LLHSC User Manual

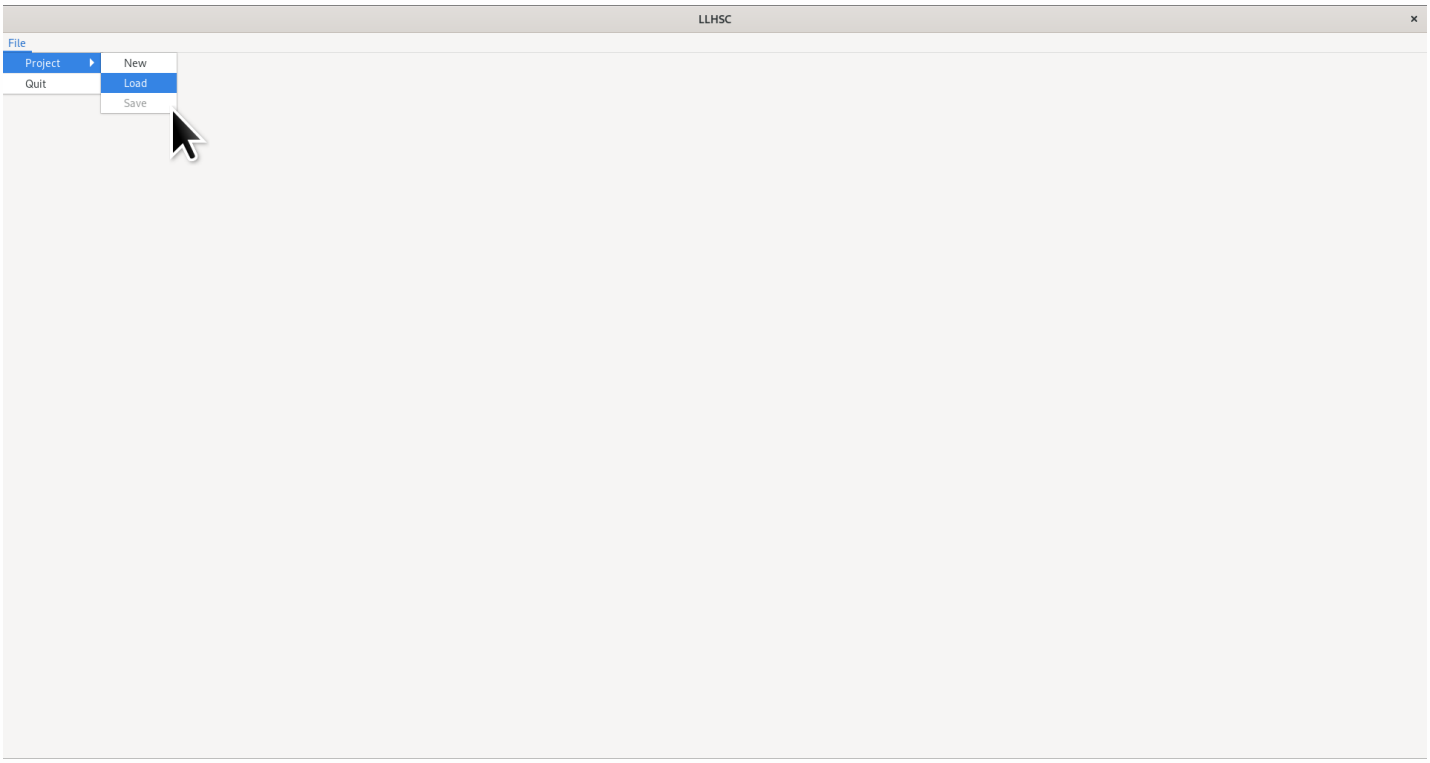
Open an existing project

The default project is located inside the "examples" directory, which contains the following files:

- qemu-aarch64-virt-minimal.dts
- qemu-aarch64-virt-minimal_nodes.xml
- delta1

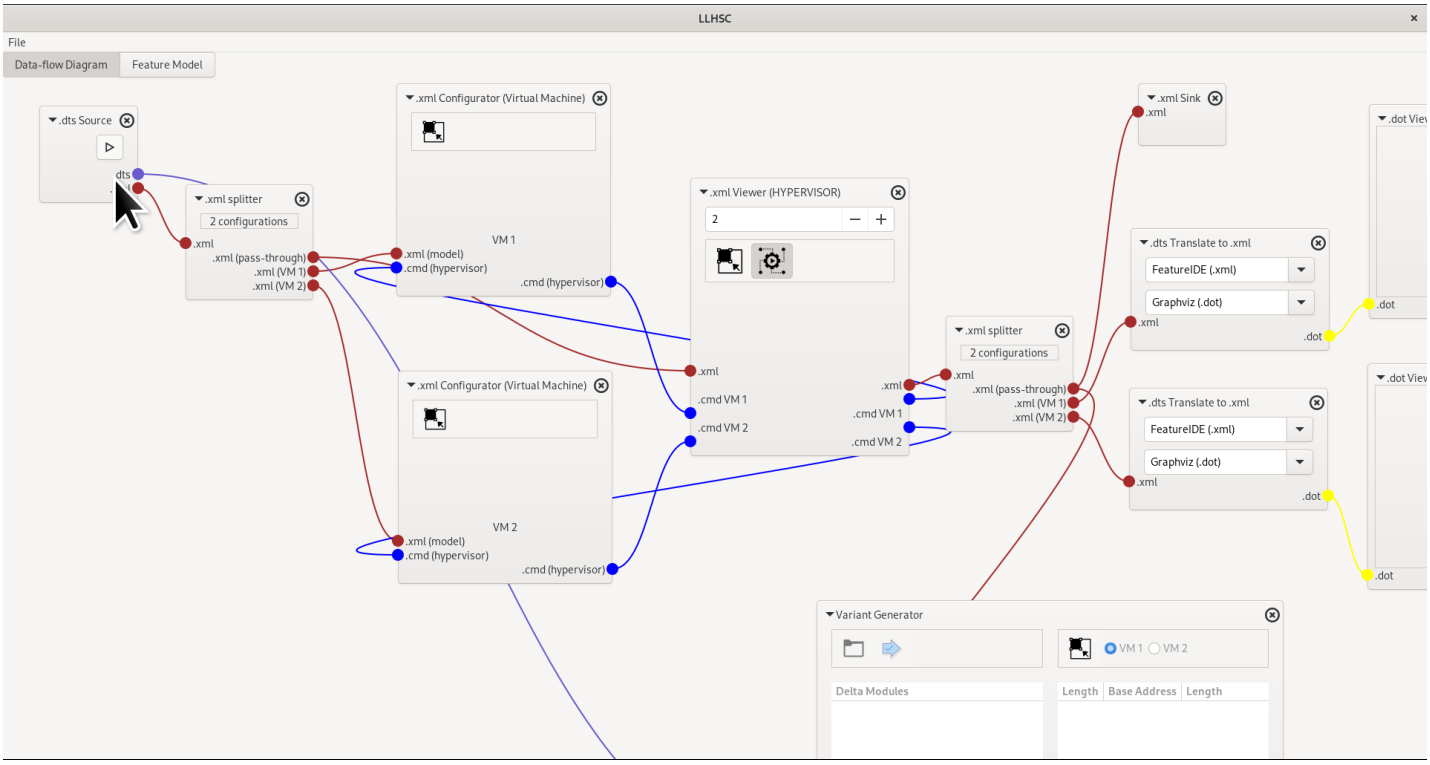
To load this project, click on ***File -> Project -> Load*** and select the file "qemu-aarch64-virt-minimal_nodes.xml".

Two tabs will be opened: ***Data-flow Diagram*** and ***Feature Model***.



To start the process it is necessary to load the core DTS file. This is accomplished by pressing the **play** button in the node ***.dts source***, as indicated in the figure below.

The next step is to edit the Feature Model and add the cross-tree constraints.



