PKtool overview (release 2.3.0)

Dipartimento DII, Università Politecnica delle Marche (Italy)

1 INTRODUCTION

PKtool (Power Kernel tool) is a simulation environment dedicated to power analysis for digital systems modelled in SystemC/C++. The main output result provided by PKtool analysis is the estimation of power dissipation, under specific power models and operative conditions. PKtool has been developed by DII department, at Università Politecnica delle Marche (Italy), where currently a research project is enabled for improving and extending the capabilities of the tool.
This document presents an overview of the PKtool version 2.3.0, which is distributed in an open-source release downloadable from http://pktool.sourceforge.net/. PKtool 2.3.0 is compatible with the SystemC 2.3.1 version provided by Accelera and downloadable from http://www.accellera.org.
Like SystemC, PKtool is based on a class library developed in C++ language; its application needs the same hardware/software resources necessary for defining and simulating an ordinary SystemC description. The realization of PKtool analysis requires some systematic steps that allow to define a suitable configuration for the examined systems. Technical and application details are discussed more deeply in a specific user manual, whereas installation instructions are provided in a separate text file. Both these documents can be found enclosed in PKtool releases.
The remainder of this document reports an introductory overview of the main PKtool features and components. The contents are organized as follows:

- section 2: fundamental aspects concerning application modalities and simulation flow.
- section 3: power model handling.
- section 4: features and application of augmented signals.
- section 5: instance of power_modules.
- section 6: extension for transaction level power analysis.

2 APPLICATION AND SIMULATION BASIC FEATURES

PKtool can be directly applied onto the single modules of a system modelled in SystemC/C++. The module abstraction is realized in SystemC by means of an entity called *sc_module*; therefore, the application of PKtool is referred to the sc_modules constituting a SystemC description. This operation consists in the definition and instance of new components called *power_modules*. A power_module is a PKtool entity that allows to extend the internal behaviour of a traditional sc_module for PKtool analysis. This enhancement mainly consists in the linkage to a power model together with additional capabilities for power estimation tasks.
In order to select an sc_module for PKtool analysis, it is necessary to replace its original instance with a matching power_module. This replacement can be made selectively, considering only those sc_modules whose power dissipation is wanted to be estimated. At the end of a PKtool analysis, the output results are given by the distinct power estimations of each instanced power_module.
A PKtool analysis consists in a simulation session involving all the instanced power_modules. All the execution and synchronization tasks are handled by an ad-hoc simulation engine called *Power Kernel*.

This component is realized on the basis of the classical SystemC simulation kernel, in the form of an extension customized for power analysis. All that allows to execute a PKtool simulation during an ordinary SystemC simulation of the monitored system. More precisely, SystemC and PKtool simulations start and end in the same times, constituting apparently a unique session. In such context, Power Kernel acts simultaneously with the SystemC kernel, in a hidden and non-intrusive way, without affecting the correct behaviour of this latter. The progress and results of the SystemC simulation are the same of an analogous session without applying PKtool. The power estimations provided by PKtool are referred to the system evolution as reproduced by the SystemC simulation.

3 POWER MODELS AND MODEL DATA

A power model is a formal definition of the power dissipated by a digital system, commonly given by an analytical/algorithmic formulation. A power model is usually not aimed at providing an exact value of the power dissipation but rather an acceptable estimation. The accuracy of such estimation may depend on several power model features, such as the reference abstraction level, the computational complexity, the amount of data required.
PKtool is not associated to a particular power model, but is linked to an internal power model library that makes available several power models. During a PKtool simulation, each power_module has to be associated to a specific power model that will be evaluated for estimating the sc_module power dissipation. The association between power_modules and power models is carried out at the beginning of the simulation by the user, who can select the power models to be applied among the ones available in the PKtool model library.
The application of a power model is usually based on specific data required in its formulation (model data). Such data are often referred to information characterizing the sc_module to which the power model is applied, e.g. technology and architectural features. From an operative point of view, we can subdivide model data into two distinct categories:

1) data known a priori, available before the beginning of a simulation.

2) data available only during the simulation, on the basis of the run-time evolution.

From now on, we can refer to these two categories respectively as *static data* and *dynamic data*. Typical examples of static data may be given by technology parameters; typical examples of dynamic data may be given by signal information such as switching activity.
PKtool implements different solutions for the acquisition and the handling of static and dynamic data, in compliance with their specific availability. More precisely, static data are specified by the user at the beginning of a PKtool simulation, through an interactive procedure. As regards dynamic data, PKtool makes available ad-hoc means addressed to their handling, such as *augmented signals*. The next section reports a general description of augmented signals and their use.

