SID Modeling

Creating the components

Table of Contents

1. Introduction and Prerequisite

This manual guides you to create the SID components. It is a reference for you to configure and customize the AndESLive components provided by Andes, furthermore, to create your own components.

Before creating the SID component, you need to have the basic concept of the SID simulation environment. For this part, please go SID's website at http://sourceware.org/sid/ to have relative information and be sure to read the SID Developer's Guide at http://sourceware.org/sid/cdk-guide/book1.html to understand the relation between the components and SID.

Note: Before proceeding to the next section, we assume you already installed the AndeSight package.

2. Create a component

This section will instruct you to create a SID component and will divide all the procedures into three parts which are Installation of SID, Design your model using SID API, and Hook up to AndESLive.

2.1 Installation of SID Dev. Package (for Cygwin only)

To install the SID environment, you can choose either method provided as follows.

Method 1: Manual shell commands

1. Unpack **sid-nsd-dev-${date}.bz2** to working directory, let's name it as **$TOP**
   > tar jxf sid-nds-dev-${date}.bz2

   This step takes time, please be patient. After it's done, you can find the folder **sid-nds-my-cyg/** unpacked.
   > cd ${SID installation path}/sid-nds-my-cyg/

2. Initialization
   > make install-sid
   Now, you can proceed to create your components.

Method 2: Shell script (provided on the download website)

1. Change working directory to where the package will be installed
   For example, **$TOP**

2. Run the **install** script, and give the path of **.bz2** to it.
   > sh ./install $path/sid-nds-dev-${date}.bz2

This will automatically complete the installation.

2.2 Design your model using SID API

2.1.1 Create your model

In order to provide the more convenient way to create the SID component, we create a template to let you use it as the framework and via inputting the necessary data, your component will be created more easily.

1. To create the template, please follow the steps below and input variables as you need.

   > cd $TOP/sid-nds-my-cyg/

   > make new-component

   Top folder path:
   Component folder name: my-comp
   Library file name: libmycomp
   Class name: MyComp
   SID configuration name: hw-my-comp
   yes to proceed: yes

   top: /cygdrive/d/wrk/sid-nds-my-cyg/.
   Component 'my-comp' source template are created at
   /cygdrive/d/wrk/sid-nds-my-cyg/./src/sid/component/my-comp
   Component 'my-comp' build are created at
   /cygdrive/d/wrk/sid-nds-my-cyg/./bld/sid/component/my-comp

2. Define and add the component abstracts and functionalities to this template and save it. For more detail information, please refer to CDK tutorial at http://sourceware.org/sid/cdk-guide/ch-tutorial.html

2.1.2 Build your model and link it with SID

1. Setup the environment (bash/Cygwin)

   > make env

   This just dumps the following 'short-cuts' but not to apply it. You would apply any of them with or without modifications. Applying these short-cuts is just for convenience and multi-lines copy-paste would help you a lot.

export TOP=${TOP}
export SID=${TOP}/ins/bin/sid
export SRC=${TOP}/src/sid/component/my-comp
export BLD=${TOP}/bld/sid/component/my-comp
export TST=${TOP}/tst/my-comp
export DYN=${TOP}/src/sid/main/dynamic

2. Add link entry to "sid.exe" within **$DYN/Makefile.am**
   > cd $DYN
   > vi Makefile.am

   Add the following line after the line of "NEARBY_LIBS = \"
   **-dlpreopen ../../component/my-comp/libmycomp.la **
   And add following line after the line of "NEARBY_DEPS_LIBS = \"
   **../../component/my-comp/libmycomp.la **

   > automake

   Running **automake** is to re-create **Makefile.in** from **Makefile.am**. And, running **config.status** in the **$TOP/bld/sid/main/dynamic** would re-generate **Makefile** from **Makefile.in**. **Make** is supposed to be aware of changes of its dependences, and then do the re-generation of **Makefile** automatically.

3. 'Make' new component
   > cd $BLD
   > make

4. Install new SID
   > cd $TOP
   > make sid

5. Test
   > cd $TST
   > make run

Note: 1. Please ignore error message "*module error, handle = 10211c68, tmp = 1022e8b0*", if you got it on a 64-bit x86 machine.
2. Those who would like to change the source file name, or to add new files, should modify or add source names within **Makefile.am** in source code

folder. After modification, invoke **automake**.

Make would be aware of this change and re-generate **makefile** when you make next time.

eg. add timer2.cxx to **sid/component/my-comp/**

modify **sid/mycomp/Makefile.am**

libmytimer_la_SOURCES = timer.cxx timer2.cxx

^^^^^^^^^^

> automake

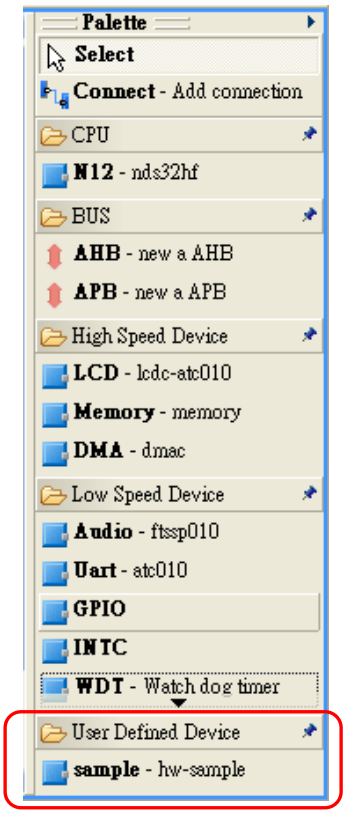
> cd $BLD

> make

2.3 Hook up to AndESLive

To hook up the component you just created in *AndESLive*, you need to finish the three steps as the follows.

