FLEXIBLE INDIRECT REGISTERS WITH UVM

Uwe Simm – Cadence Design

cadence®

Terminology

- A ’normal’ register: Data := read(ADDR)
- indirect register: Data := read(getAddress(INDEX))
- Data register: register that performs the indirect operation when accessed

There are lots of wishes

- Set of registers indexed by a register field
- Set of registers indexed by a whole register
- Set of registers indexed by multiple fields (multi-dim)
- Set of fields indexed by other field
- Set of registers indexed by reg/wire/port
- Set of registers selected via pattern match from reg
- …

- Obviously any combination of the above ☹

Problem Statement

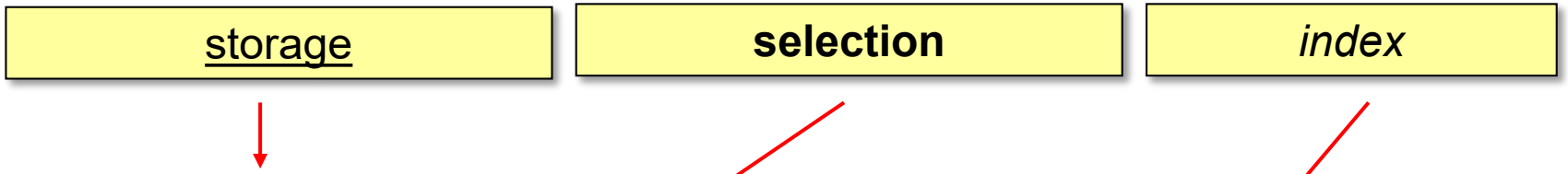
- It is impossible to enumerate or implement all of them entirely in a library
- UVM just has a single ‘kind’ of ‘indirect’ register

- Can we structure a framework
  - to support arbitrary storage, index and selection?
  - into a generic core and a user side?
  - that is user extensible?

Object Oriented Design

- Separate what changes (or is unknown) from what stays the same
- Program to an interface
- Encapsulate behaviors into objects

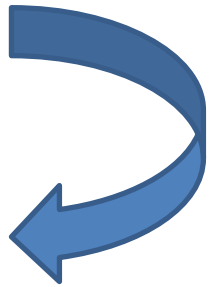
Revisiting Engineers Wishes



- Set of registers **indexed** by a *register field*
- Set of registers **indexed** by a *whole register*
- Set of registers **indexed** by *multiple fields*
- Set of fields **indexed** by other *register field*
- Set of registers **indexed** by *reg/wire/port*
- Set of registers **selected via pattern** from *reg*

Structuring a solution

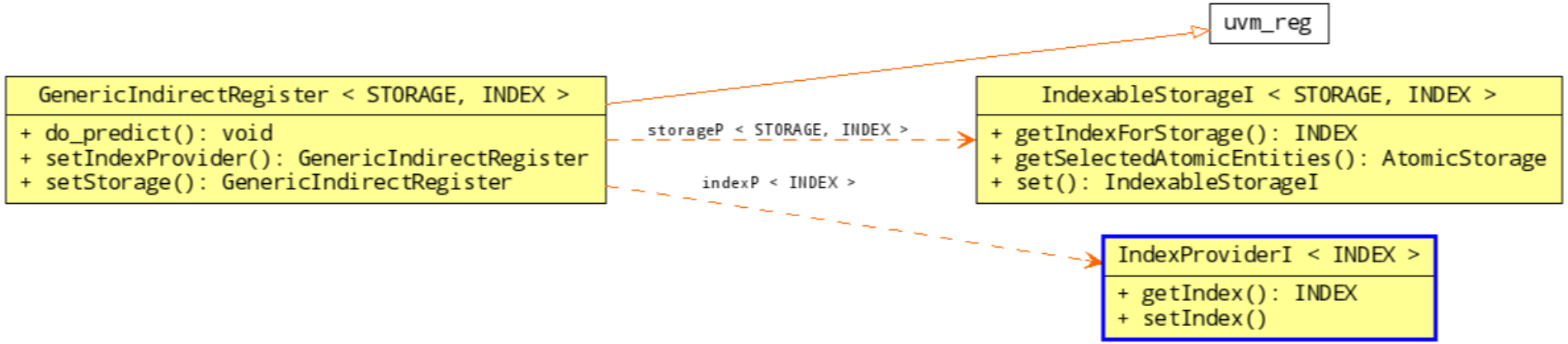
```
Data := applyOperation(getElements(getIndex()))
```



```
Data := applyOperation(
                storageP.getElements(
                        indexP.getIndex()
                ))
```

