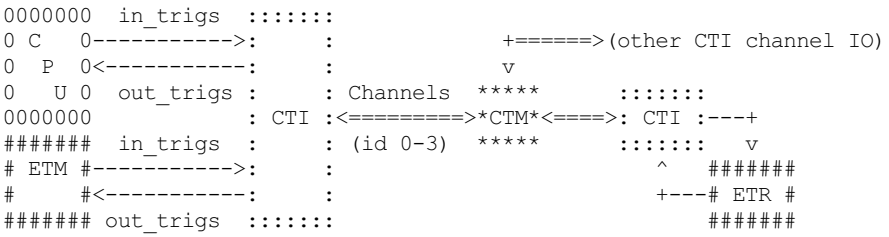
CoreSight Embedded Cross Trigger (CTI & CTM).

Author: Mike Leach <mike.leach@linaro.org>
Date: November 2019

Hardware Description

The CoreSight Cross Trigger Interface (CTI) is a hardware device that takes individual input and output hardware signals known as triggers to and from devices and interconnects them via the Cross Trigger Matrix (CTM) to other devices via numbered channels, in order to propagate events between devices.

e.g.:



The CTI driver enables the programming of the CTI to attach triggers to channels. When an input trigger becomes active, the attached channel will become active. Any output trigger attached to that channel will also become active. The active channel is propagated to other CTIs via the CTM, activating connected output triggers there, unless filtered by the CTI channel gate.

It is also possible to activate a channel using system software directly programming registers in the CTI.

The CTIs are registered by the system to be associated with CPUs and/or other CoreSight devices on the trace data path. When these devices are enabled the attached CTIs will also be enabled. By default/on power up the CTIs have no programmed trigger/channel attachments, so will not affect the system until explicitly programmed.

The hardware trigger connections between CTIs and devices is implementation defined, unless the CPU/ETM combination is a v8 architecture, in which case the connections have an architecturally defined standard layout.

The hardware trigger signals can also be connected to non-CoreSight devices (e.g. UART), or be propagated off chip as hardware IO lines.

All the CTI devices are associated with a CTM. On many systems there will be a single effective CTM (one CTM, or multiple CTMs all interconnected), but it is possible that systems can have nets of CTIs+CTM that are not interconnected by a CTM to each other. On these systems a CTM index is declared to associate CTI devices that are interconnected via a given CTM.

Sysfs files and directories

The CTI devices appear on the existing CoreSight bus alongside the other CoreSight devices:

```
>$ ls /sys/bus/coresight/devices
 cti_cpu0  cti_cpu2  cti_sys0  etm0  etm2  funnel0  replicator0  tmc_etr0
 cti_cpu1  cti_cpu3  cti_sys1  etm1  etm3  funnel1  tmc_etf0     tpiu0
```

The `cti_cpu<N>` named CTIs are associated with a CPU, and any ETM used by that core. The `cti_sys<N>` CTIs are general system infrastructure CTIs that can be associated with other CoreSight devices, or other system hardware capable of generating or using trigger signals.:

```
>$ ls /sys/bus/coresight/devices/etm0/cti_cpu0
channels  ctmid  enable  nr_trigger_cons mgmt  power powered  regs
connections subsystem triggers0 triggers1  uevent
```

Key file items are:-

- `enable`: enables/disables the CTI. Read to determine current state. If this shows as enabled (1), but `powered` shows unpowered (0), then the enable indicates a request to enabled when the device is powered.
- `ctmid` : associated CTM - only relevant if system has multiple CTI+CTM clusters that are not interconnected.
- `nr_trigger_cons` : total connections - triggers<N> directories.
- `powered` : Read to determine if the CTI is currently powered.

Sub-directories:-

- `triggers<N>`: contains list of triggers for an individual connection.
- `channels`: Contains the channel API - CTI main programming interface.
- `regs`: Gives access to the raw programmable CTI regs.
- `mgmt`: the standard CoreSight management registers.
- `connections`: Links to connected *CoreSight* devices. The number of links can be 0 to `nr_trigger_cons`. Actual number given by `nr_links` in this directory.

triggers<N> directories

Individual trigger connection information. This describes trigger signals for CoreSight and non-CoreSight connections.

Each triggers directory has a set of parameters describing the triggers for the connection.

