DsxvmHardware

This page explain how to describe the hardware in Dsx-vm.

General description

We first start by creating a Hardware object:

```
Hardware(cluster_x, cluster_y, nb_proc)
```

- `cluster_x` : number of cluster in the abscisse axe (int)
- `cluster_y` : number of cluster in the ordinate axe (int)
- `nb_proc` : number of proc by cluster

Both cluster_x and cluster_y parameter represent the number of cluster in the platform. If you have a non-clustered platform, you should then set each value to 1.

Peripehrals

Once we have described the platform, we could attach different peripheral to it:

- TIMER : a timer peripheral, useful for the scheduling of the processors Timer(name, pbase, channel_size, nb_channel)
- ICU : a concentrator of interrupt line, mandatory if we got irqs on the platform ICU(name, pbase, channel_size, nb_channel)
- XICU : a concentrator of interrupt line and a timer, this component and the couple Timer/Icu can't be in the same platform XICU(name, pbase, channel_size, nb_channel)
- Dma : a dma component Dma(name, pbase, channel_size, nb_channel)
- Tty : a multi-tty terminal Tty(name, pbase, channel_size, nb_channel)
- Fbf : a frame-buffer Fbf(name, pbase, channel_size, nb_channel)
- RAM : a RAM memory RAM(name, pbase, size)
- ROM : a ROM memory, we must have one at 0xbfc00000 address with the a minimal size of 0x1000 ROM(name, pbase, size)
- MwmrCoprocTaskWrapper : declare a coproc which is able to simulate the a C task by wrapping it in a SystemC coproc. The coproc (a SystemC module) will only be generated if a task have been mapped MwmrCoprocTaskWrapper(name, pbase, channel_size, nb_channel, sc_name)

The parameters are the following:

- name : name of the component
- pbase : physical base address of the component
- channel_size : the size of one channel, or the size of the component if it doesn't support multiple channel
- nb_channel : number of channel, set to '1' if no channel
- size : size of the memory
- sc_name : the SystemC name of the coproc

Irqs

This is a special component able who is able to describe the the routing of the interrupt line.

```
Irq(proc_id, cluster_id, icu_irq_id, peri, channel_id)
```

- `proc_id` : the proc_id at which the irq is attached
- `icu_id` : the icu_id line at which the irq is attached
- `cluster_id` : the cluster_id which contain the proc, icu and the peri
- `peri` : the class of the peripheral from which the request is sent
- `channel_id` : the channel_id of the peripheral

Examples

Here's an example for a one cluster platform:

```
#!/usr/bin/env python

from dsx.hard.hard import *


def AlmoArch(nb_proc = 4, nb_tty = 2, timer_pbase = 0x91000000, icu_pbase = 0x9F000000):

    cluster_x = 1
    cluster_y = 1
    nb_clusters = cluster_x * cluster_y
    hd = Hardware(cluster_x, cluster_y, nb_proc)

    ####### peripherals ########
    hd.add(Tty('tty', pbase = 0x90000000, channel_size = 16, nb_channel = nb_tty))

    for cl in range(nb_clusters):
        hd.add(Timer('timer%d'%cl, pbase = timer_pbase + (cl * hd.cluster_span), channel_size =
        hd.add(Icu  ('icu%d'%cl  , pbase = icu_pbase + (cl * hd.cluster_span)  , channel_size =


    ######## irqs ########
    for i in xrange(nb_proc):
        hd.add(Irq(proc_id = i , cluster_id = 0, icu_irq_id = i, peri = Timer , channel_id = i )


    ##### mwmr_coproc ######
    hd.add(MwmrCoprocTaskWrapper("tg_coproc"    , pbase = 0x94000000, channel_size = 32, sc_name
    hd.add(MwmrCoprocTaskWrapper("ramdac_coproc", pbase = 0x95000000, channel_size = 32, sc_name

    ######## Mems ########
    hd.add(ROM("PSEG_ROM", pbase = 0xbfc00000, size = 0x00010000))
    hd.add(RAM("PSEG_RAU", pbase = 0x00000000, size = 0x01000000))
    hd.add(RAM("PSEG_RAK", pbase = 0x80000000, size = 0x00100000))

    return hd
```

Another example for a multi-cluster platform:

```
#!/usr/bin/env python

from dsx.hard.hard import *


def ClusteredArch(cluster_x = 2, cluster_y = 2, nb_tty = 2, wcoproc = False):

    nb_proc = 1
    nb_cluster = cluster_x * cluster_y
    ram_pbase   = 0x00000000
    icu_pbase   = 0x00100000
    dma_pbase   = 0x00200000
    timer_pbase = 0x00300000
```

```
tty_pbase   = 0xFFD00000

hd = Hardware(cluster_x = cluster_x , cluster_y = cluster_y, nb_proc = nb_proc) #nb_proc : p

######### peripherals ##########
hd.add(Tty('tty',  pbase = tty_pbase, channel_size = 16, nb_channel = nb_tty))

for cl in range(nb_cluster):
    hd.add(Timer('timer%d'%cl,  pbase = timer_pbase + (cl * hd.cluster_span), channel_size =
    hd.add(Icu  ('icu%d'%cl,    pbase = icu_pbase + (cl * hd.cluster_span),   channel_size =

######### mwmr_coproc #########
hd.add(MwmrCoprocTaskWrapper("tg_coproc",      pbase = 0xFFE00000, channel_size = 32, sc_nam
hd.add(MwmrCoprocTaskWrapper("ramdac_coproc",  pbase = 0xFFF00000, channel_size = 32, sc_nam

############## IRQ ############
for j in xrange(nb_cluster):
    for i in xrange(nb_proc):
        hd.add(Irq(cluster_id = j, proc_id = i, icu_irq_id = i, peri = Timer, channel_id = i

############# MEMS ############
hd.add(ROM("SEG_ROM",  pbase = 0xbfc00000, size = 0x000F0000))

for cl in xrange(nb_cluster):
    hd.add(RAM("SEG_RAM_%d"%cl,  pbase = ram_pbase + (cl * hd.cluster_span), size = 0x001000


return hd
```