MappingTable Functional Description

This object is NOT an hardware component. It can be used by the system designer to describe the memory mapping and address decoding scheme of any hardware architecture build with the SoCLib hardware components.

This object is basically an associative table, where each entry define a segment descriptor.

The mapping table is a centralized description of both the address space segmentation, and the mapping of the segments on the VCI physical targets.

From this centralized description, it is possible to derive the **routing tables** used by the hardware interconnect to decode the VCI address, and route the VCI packets to the proper target.

All VCI initiators and VCI targets share the same address space, but the address decoding scheme can support a structured interconnect.

Usage

Mapping Table

Mapping table must know:

- The address width
- The number interconnection levels (most of the time 1 or 2)
- The widths of the address subfields used for command routing
- The widths of the RSRCID subfields used for response routing
- The mask used in XCache for determining if an address is cacheable or not

The number of interconnection levels is implicit by the length of lists of subfields lengths.

Mapping Table instanciation is:

```
MappingTable maptab( addr_width, addr_bits, srcid_bits, cacheability_mask);
```

For instance:

```
MappingTable maptab( 32, IntTab(8, 4), IntTab(8, 2), 0x000c0000);
```

This creates a 32-bit Mapping Table, with 2 level interconnection, 8 address bits for global command routing, 4 address bits for local command routing, 8 RSRCID bits for global response routing, 2 RSRCID bits for local response routing. This makes RSRCID field 10-bit wide, this should be enforced.

Segments

Segments holds information about a portion of addressable memory region, with some attributes:

- a name
- a base address
- a size

- a target_index (an IntTab)
- a cacheability flag

The size field is a number of bytes. The target_index field identifies the VCI target that contains the corresponding segment, and is used by the interconnect to route a command packet to the proper target. The cacheability field can be used by the cache controllers to define if the corresponding segment is cacheable.

All segments defined by the system designer for a given architecture must be non-overlaping.

A Segment instanciation is:

```
Segment( name, base_address, size, target, cacheable )
```

For instance:

```
Segment( "seg0", 0x50000, 0x1000, IntTab(3,2), true )
```

This segment is associated to target on port 2 in cluster no 3. It is cacheable.

It can be added in Mapping Table with the method add():

```
maptab.add(Segment( "seg0", 0x50000, 0x1000, IntTab(3,2), true ))
```

For each new segment added, the mapping table will ensure the segments are not overlapping.

Cacheability

Cacheability is a by-segment attribute. It is associated to the address. Here, "seg0" is at address 0x50000, and cacheability mask is 0xc0000, resulting cacheability-determining-address is `0x50000 & 0xc0000 = 0x40000`. Now, for any other address matching `address & 0xc0000 = 0x40000`, cacheability will be true. This will be enforced by mapping_table or you'll get an `"Incoherent MappingTable" exception`.

Interconnection examples

One level interconnect

This is the simplest case, where all VCI targets and VCI initiators are connected to a "flat" interconnect.

- each VCI component is identified by a simple index.
- all VCI targets must have different indexes.
- all VCI initiators must have different indexes.
- The initiator index must be equal to the VCI SRCID value. Most hardware interconnects (such as the PibusBcu or the VciVgmn components) make the assumption that the initiator indexes are between 0 and M -1 , where M is the total number of VCI initiators.
- The VCI ADDRESS field is structured in two fields: `| MSB | OFFSET |`
    - The MSB field is decoded by the flat interconnect to route the command packet to the proper VCI target.
    - The OFFSET field is decoded by the VCI target.
- The VCI SRCID field is used by the interconnect to route the response packet to the proper VCI initiator.
- Most flat hardware interconnects (such as the PibusBcu or the VciVgmn components) make the assumption that the target indexes are between 0 and T -1 (where T is the total number of VCI targets), and the initiator indexes are between 0 and M -1 (where M is the total number of VCI initiators).

The flat interconnect must contain a ROM implementing a **routing table** indexed by the VCI address MSBs and containing the corresponding target index. The content of this **routing table** is automatically computed by a method associated to the mapping table.

Two-level interconnect

With a two-level interconnect, the hardware architecture is supposed to be split into several subsystems (or clusters), with a global interconnect for inter-cluster communications, and one local interconnect in each cluster for intra-cluster communications.

- each VCI component is identified by a structured index containing two indexes:
  - a global index that identifies the subsystem (or cluster) index.
  - a local index, that identifies the VCI component in the cluster.
- all VCI components in the same cluster must have the same global index.
- all VCI components in the same cluster must have different local indexes.
- The VCI ADDRESS field is structured in three fields : `| MSB | LSB | OFFSET |`
  - The MSB field is decoded by the global interconnect to route the command packet to the proper cluster.
  - The LSB field is decoded by the local interconnect to route the command packet to the proper target.
  - The OFFSET field is decoded by the VCI target.
- The VCI SRCID field is structured in two fields : `| MSB | LSB |`
  - The SRCID MSB field must be equal to the initiator global index.
  - The SRCID LSB field must be equal to the initiator local index.

The services provided by the Mapping Table for a two level interconnect are the following :

- It generates the **Global Routing Table**, indexed by the VCI ADDRESS MSB bits, and containing the corresponding global index (cluster index). This Global Routing Table is used by the global interconnect.
- It generates - for each cluster - the **Local Routing Table**, indexed by the VCI ADDRESS LSB bits and containing the corresponding target local index. Depending on the mapping, each cluster can have a different Local Routing Table.

More levels

You may imagine those schemes are extensible to arbitrary level of interconnexion. The current MappingTable, with the help of `IntTabs`, is not limited.