- `name` : name of connection
- `in_signals` : input trigger signal indexes used in this connection.
- `in_types` : functional types for in signals.
- `out_signals` : output trigger signals for this connection.
- `out_types` : functional types for out signals.

e.g:

```
>$ ls ./cti_cpu0/triggers0/
in_signals  in_types  name  out_signals  out_types
>$ cat ./cti_cpu0/triggers0/name
cpu0
>$ cat ./cti_cpu0/triggers0/out_signals
0-2
>$ cat ./cti_cpu0/triggers0/out_types
pe_edbgreq pe_dbgrestart pe_ctiirq
>$ cat ./cti_cpu0/triggers0/in_signals
0-1
>$ cat ./cti_cpu0/triggers0/in_types
pe_dbgtrigger pe_pmuirq
```

If a connection has zero signals in either the 'in' or 'out' triggers then those parameters will be omitted.

Channels API Directory

This provides an easy way to attach triggers to channels, without needing the multiple register operations that are required if manipulating the 'regs' sub-directory elements directly.

A number of files provide this API:

```
>$ ls ./cti_sys0/channels/
chan_clear         chan_inuse      chan_xtrigs_out     trigin_attach
chan_free          chan_pulse      chan_xtrigs_reset   trigin_detach
chan_gate_disable  chan_set        chan_xtrigs_sel     trigout_attach
chan_gate_enable   chan_xtrigs_in  trig_filter_enable  trigout_detach
trigout_filtered
```

Most access to these elements take the form:

```
echo <chan> [<trigger>] > /<device_path>/<operation>
```

where the optional <trigger> is only needed for trigXX_attach | detach operations.

e.g.:

```
>$ echo 0 1 > ./cti_sys0/channels/trigout_attach
>$ echo 0 > ./cti_sys0/channels/chan_set
```

Attaches trigout(1) to channel(0), then activates channel(0) generating a set state on cti_sys0.trigout(1)

API operations

- `trigin_attach, trigout_attach`: Attach a channel to a trigger signal.
- `trigin_detach, trigout_detach`: Detach a channel from a trigger signal.
- `chan_set`: Set the channel - the set state will be propagated around the CTM to other connected devices.
- `chan_clear`: Clear the channel.
- `chan_pulse`: Set the channel for a single CoreSight clock cycle.
- `chan_gate_enable`: Write operation sets the CTI gate to propagate (enable) the channel to other devices. This operation takes a channel number. CTI gate is enabled for all channels by default at power up. Read to list the currently enabled channels on the gate.
- `chan_gate_disable`: Write channel number to disable gate for that channel.
- `chan_inuse`: Show the current channels attached to any signal
- `chan_free`: Show channels with no attached signals.
- `chan_xtrigs_sel`: write a channel number to select a channel to view, read to show the selected channel number.
- `chan_xtrigs_in`: Read to show the input triggers attached to the selected view channel.
- `chan_xtrigs_out`:Read to show the output triggers attached to the selected view channel.
- `trig_filter_enable`: Defaults to enabled, disable to allow potentially dangerous output signals to be set.
- `trigout_filtered`: Trigger out signals that are prevented from being set if filtering `trig_filter_enable` is enabled. One use is to prevent accidental `EDBGREQ` signals stopping a core.

- `chan_xtrigs_reset`: Write 1 to clear all channel / trigger programming. Resets device hardware to default state.

The example below attaches input trigger index 1 to channel 2, and output trigger index 6 to the same channel. It then examines the state of the channel / trigger connections using the appropriate sysfs attributes.

The settings mean that if either input trigger 1, or channel 2 go active then trigger out 6 will go active. We then enable the CTI, and use the software channel control to activate channel 2. We see the active channel on the `choutstatus` register and the active signal on the `trigoutstatus` register. Finally clearing the channel removes this.

e.g.:

```
.../cti_sys0/channels# echo 2 1 > trigin_attach
.../cti_sys0/channels# echo 2 6 > trigout_attach
.../cti_sys0/channels# cat chan_free
0-1,3
.../cti_sys0/channels# cat chan_inuse
2
.../cti_sys0/channels# echo 2 > chan_xtrigs_sel
.../cti_sys0/channels# cat chan_xtrigs_trigin
1
.../cti_sys0/channels# cat chan_xtrigs_trigout
6
.../cti_sys0/# echo 1 > enable
.../cti_sys0/channels# echo 2 > chan_set
.../cti_sys0/channels# cat ../regs/choutstatus
0x4
.../cti_sys0/channels# cat ../regs/trigoutstatus
0x40
.../cti_sys0/channels# echo 2 > chan_clear
.../cti_sys0/channels# cat ../regs/trigoutstatus
0x0
.../cti_sys0/channels# cat ../regs/choutstatus
0x0
```