |
|------|-------|-------|
| none | | |

| Attributes | Type | Description |
|------------|--------|--|
| file | string | Required, path to processing element library |

4.11.8 `<log_*>`

```
<system>
  <constraints>
    <log_packet>
    <log_token>
    <log_summary>
    <log_pe>
    <log_app>
```

| Tags | Count | Notes |
|------|-------|-------|
| none | | |

| Attributes | Type | Description |
|------------|--------|------------------------------|
| file | string | Required, file to store logs |

4.11.9 <cost_function>

```
<system>
  <constraints>
    <cost_function>
```

| Tags | Count | Notes |
|------|-------|-------|
| none | | |

| Attributes | Type | Description |
|------------|--------|---|
| func | string | Required, function to be evaluated, parser understands floating point numbers, four basic operators (+ − */), parenthesis and variables listed in the table below |

| Variable | Description |
|--|--|
| <i>pu</i> .[<i>n</i>] | Average PE utilization for PE with id <i>n</i> , value from zero to one |
| <i>pu</i> .[<i>name</i>] | Average PE utilization for PE with name <i>name</i> , value from zero to one |
| <i>pu_avg</i> | Average PE utilization for all PEs, value from zero to one |
| <i>pf</i> .[<i>id</i>] | Frequency (MHz) of PE with id <i>id</i> |
| <i>tc</i> .[<i>id</i>] | Total times task with id <i>id</i> has been triggered |
| <i>tc</i> .[<i>name</i>] | Total times task with name <i>name</i> has been triggered |
| <i>tc_tot</i> | Total times all tasks have been triggered |
| <i>tt</i> .[<i>id</i>].[<i>n</i>] | Time when task <i>id</i> has been triggered <i>n</i> times, in seconds |
| <i>ec</i> .[<i>n</i>] | Total times event with id <i>n</i> has happened |
| <i>ec_tot</i> | Total times events have happened |
| <i>lat</i> .[<i>src</i>].[<i>dst</i>]. <i>min</i> | Minimum latency for token from out_port <i>src</i> to in_port <i>dst</i> |
| <i>lat</i> .[<i>src</i>].[<i>dst</i>]. <i>max</i> | Maximum latency for token from out_port <i>src</i> to in_port <i>dst</i> |
| <i>lat</i> .[<i>src</i>].[<i>dst</i>]. <i>avg</i> | Average latency for token from out_port <i>src</i> to in_port <i>dst</i> |
| <i>latf</i> .[<i>src</i>].[<i>dst</i>]. <i>min</i> | Minimum latency for token from out_port <i>src</i> to in_port <i>dst</i> , only fully sent counted |
| <i>latf</i> .[<i>src</i>].[<i>dst</i>]. <i>max</i> | Maximum latency for token from out_port <i>src</i> to in_port <i>dst</i> , only fully sent counted |
| <i>latf</i> .[<i>src</i>].[<i>dst</i>]. <i>avg</i> | Average latency for token from out_port <i>src</i> to in_port <i>dst</i> , only fully sent counted |
| <i>path</i> .[<i>src</i>].[<i>dst</i>]. <i>min</i> | Minimum latency for a token to travel through path from out_port <i>src</i> to in_port <i>dst</i> |
| <i>path</i> .[<i>src</i>].[<i>dst</i>]. <i>max</i> | Maximum latency for a token to travel through path from out_port <i>src</i> to in_port <i>dst</i> |
| <i>path</i> .[<i>src</i>].[<i>dst</i>]. <i>avg</i> | Average latency for a token to travel through path from out_port <i>src</i> to in_port <i>dst</i> |
| <i>path</i> .[<i>src</i>].[<i>dst</i>]. <i>count</i> | How many times path from out_port <i>src</i> to in_port <i>dst</i> has been completed |

Tokens that were not fully sent through the network get a latency from their creation to the end of the simulation for *lat*_ variables. *latf*_ variables don't include tokens that were not completely sent. Latencies are measured in seconds.