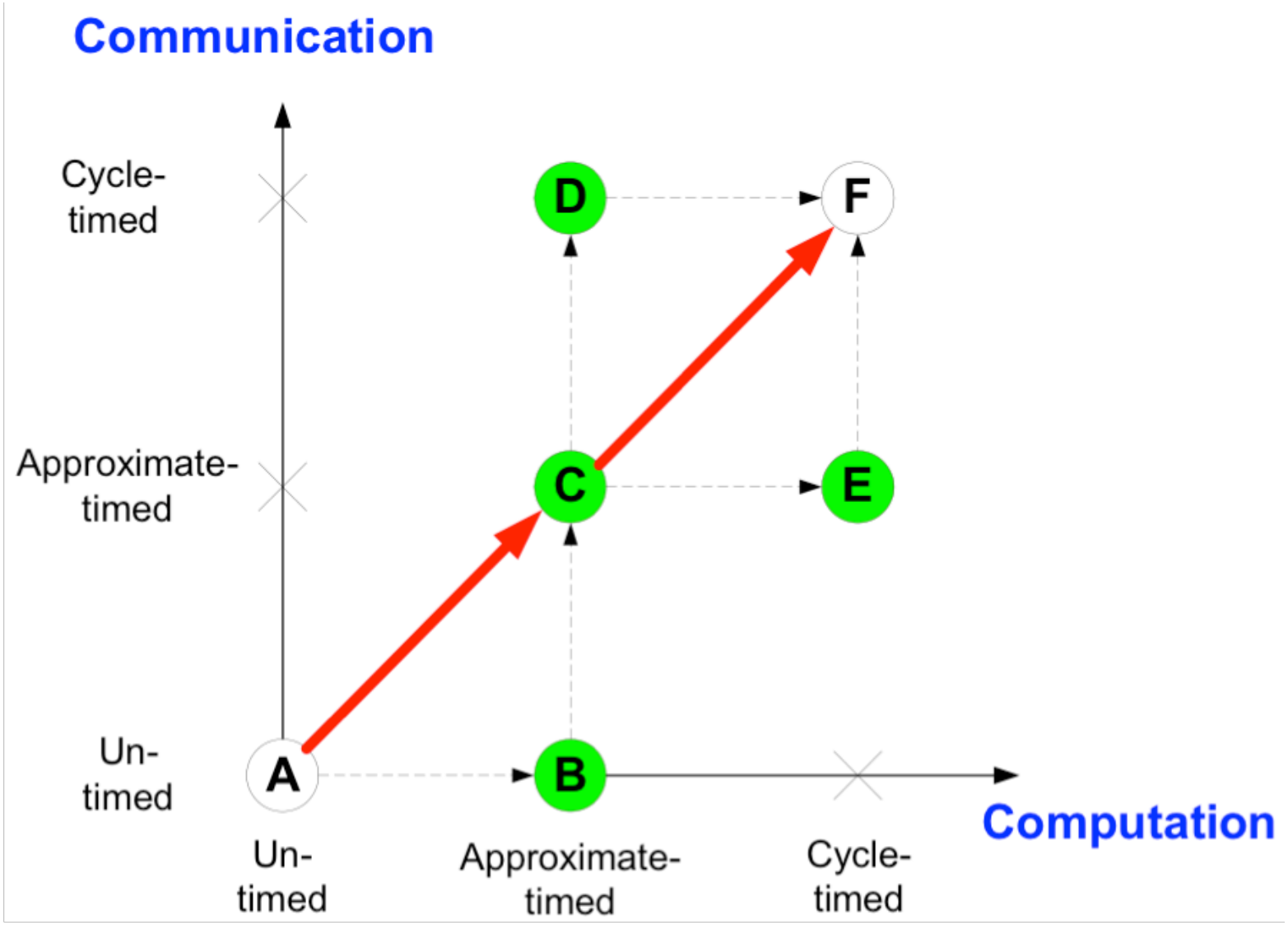