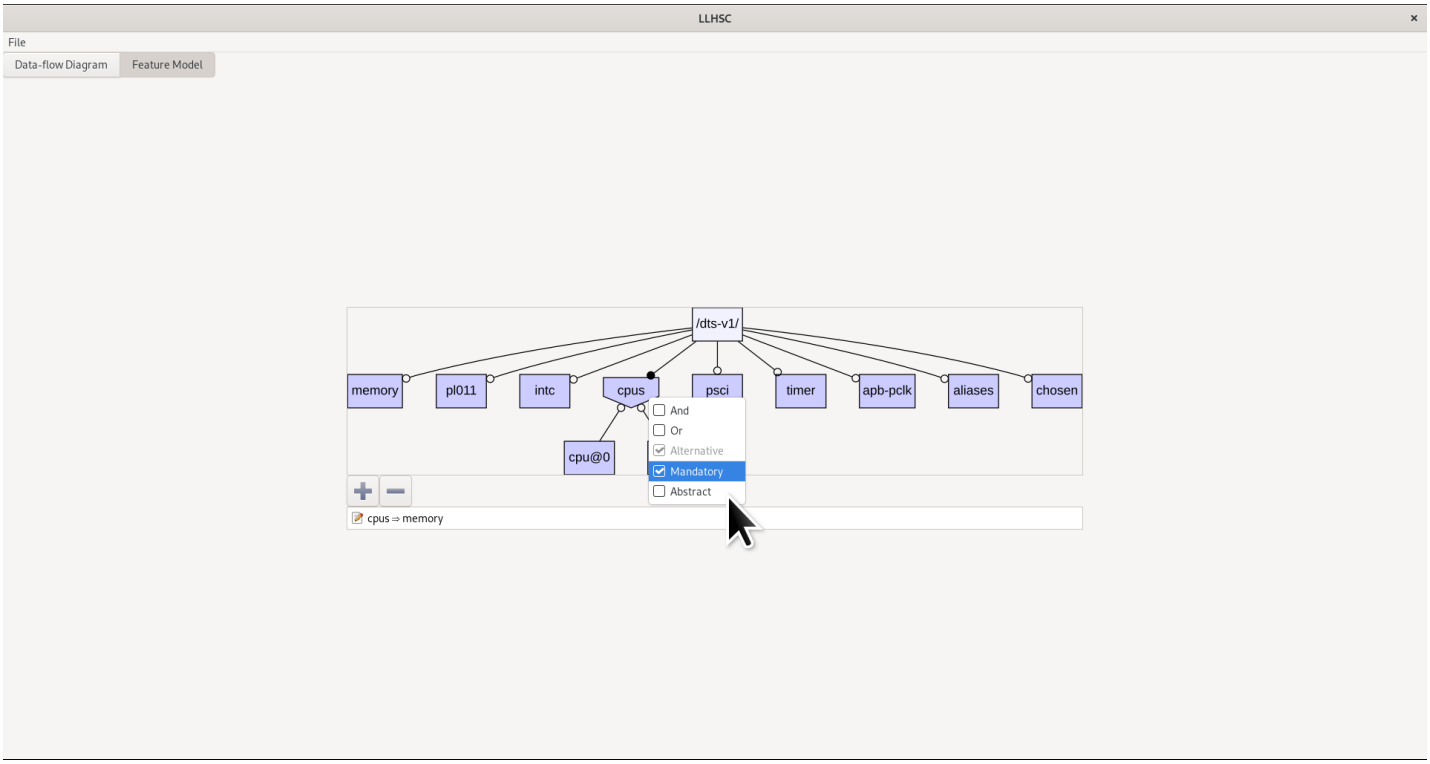
Editing the Feature Model

In the Feature Model editor, it is possible to edit each node found inside the DeviceTree and restrict the possible products generated by the model.

In this project, it is specified that ***/dts-v1/cpus*** is a "mandatory" feature, which indicates that each virtual machine needs to have this feature selected.
The same node is also marked as "alternative", which means that only one of the two CPUs can be selected inside one product.

Additionally, the cross-tree constraint ***cpus >> memory***, specifies that if any of the CPUs is selected, then the memory feature must also be selected.

The next step is to setup the number of virtual machines, and generate a product for each virtual machine.

