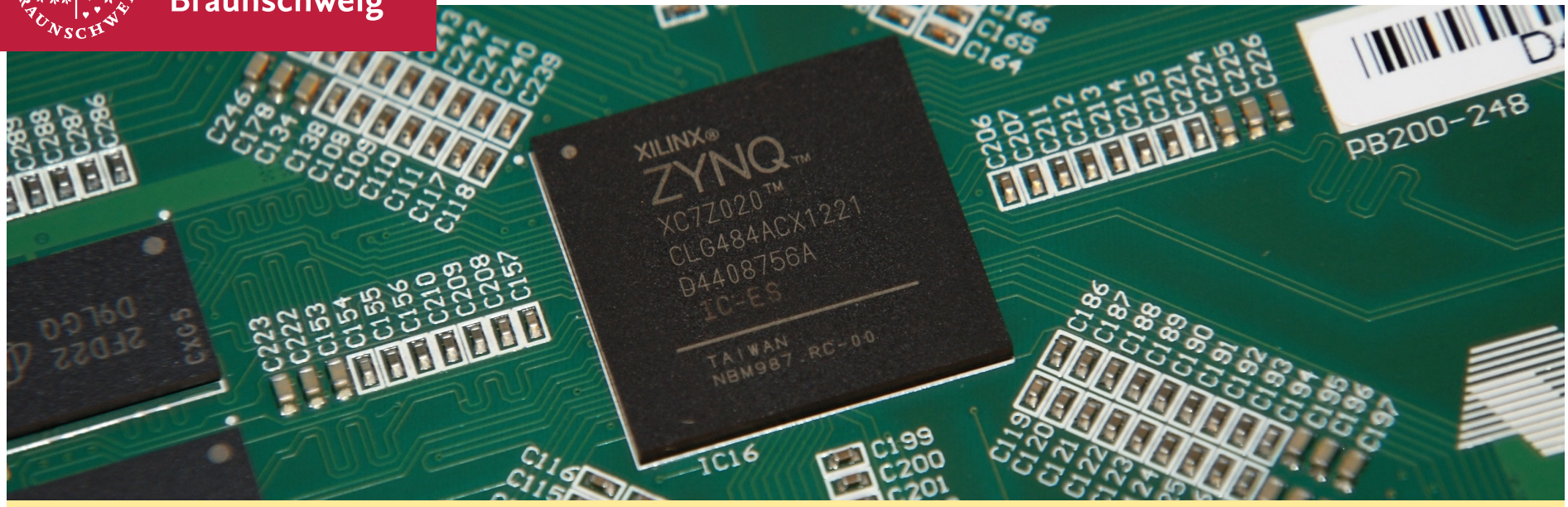
Technische Universität Braunschweig

Chair for Chip Design for Embedded Computing



SoCRocket Week 2014

Introduction of SoCRocket

Rolf Meyer, 30. Juni 2014

Teil I

Introduction

Contens

- **File structure**
- **Python - Crash course**
- **Build System**
  - Waf
  - Python tools

File structure - 1

- /contrib
  - Patches for external dependencies
  - Rewritten dependencies
  - Own dependencies
- /common
  - Library for common used classes and functions
  - Small platform helpers
  - VerbosityKit, MsgLogger, Power, EndianessHelper
- /models
  - Base directory for all models
  - Each model is a library in an own subdirectory
  - AHBCtrl, APBCtrl, AHBMem, Irqmp, ...

File structure - 2

- /models/utils
  - Suporting infrastructure to build models
  - APBDevice, AHBDevice, AHBMaster, AHBSlave, MEMDevice
- /platforms
  - Basedirectory for all paltforms
  - All platforms formed from the models are stored in this directory
  - leon3mp.platform, yuvstream.platform, mips1sp.platform
- /templates
  - Basedirectory for the configuration files of a platform
  - Needed for the generator wizard
  - XML Templates (.tpa), Configfile (.json)

File structure - 3

- /adapters
  - VHDL-Transactors
- /tools
  - Home to all scripts
  - Including generator wizard and build system scripts
  - get_json_attr, set_json_attr, list.sh

Python - Explaination of the Syntax

- Most languages use curly braces to group pieces of code.
- Like everything in a for loop will be surrounded by a { and a }.
- Not in Python.
- Indention is the key.
- Everything on the same indention level is part of the same group of code.