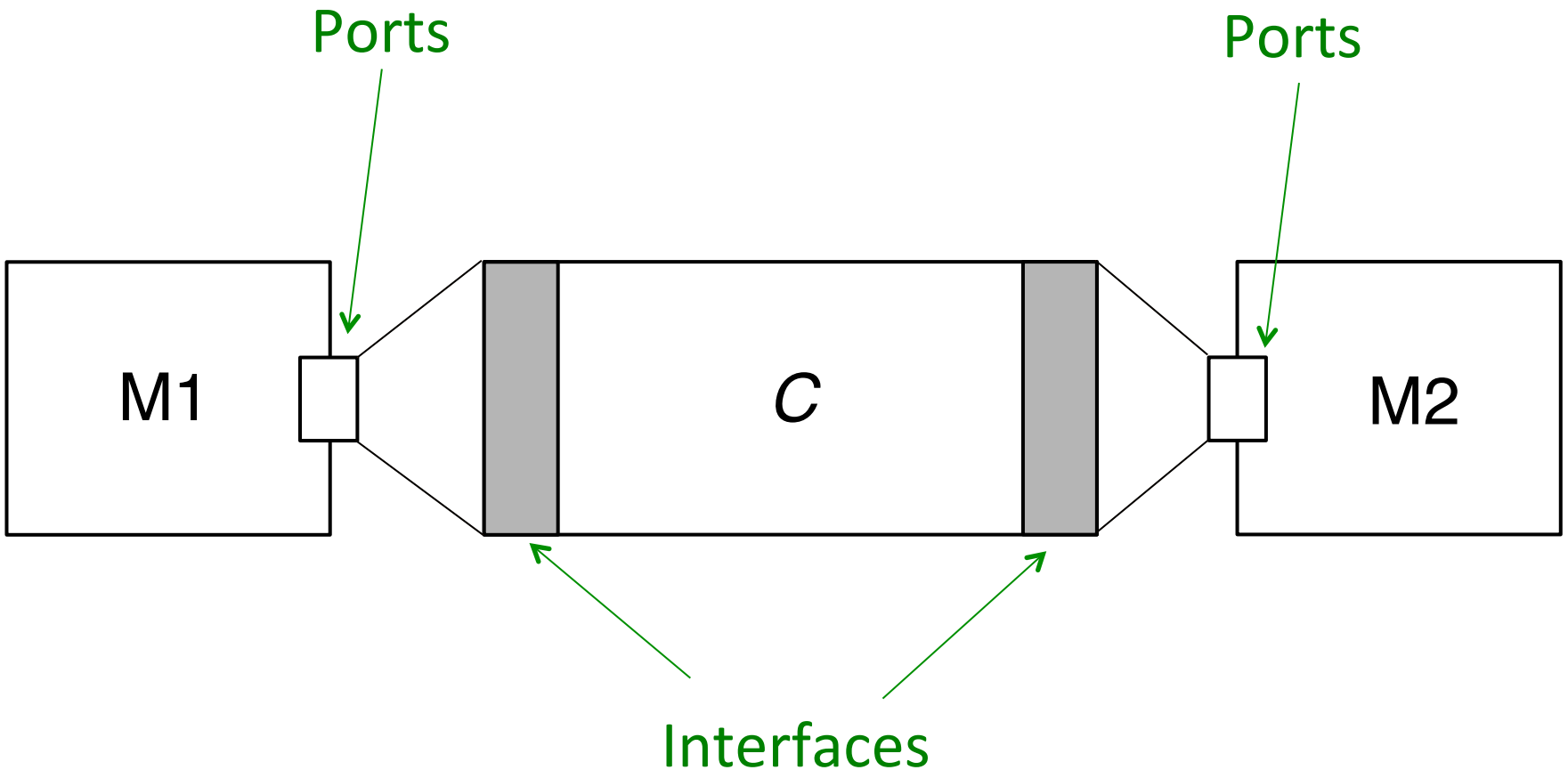
System-on-Chip Design
Refinement