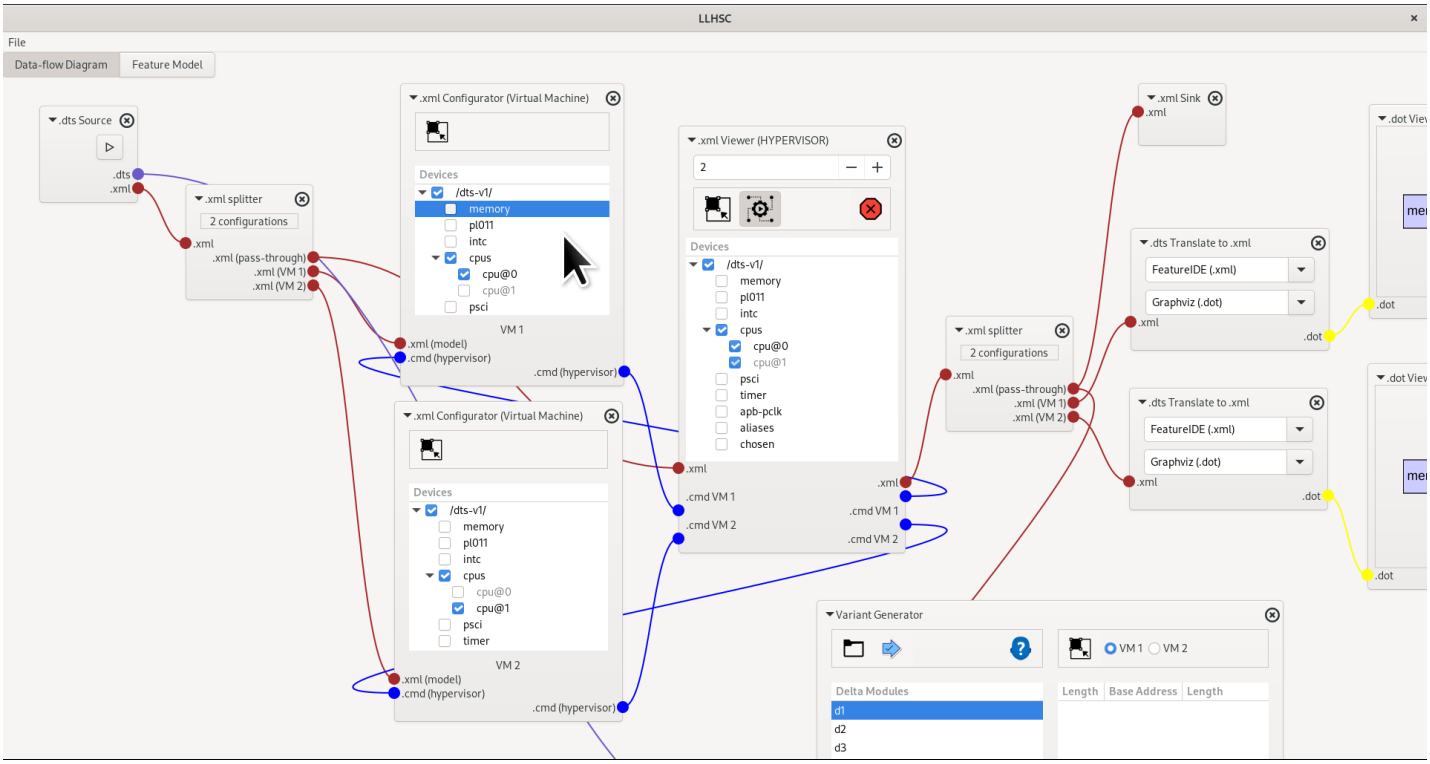


Virtual machines (and Hypervisor) configuration

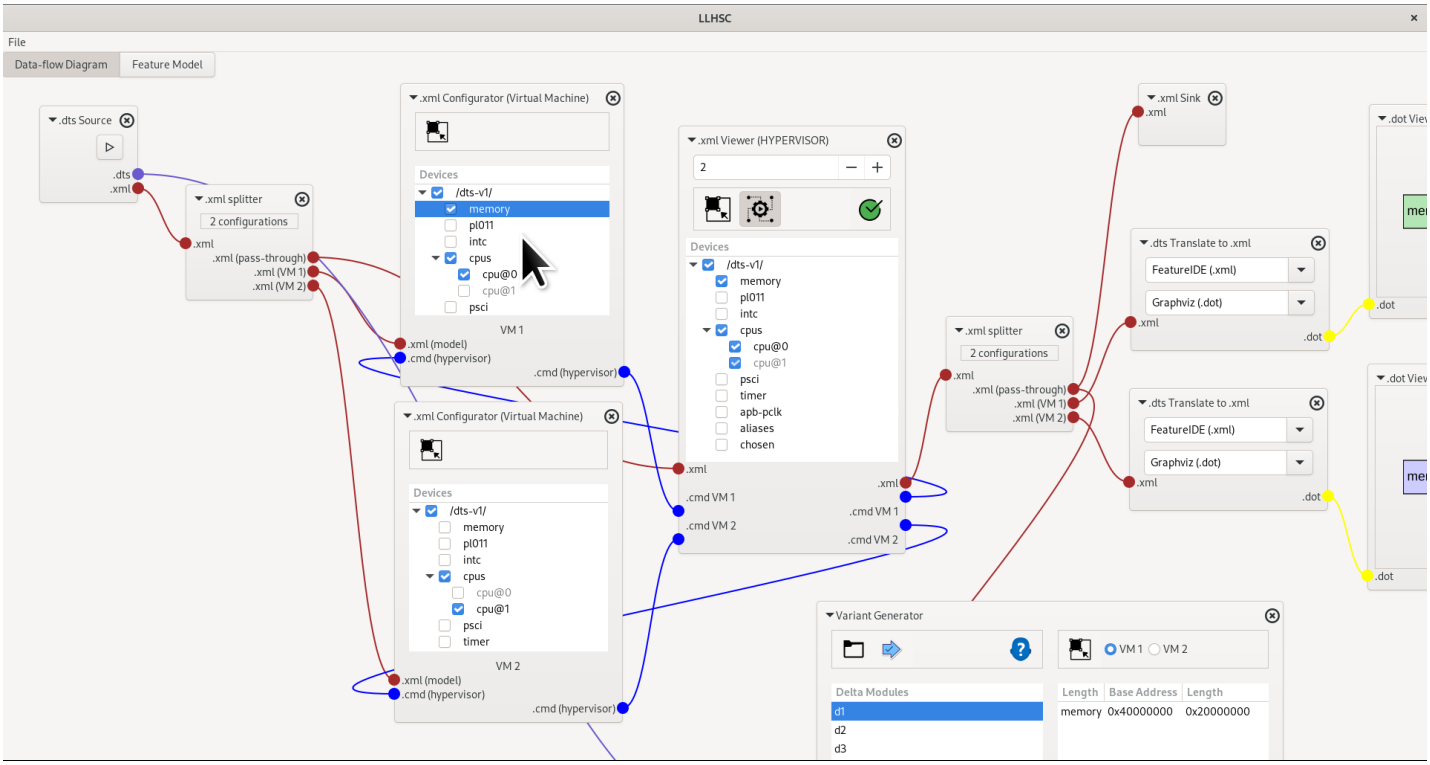
At this point is possible to generate products from virtual machines. In the current configuration, there are two virtual machines. These are shown inside the ***.xml Configurator (Virtual Machine)*** nodes, where each feature is represented by a checkbox.