Python - Loops

```
for loop stuff:
    weee
    fun
this_isnt_part_of_the_loop()
```

Each statement is (usually) on its own line whereas a semicolon is used to delimit lines in most other programming language syntaxes.

Python - Functions

Functions are defined using the def statment. It goes like this...

```
def functionname(parameter1, parameter2):
    code goes here
    and here
    etc.
```

Python - Classes

Classes are defined by the class statement.

```
class Classname(ParentClass):
    def __init__(self):
        self.member = 5
```

Member functions always have the first prameter self. It is the equivalent to this.

Python - Beware the Indetion!

And you can have more indentions under a def in the same way you would have used curly braces before...

```
def foo():
    do stuff
    do more stuff
    for loop time:
        this is part of the loop
        so is this
    but this is not, this is just part of the rest of the
        function
```

Python - Lists

- Lists are very versatile.
- Lists do not all have to contain the same type of data.
- It's ok to mix strings and numbers and objects and nested lists, etc.

```
shopping_list = ["bread","milk","hamster␣brains"]
```

then if you want to add an item to the list...

```
shopping_list = shopping_list + ["turnips"]
```

Python - Lists access

If you want to access just one item in that list you can do it like this...

```
shopping_list[0]
```

That's the first item in the list.

```
shopping_list[1]
```

would be the 2nd, etc.

Python - Lists range

If you want a range of items in the list you can do that with colon magic...

```
shopping_list[0:3]
```

Accesses the first 3 items in the list.

- The way it works is the first number is where you want to start
- The second number is where you want to end (not how many you want).
- It goes up to but not including the 2nd number.
- Leave them empty to go frome the start/to the end.
- `shopping_list[:]` gets a copy of the hole list.

Python - For loops

This is the biggest change from most languages:

- Traditionally, a variable increments until a condition is false.

For loops in Python:

- The interpreter goes throug to each item of a list with a variable
- Making the variable equal to each one as it goes...

```
for thingy in shoppin_list:
    buy(thingy)
```

there's a loop (assuming we're using the same shopping list from earlier)

Python - Loop over integers

If you want a list of numbers that go from x to y, here's how you do that:

```
list_of_nums = range(x, y + 1)
```

- Then list of nums will be a list of numbers...in that range.
- It always starts at the first number and goes up to (but not including) the 2nd number.
- That's what that + 1 thing is in there for.

Python - If statements

```
if some condition:
    do this
```

Simple enough:

```
if x > 4:
    print "OMG, x must be HUGE!"
```

Python - Or else!

Then there's else:

```
if x > 4:
    print "OMG, x must be HUGE!"
else:
    print "x ain't nuttin'"
```

Python - Or else if!

else if's are denoted by elif's

```
if x > 4:
    print "OMG, x must be HUGE!"
elif x == 4:
    print "x is four"
else:
    print "x is tiny"
```

Python - While loops

While loops look just like if's and work like how you would imagine them

```
while condition:
    do this
```

Python - Strings

Strings are concatenated with + signs

```
greeting = "Hello, " + "World!"
```

To define a string you can use ", ', ör '''. This saves the hassle of escaping. And vprintf replacment ist build in (`%s, %d, %f...`):

```
name = "%(first)s %(last)s" %  {
    'first':name[0],
    'last':name[1]
}
greeting = """Hello %s""" % name
```

Python - == and =

== and = are just like most languages:

```
foo = "yo"
if (foo == "yo"):
    print "foo is yo"
```

Python - Comments

Comments are done with a #:

```
print "this is some code" #this line prints stuff
```

Python - And and Or

There are no && or ||, it's just and and or

```
if (x == 2) and (y > 5):
    do_stuff()
```

Python - But there is much more

- There are tonns of usefull features in the python language.
  - Go to "https://docs.python.org/2/tutorial/index.html"for a full tutorial.
  - Language references you can finde here:
  - https://docs.python.org/2/reference/index.html
- And like Java Python comes with a standard libray
  - Which is full of useful things
  - os, json, base64, HTMLParser, shlex, etc.
  - https://docs.python.org/2/library/index.html

Build System

- Waf 1.7
- Python based
- Split into serveral files: “wscript”s and python scripts!
- It’s simple:

```
./configure; make; make install
```