- indexP: object providing the index
- storageP: object performing selection of storage elements for a given index

Solution Base Classes



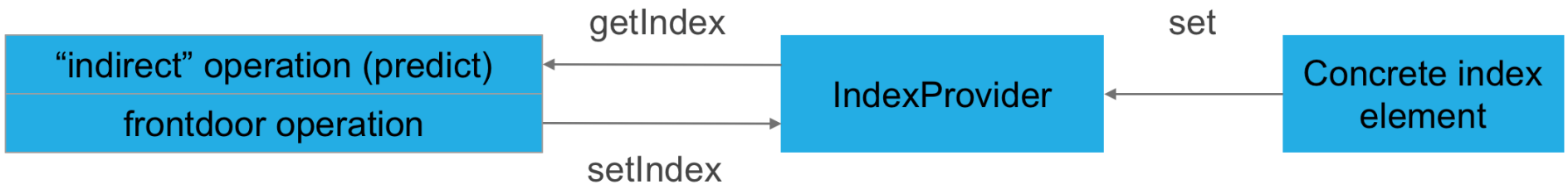
- Expects that the user provides at least an implementation of IndexProviderI and IndexableStorageI
- Expects that the user configures and connects the register and index/storage providers

Generic indirect register (=data register)

```
// indirect register with STORAGE=(Array of uvm_reg) and
// INDEX=(int unsigned)
class GenericIndirectRegister#(type STORAGE=uvm_reg,
                               INDEX=int unsigned)
      extends uvm_reg;

virtual function void do_predict (uvm_reg_item rw,
            uvm_predict_e kind = UVM_PREDICT_DIRECT,
            uvm_reg_byte_en_t be = -1);
begin
      INDEX idx = indexP.getIndex();
      STORAGE rg[] =
      storageP.getSelectedAtomicEntities(idx);
      foreach(rg[idx])
            rg[idx].do_predict(rw, kind, be);
      end
endfunction
```

IndexProviderI Example




```
class URFindexProvider extends IndexProviderI#(int unsigned);
        // reference to uvm_reg_field holding the
        // actual numeric index
        local uvm_reg_field store;
…
        virtual function INDEX getIndex();
                return store.get_mirrored_value();
        endfunction
…
endclass
```

Storage Provider example

```
// example provider simply selects a single element via
// an unsigned index from array
// types+fields in base class:
// typedef STORAGE AtomicStorage[];
// protected AtomicStorage thisStore;

class MyIndexableStorageI extends
       IndexableStorageI#(uvm_reg,int unsigned);
...
       virtual function AtomicStorage
              getSelectedAtomicEntities(const ref INDEX idx);
              AtomicStorage t=new[1];
              t[0]=thisStore[idx];
              return t;
       endfunction
endclass
```

Frontdoor support

- Framework can be extended for automatic frontdoor support
- Reverse operations needed:
  - StorageProviderI needs support to get an Index for a Set of Storage elements
  - IndexProviderI needs support to set an Index
- Then a generic frontdoor can map a direct access into an indirect bus access

Summary

- Limited ‘indirect’ register access support in current UVM
- A generic framework to handle arbitrary indirect registers on top of UVMREG has been presented
- user can provide own types and mechanisms for storage, index and selection
- Code is available http://forums.accellera.org/files/file/125-flexible-indirect-registers-with-uvmzip/

Questions