The configuration of the hypervisor is shown inside the ***.xml viewer (HYPERVISOR)*** node, which is performed the validation of the virtual machine products. In this view, the checkboxes are not editable.

In the figure below, the **/dts-v1/memory** feature is de-selected and the feature **cpu@0** is selected. Because the cross-tree constraint specified earlier is not satisfied, the configuration of the hypervisor is invalid. This is shown by the red icon on the top-right side of the node.

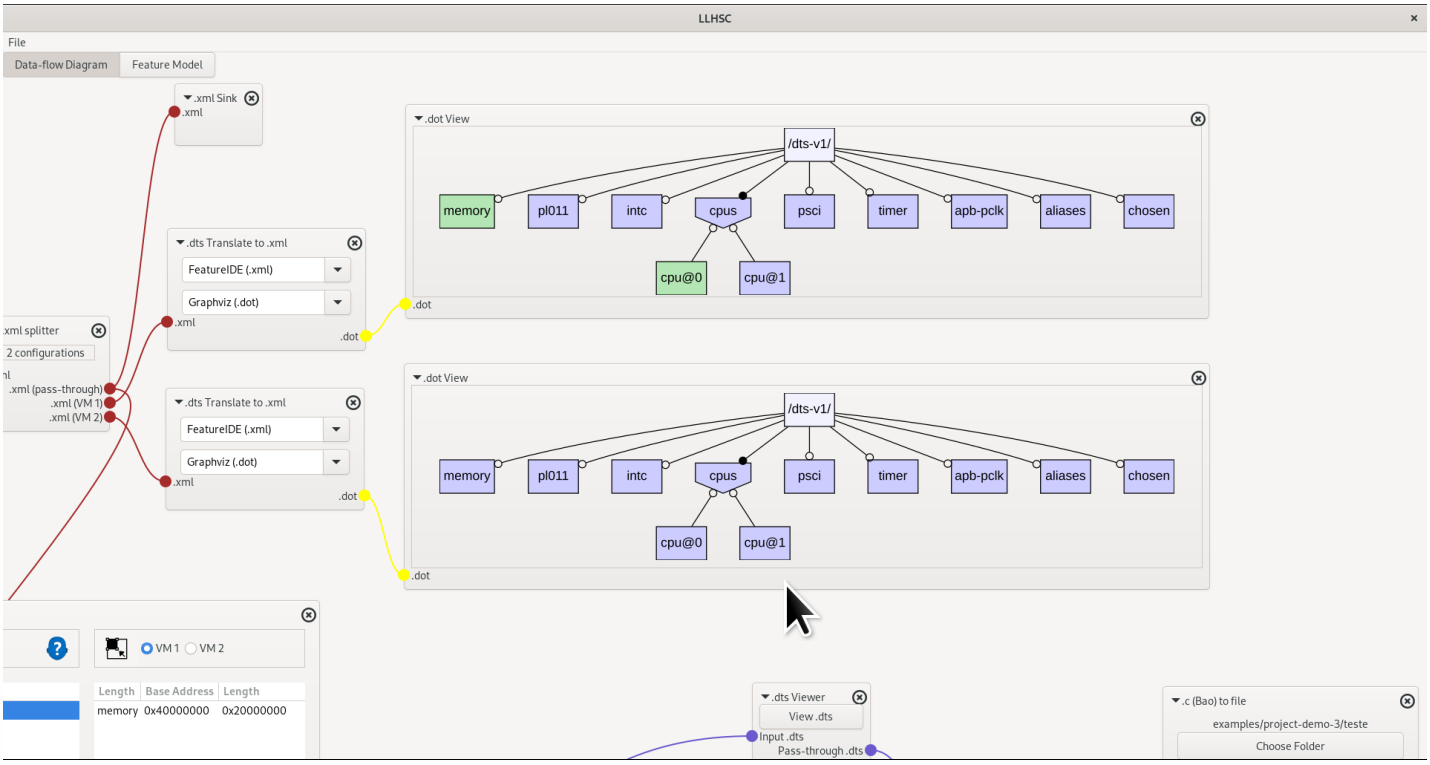


The figure below shows that if one of the virtual machines has the **/dts-v1/memory** feature selected, then the hypervisor configuration turns into valid.