For waf:

```
./waf configure; ./waf build; ./waf install
./waf configure build install # in short
```

Build System

- Build ist default!
- To reduce build time compile only needed targets

```
./waf --target=<target>
```

Usefull:

```
./waf list      # List all targets
./waf clean     # Clean build dir
./waf distclean # Remove build dir
```

Waf - Top-level wscript

The top-level wscript is the collection point to a common build system and initializes all the tools needed to build.
Most of them are outsourced in python scripts which residents in /tools/waf/ The only remains are the phase functions and call hooks:

```
#! /usr/bin/env python
# vim : set fileencoding=utf-8 expandtab noai ts=4 sw=4
    filetype=python :
APPNAME = 'SoCRocket'
top = '.'
out = 'build'

import sys
from waflib.Tools import waf_unit_test
from tools.waf.logger import Logger

LOAD = ['compiler_c', 'compiler_cxx', 'python',
        'swig', 'waf_unit_test', ...]
```

Waf - Top-level wscript 2

```
TOOLS = ['common', 'flags', 'virtualenv', 'pthreads',
         'boost', 'modelsim', 'systools', ...]

def options(self):
    self.load(LOAD)
    self.load(TOOLS, tooldir='tools/waf')

def configure(self):
    self.load(LOAD)
    self.check_waf_version(mini='1.6.0')
    self.check_python_version((2,4,0))
    self.check_python_headers()
    self.load(TOOLS, tooldir='tools/waf')

def build(self):
    self.recurse_all()
```

Waf - Recursion wscripts

Some wscripts are just needed for reaching further subdirectories: Models, platforms, software, . . .

```
#! /usr/bin/env python
# vim : set fileencoding=utf-8 expandtab noai ts=4 sw=4
    filetype=python :
top = '..'

def build(self):
  self.recurse_all()
  self.recurse_all_tests()
```

The "recurese_*members come from the common file (/tools/waf) included in the top-level wscript as first tool

Waf - Models wscript

Each models is a static C++ Libary! Nothing more

```
#! /usr/bin/env python
# vim : set fileencoding=utf-8 expandtab noai ts=4 sw=4
    filetype=python :
top = '../..'

def build(self):
  bld(
    target          = 'gptimer',
    features        = 'cxx cxxstlib',
    source          = 'gptimer.cpp gpcounter.cpp',
    export_includes = '.',
    uselib          = 'BOOST SYSTEMC TLM AMBA GREENSOCS',
    use             = 'common signalkit utils',
    install_path    = '${PREFIX}/lib',
  )
```

Waf - Platforms wscript

Tests or Platforms are C++ applications linked with all libraries needed for execution

```
#! /usr/bin/env python
# vim : set fileencoding=utf-8 expandtab noai ts=4 sw=4
    filetype=python :
top = '../..'

def build(self):
    bld(
        target          = 'leon3mp.platform',
        features        = 'cxx cprogram pyembed',
        source          = 'sc_main.cpp',
        includes        = '.',
        use             = 'ahbctrl irqmp gptimer apbctrl ...'
        uselib          = 'TRAP BOOST ELF_LIB SYSTEMC ...'
    )
```

Waf – Target software wscripts

Waf can be used to compile Target software exactly the same as for host software (Models/Libraries or Platforms/Applications):

```
#! /usr/bin/env python
# vim : set fileencoding=utf-8 expandtab noai ts=4 sw=4
    filetype=python :
top = '../..'
def build(self):
  bld(
      features      = 'c␣cprogram␣sparc',
      target        = 'hello.sparc',
      cflags        = '-static␣-g␣-O1␣-lm␣-mno-fpu',
      linkflags     = '-static␣-g␣-O1␣-lm␣-mno-fpu',
      source        = ['hello.c'],
      install_path = None,
  )
```

The only difference is the feature `sparc`

Waf – Unittests

If a test shall be performed on a host target just add the feature `test`. If it needs to be executed in a specific platform (host and target code) use a systest:

```
#! /usr/bin/env python
# vim : set fileencoding=utf-8 expandtab noai ts=4 sw=4
    filetype=python :
top = '../..'

def build(self):
  bld(
      features     = 'systest',
      system       = 'leon3mp.platform',
      rom          = 'sdram.prom',
      ram          = 'hello.sparc',
  )
```