First of all, you need to write a component descriptor. Second, put them into **$ANDESIGHT_ROOT\vep\component\user**. Third, restart AndeSELive and you will see it appear in category "**User Defined Device**" of device palette.



The most important part is to understand how to write a descriptor that specifies your component properties. The detail will be described in the following sections.

2.3.1 Define component

Use tag **defcomponent** to define a component.

```
<?xml version="1.0" encoding="UTF-8"?>
<defcomponent name="hw-sample" shortname="sample">
    <sid-lib dlsym="sample_component_library" name="libsample.la" />
    …
</defcomponent>
```

Defcomponent::name sid component name use to new your component
Defcomponent::shortname name to show in the VepBuilder
sid-lib::dlsym dynamic linking symbol for your component library
sid-lib::name share library file name

2.3.2 Define buses

The following defines bus master port named “master” and bus slave port named “registers” which use AHB protocol.

```
    <!--sid bus accesor-->
    <busmaster name="master" type="AHB"/>
    <!--sid bus interface-->
    <busslave name="registers" type="AHB"/>
```

busmaster::name port name
busmaster::type AHB or APB

2.3.3 Define pins

It’s quite simple to define a signal pin with the pin name and direction.

```
    <!--pin-->
    <defpin name="out" direction="out"/>
    <defpin name="in" direction="in"/>
```

For pin that dedicates for issue or receive interrupt signal.

```
<!--interrupt pin-->
<defpin name="intr" direction="out" type="interrupt"/>
```

If you have series of pins needed to define. Use **defpins** tag.

```
<!--define multiple pins-->
<defpins name="sample-out-" from="0" to="3" direction="out"/>
<defpins name="sample-in-" from="0" to="3" direction="in"/>
```

Above is equivalent to following tags.

```
<defpin name=" sample-out-0" direction="out"/>
<defpin name=" sample-out-1" direction="out"/>
<defpin name=" sample-out-2" direction="out"/>
<defpin name=" sample-out-3" direction="out"/>

<defpin name=" sample-in-0" direction="in"/>
<defpin name=" sample-in-1" direction="in"/>
<defpin name=" sample-in-2" direction="in"/>
<defpin name=" sample-in-3" direction="in"/>
```

2.3.4 Define attributes

```
<!--sample attributes-->
<defattribute category="setting" name="string" default="any string"/>
<defattribute category="setting" name="integer" type="integer" default="-65535"/>
<defattribute category="setting" name="positive integer" type="positive integer"
    default="65535"/>
<defattribute category="setting" name="option" type="{opt0, opt1, opt2}"
    default="opt0"/>
<defattribute category="setting" name="range_ex" type="[0..100]" default="50" />
```

Type description:

- string
- integer
- positive integer
- option ex: {0, 1}, {true, false}, {red, green, blue}

- range [l..u] l=lower bound, u=upper bound, ex: [0..50]

2.3.5 Schedule event and time

Define a clock input for this component, and they will be connected to SID scheduler component according to your clock setting in the VEP Builder.

defclockinput::name	name for the clock input (It will show up in the clock setting view.)
defclockinput::event	event pin name"
defclockinput::control	control pin name"

```
<defclockinput
    name="clockIn"
    event="event"
    control="control"
/>
```

Define a time query client for your component if you have one.

```
<time-query-client
    query="time-query"
    high="time-high"
    low="time-low"
/>
```

2.3.6 Life cycle control pin

There are some events relevant to SID simulation environment and component life cycle can be listened by a component via driving its pin.
You can specify component life cycle control pin by **control-pin** tag and corresponded pin will be driven when the system initializes, de-initializes or resets. If you don't want to describe the event, just ignore it.

```
<control-pin
    init="init"
    deinit="deinit"
    reset="reset"
```

/>

Note: The above description is also provided in the **$ANDESIGHT_ROOT\vep\component** directory as the example **sample.comp.xml**

3. Write SW for your model

So far, you have finished creating the component and from now on, you can write the software to access registers of your model and run software and hardware co-simulation.

We provide a software access example **tst-my-comp.c** for your reference in the **$TOP/tst/my-comp**. And to demonstrate the co-simulation process, we also provide a configuration file **tst-my-comp.conf** which simulates the whole environment including the simple example component.

/cygdrive/d/sid-develop/20061208/sid-nds-my-cyg/tst/my-comp> **make run**

Result printed as follow:

$SID `pwd`/tst-my-comp.conf

Set cpu register <ir14> 0x00000002

Get cpu register <ir14> 0x00000002

Polling mode test

Set timer register <LOAD_REG> 0x00000028

Get timer register <VAL_REG> = 0x00000028

Get timer register <VAL_REG> = 0x00000028

Get timer register <VAL_REG> = 0x00000028

Get timer register <VAL_REG> = 0x00000001

Get timer register <VAL_REG> = 0x00000001

Get timer register <VAL_REG> = 0x00000001

Get timer register <VAL_REG> = 0x00000001

Get timer register <VAL_REG> = 0x00000001

Get timer register <VAL_REG> = 0x00000001

Get timer register <VAL_REG> = 0x00000028

Interrupt mode test

Set timer register <LOAD_REG> 0x00000190

Set timer register <CLR_REG> 0

Set timer register <CLR_REG> 0

Set timer register <CLR_REG> 0

Set timer register <CLR_REG> 0

Set timer register <CLR_REG> 0

Set timer register <CLR_REG> 0

Set timer register <CLR_REG> 0

Set timer register <CLR_REG> 0

Set timer register <CLR_REG> 0

Set timer register <CLR_REG> 0