In order to check which features were manually selected, it is possible to visualize the products corresponding to each virtual machine configuration.

In this example, the second product is, in fact, automatically configured because of the constraints imposed in the feature model. Since the **/dts-v1/ cpus** feature is both "mandatory" and "alternative", once one cpu is selected for the first virtual machine, the cpu in the second virtual machine is automatically selected, as determined by the constraint system.



Generating delta-oriented DeviceTree variants

The next step is to generate variants of the core DTS. For this purpose a delta-oriented software product line is used. The set of deltas is specified inside the **delta1** file. The deltas **d1**, **d2**, **d3** and **d4** as listed below.

The delta **d1** adds a new binding to the core DTS under the node "vEthernet", corresponding to a virtual ethernet device using a 32-bit address space. The activation condition for this delta is "cpu@1", which that this delta is applied whenever the feature **/dts-v1/cpus/cpu@1** is selected.

In order to be correctly applied, the binding "vEthernet" must be present in the DTS to which the delta is applied. For this purpose, it is used the delta order specfied by the "when" clause. In this case, **d1** must be applied after **d3**.

```
delta d1 after d3 when(cpu@1) {
    adds binding vEthernet {
        veth1@80000000{
            compatible="veth";
            reg=<0x80000000 0x10000000>;
            id=<1>;
        };
    }
}
```

Similarly to the previous delta, **d2** adds a new device virtual device for ethernet network whenever the feature **/dts-v1/cpus/cpu@0** is selected.

```
delta d2 after d3 when(cpu@0) {
```

```
    adds binding vEthernet {
        veth0@70000000{
            compatible="veth";
            reg=<0x70000000 0x10000000>;
            id=<0>;
        };
    }
}
```

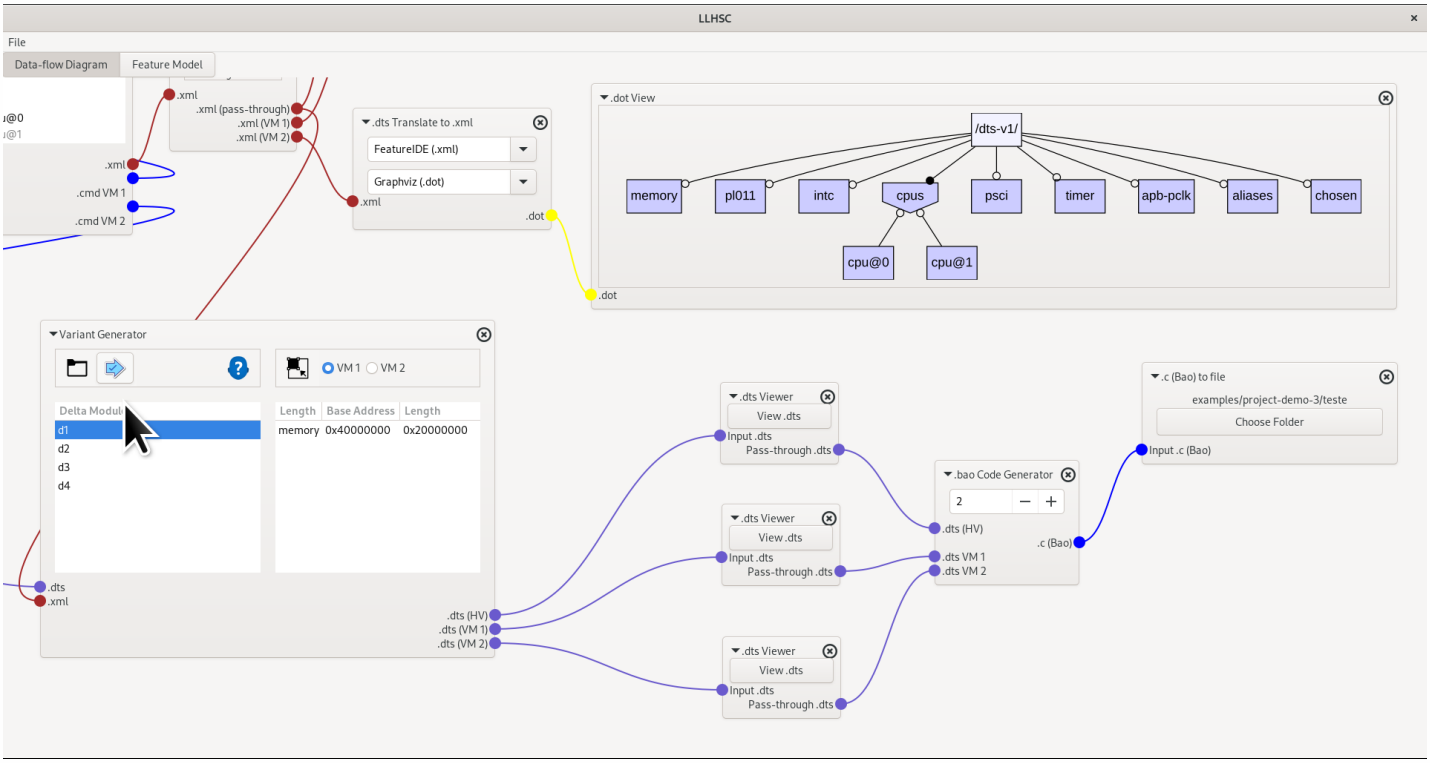
The delta **d3** is applied when the activation condition "(cpu@0 | cpu@1)" is true. In this case, the address space is converted to 32-bit, by setting the "#address-cells" and "#size-cells" to 1.

```
delta d3 when(cpu@0|cpu@1) {
    modifies / {
        #address-cells=<1>;
        #size-cells=<1>;
        vEthernet{ };
    }
}
```

