Save compilation and link time

- Introducing dynamic creation overhead
- No need to recompile and link the design multiple times
- Introduces small instantiation overhead
- Allows scripting flexibility into the simulation

Top-Level Python script

```python
import usi

class Top(usi.Module):
    def __init__(self, modulename):
        super(Top, self).__init__(modulename)
        self.ahbctrl = usi.registry.AHBCtrl('ahbctrl', rrobin=True)
        self.ahbctrl.ahbOUT.bind(self.apbctrl.ahb)
        # ...

top = Top("top")
usi.start()
```

Registration of a simple sc_model

```c
#include <systemc>
#include <sr_registry>

SR_HAS_MODULE(AHBCtrl);

class AHBCtrl : public sc_module {
    // ...

    // Registers a sc_module class
    // If more constructor parameters than the class name are needed a
    // constructor function needs to be provided.
```

Dynamic library loading

- Collect loaded Libraries
- Create tempdir
- Symlink all libraries into tempdir
- Inclue tempdir as LD_LIBRARY_PATH in environment
- Replace Simulation with itself and modified environment
- Run normal Simulation
- Directly load Libraries via dlopen

Support for advanced scenarios
- Binary distribution of models
- Runtime extendability of simulations

Sources

The implementation is online available to everyone:

- **Source code:** https://github.com/socrocket/sr_registry
- **Licence:** Apache=2.0

DVCon US 2017, San Joes, California