Waf – Modelsim Targets

The feature modelsim uses modelsim to compile. You can specify it as a unittest too.

```
#! /usr/bin/env python
# vim : set fileencoding=utf-8 expandtab noai ts=4 sw=4 ...
top = '..'
def build(self):
  bld(
    target   = 'gptimer.1.rtl.test',
    features = 'modelsim test',
    source   = ['gptimer_wrapper.vhd',
                'test1/testbench.cpp', 'test1/top_rtl.cpp',
                '../../../adapters/rtltransactor.cpp',
                '../../utils/clkdevice.cpp',
                '../../utils/apbdevice.cpp'],
    uselib   = 'AMBA GREENSOCS BOOST',
    includes = '. test1 ../../utils',
  )
```

Waf Python Tools - Common

Common registes a basic set of tools to compile SoCRocket.

- The recursion utilities
- helper for fetching and compiling dependencies

Waf Python Tools - Configuration

In the configuration phase all dependencies are checked and stored

```
#! /usr/bin/env python
# vim : set fileencoding=utf-8 expandtab noai ts=4 sw=4
    filetype=python :
top = '..'

def find(self):
    self.check_cxx(
        lib='elf',
        uselib_store='ELF_LIB',
        mandatory=True,
        libpath = libpath,
        errmsg  = "not found"
    )

def configure(self):
    find(self)
```

Waf Python Tools - Deps

Moreover socrocket is fetiching missing dependencies (via tar or git):

```
#! /usr/bin/env python
# vim : set fileencoding=utf-8 expandtab noai ts=4 sw=4 ...
top = '..'

def configure(self):
    try:
        find(self)
    except:
        name    = "libelf"
        version = "0.8.13"
        self.dep_build(
            name    = name, version = version,
            tar_url = "http://www.mr511.de/%(base)s.tar.gz",
        )
        find(self, self.dep_path(name, version))
```

Teil II

ToDos

Contens

- **What needs to be done**

ToDos

- Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam
- At vero eos et accusam et justo duo dolores et ea rebum.
  - Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet!
    - Nam eget dui.
    - Maecenas tempus, tellus eget condimentum rhoncus, sem quam semper libero, sit amet adipiscing sem neque sed ipsum.
  - Duis leo
- Aliquam lorem ante, dapibus in, viverra quis, feugiat a, tellus.

SoCRocketDevice

- A common parant device to derive from.
- All models should derive from it (over AHBDevice, APBDevice)
- A common class to implement API calls

GS_Config Parameter

- All platforms use gs_params to implement component generics like in grlib designs.
- But these parameters are generics of the components/models.
- They should be implemented inside the models.
- A initializer function called `init_configuration()` should be present in each model.
- In this method all descriptions and attributes from the `.tpa`-file should be added.

Register Init

- Like the gs_config specific function `init_configuration()` a
- In this method all descriptions and attributes from the `.tpa`-file should be added.

sc_report VerbosityKit backend

Documentation

- Update Doxygen documentation from the user manual.
- Each directory needs to have a `<model>.md`
- `doc/*.md` for common chapters from the user manual and hldse

Documentation guid

Coding Style

We will follow the google coding guidlines (It's not perfect but already existing, including tools)

- http://google-styleguide.googlecode.com/svn/trunk/cppguide.xml

Make use of coding style tools like:

- Cpplint.py (from goole)
- OCLint
- clang sanitizer

#define Guards

- All header files should have #define guards to prevent multiple inclusion.
- The format of the symbol name should be <PROJECT>_<PATH>_<FILE>_H_.
- To guarantee uniqueness, they should be based on the full path in a project's source tree.
- For example, the file foo/src/bar/baz.h in project foo should have the following guard:

```
#ifndef SOCROCKET_MODELS_GPTIMER_GPTIMER_H_
#define SOCROCKET_MODELS_GPTIMER_GPTIMER_H_
//...
#endif  // SOCROCKET_MODELS_GPTIMER_GPTIMER_H_
```

includes

- All of a project's header files should be listed as descendants of the project's source directory.
- Without use of UNIX directory shortcuts . (the current directory) or .. (the parent directory).

For example, socrocket/models/gptimer/gptimer.h should be included as

```
#include "models/gptimer/gptimer.h"
```

Texts

This will break our project structure

But everythng will get more consistent.
every target needs to include self.root (waf)