Finally, the delta **d4** specifies the valid address space inside the DTS.

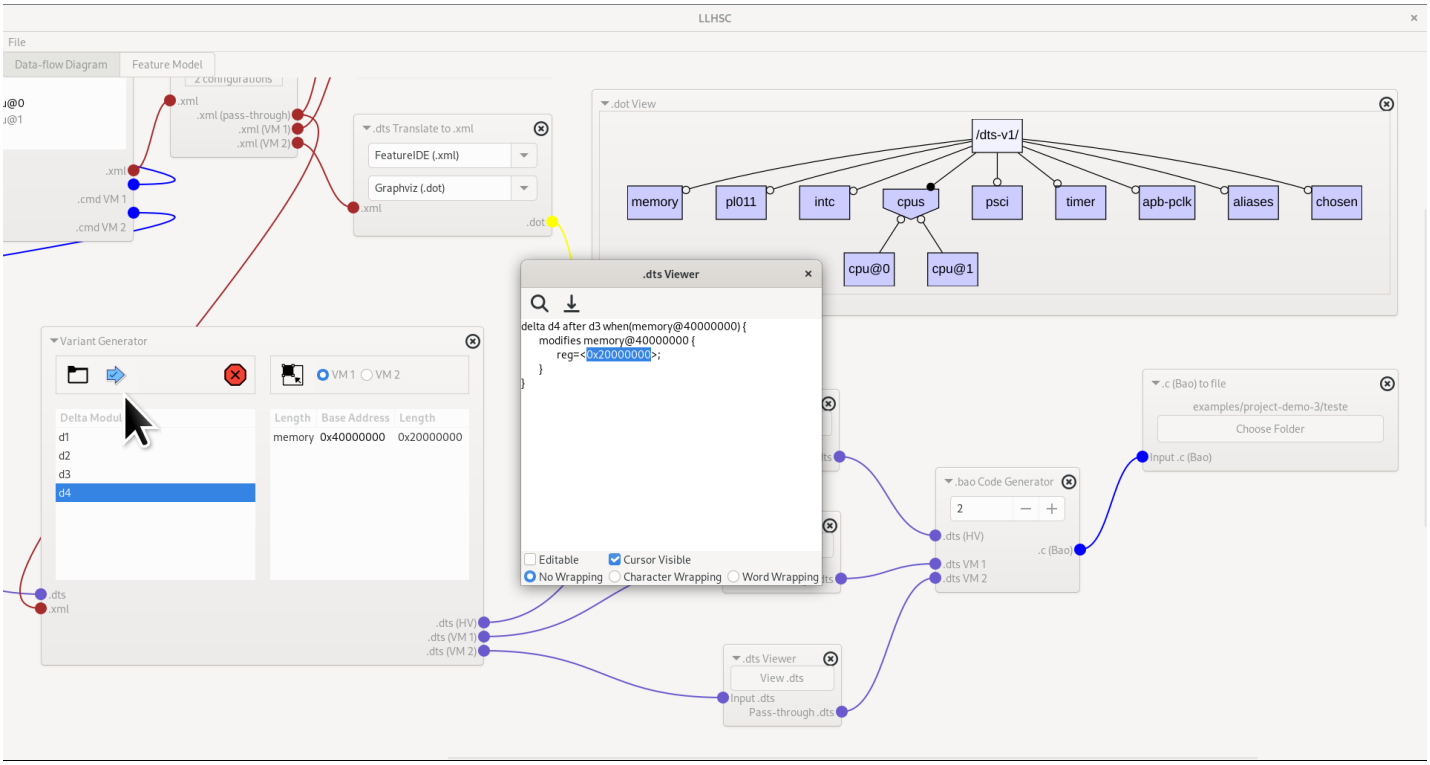
```
delta d4 after d3 when(memory@40000000) {
    modifies memory@40000000 {
        reg=<0x40000000 0x20000000>;
    }
}
```

Delta application is done by pressing the "blue" arrow at the top-left of the ***Delta Generator*** window. Depending of the result provided by the checker, the icon in the top-right is either "green", if no errors were found, or "red", otherwise.