4 AUGMENTED SIGNALS

Augmented signals are PKtool components targeted to provide signal data often required by a power model. An augmented signal can be considered as a smart signal, capable to show a traditional behaviour with additional abilities to compute and make available power-related information. Augmented signals are provided by PKtool through a framework of augmented signal types. From the user's point of view, the application of augmented signals consists in instance instructions that entail simple

modifications in the code of an sc_module. As an example let us consider the following description, which represents the class definition of an sc_module called *example_mod*:

```
SC_MODULE(example_mod)
{
  // input ports
  sc_in<sc_uint<32> > in_1, in_2;
  sc_in<bool> reset;
  sc_in_clk clk;

  // output port
  sc_out<unsigned> out;

  // internal signals
  sc_uint<3> ctr_1;
  sc_uint<2> ctr_2;

  // rest of the code
  ...
}
```

The above code shows only the most relevant parts, in particular the signals instanced inside the sc_module class. We can notice three input ports (in_1, in_2, reset), one clock port (clk), one output port (out), and two internal signals (ctr_1, ctr_2).
If we want to select some of these signals for monitoring their run-time commutations, we must convert them into augmented counterparts. This operation is realized by replacing the original type of each signal with a matching augmented type. In particular, if we want to monitor the input ports in_1 and in_2, we should modify the previous description in this way:

```
SC_MODULE( example_mod)
{
  //input ports
  sc_in_aug<sc_uint<32> > in_1, in_2;
  sc_in<bool> reset;
  sc_in_clk clk; 4

  //output port
  sc_out<unsigned> out;

  // internal signals
  sc_uint<3> ctr_1;
  sc_uint<2> ctr_2;
  ...
}
```

where the ports in_1, and in_2 have been replaced with the type sc_in_aug<sc_uint<32> >. This latter represents the augmented version of the original type sc_in<sc_uint<32> >. At this point, in_1 and in_2

have become augmented signals and have gained all the specific capabilities. In particular, during a PKtool simulation these signals are able to sample their run-time commutations for power model evaluation. As shown by the example, instancing augmented signals is a selective task that may involve a subset of the sc_module signals. During a PKtool simulation, all the signals not augmented simply retain their original behaviour.
The instance of augmented signals represents the main modification to be made on the original code of a monitored sc_module. However, such intervention is not strictly necessary, but should be considered only if the applied power model requires specific signal information in its formulation. With reference to transaction level applications, PKtool provides an augmented signal specialization called *augmented socket*.

5 POWER_MODULES

In order to select an sc_module for PKtool analysis, the user must define and instance a corresponding power_module. A power_module is the component that allows the interaction between an sc_module and the PKtool simulation engine, representing also the place where some configuration settings can be specified. From an external point of view, in particular with regard to the I/O ports, a power_module retains the original sc_module structure.
As an example to show how a power_module is handled, we can consider the sc_module introduced in the previous section. The power_module conversion requires to realize two main steps. First of all, the power_module class must be defined:

```
POWER_MODULE_CLASS ( example_mod )
{
 // implementation details
 ...
}
```

The class title is expressed by the macro POWER_MODULE_CLASS, with the name of the sc_module as parameter. All the elements that can be defined in the class body are widely explained in the user manual.
The second step consists in the instance of power_module objects. We can consider this task through the following description, which represents a top-level sc_main function:

```
int sc_main ()
{

  // connection signals

  sc_signal<unsigned> insig_1, insig_2;
  sc_signal<bool> insig_3;
  sc_signal<sc_int<32> > outsig_1, outsig_2;



  // sc_module instances and connection instructions
```

```
  example_mod mod_1 ("mod_1");
  mod_1.in_1(insig_1);
  mod_1.in_2(insig_2);
  mod_1.reset(insig_3);
  mod_1.out(outsig_1);

  example_mod mod_2 ("mod_2");
  mod_2.in_1(insig_1);
  mod_2.in_2(insig_2);
  mod_2.reset(insig_3);
  mod_2.out(outsig_2);
  ...

};
```

In the function there are reported the instances of two modules, mod_1 and mod_2, with the related connections. If we want to select mod1 for PKtool analysis, we must turn it into a matching power_module. For this purpose, it is necessary to replace the original type of mod_1 with the matching power_module type. This is achieved by modifying the instance instruction in this way:

```
POWER_MODULE(example_mod) mod_1 ("mod_1");
```

where the original type has been wrapped in the macro POWER_MODULE. This is the only intervention to be done; due to the external equivalence between modules and power_modules, all the connection instructions are not to be modified. As concerns the other module (mod_2), during a PKtool simulation it retains its traditional behaviour, since its original type has not been converted into a power_module counterpart.

6 EXTENSION FOR TRANSACTION LEVEL POWER ESTIMATIONS

The most recent developments of PKtool have led to an extension for transaction level power analysis. Such extension can be referred to as PKtool/TLM and is applicable jointly with the TLM 2.0 framework of SystemC (SystemC/TLM). More precisely, PKtool/TLM allows to configure power analysis on transaction level models described in SystemC/TLM. For this purpose, PKtool/TLM introduces some enhancements in the basic PKtool capabilities for a proper adaptation to the main constructs of SystemC/TLM. A comprehensive treatment of PKtool/TLM and its applications is reported in a specific documentation that integrates the PKtool user manual.