In order to test the checker for errors, the following procedure can be taken. Edit the **d4** delta by double-clicking on the corresponding row. Then edit the text in the floating window, and remove the base address of the memory bank. Then click on the "save" icon.

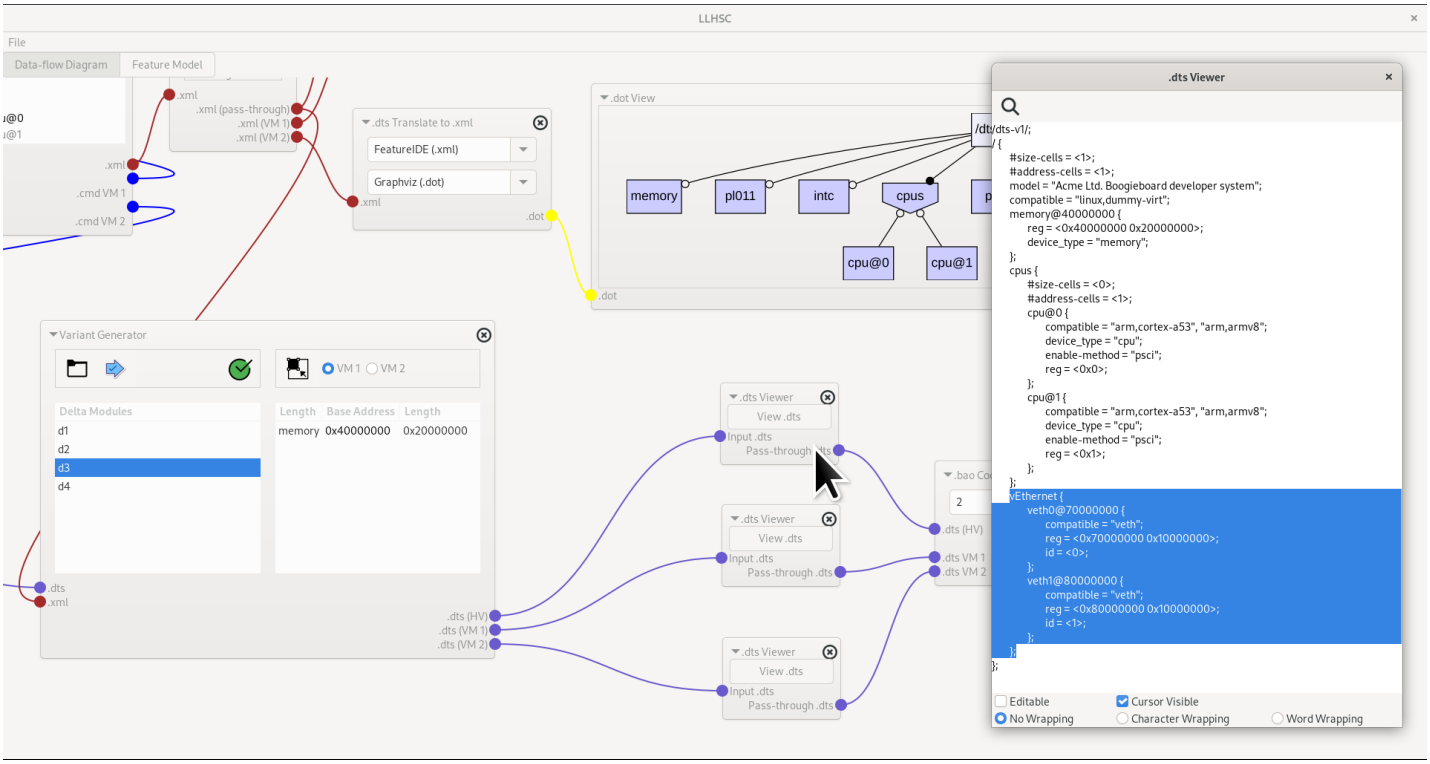
When pressing the "apply" button again, the result of the checker is an error, and the "red" icon is show, as shown in the figure below.



Visualize DeviceTree variants

The last step in the process is to visualize the DTS variants. The windows called ***.dts Viewer*** are used for this purpose. When clicking on the button inside, a floating window appears showing the corresponding variant.

The figure bellow show the DTS of the hypervisor platform, where the virtual devices for ethernet network communication are already present. The Bao configuration file is then automatically generated from this DTS. The remaining DTSs are used by the hypervisor to configure each of the two virtual machines.



April, 10, 2023