Dr. Hao Zheng
Comp. Sci & Eng.
U of South Florida

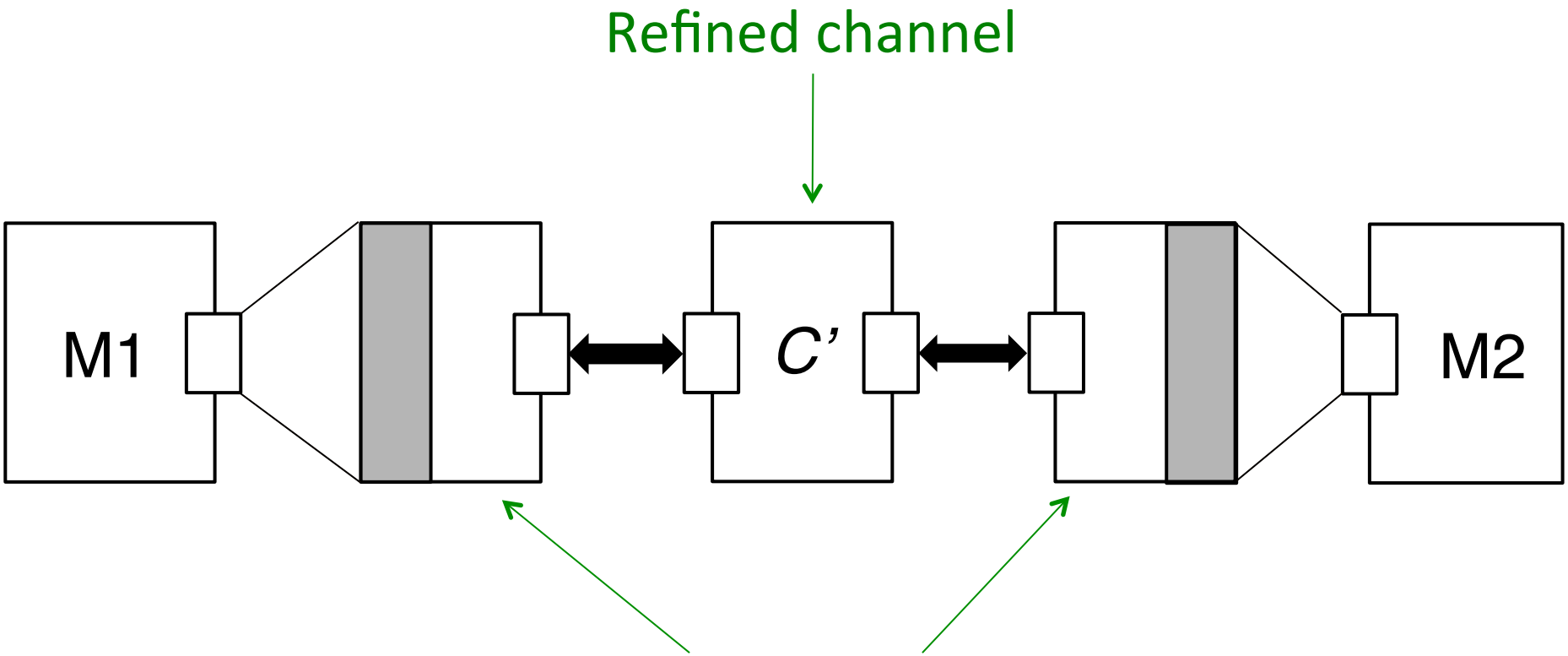
Levels of Abstraction



Refinement: General Idea

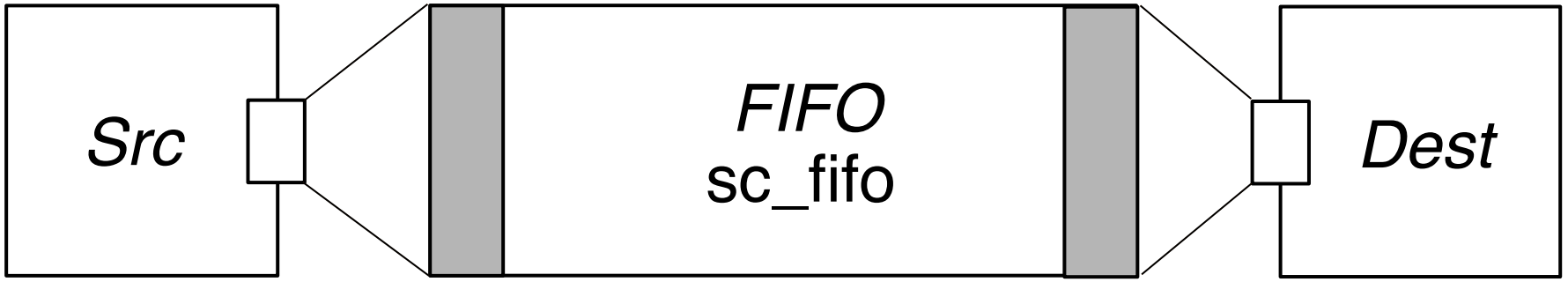


Refinement: General Idea

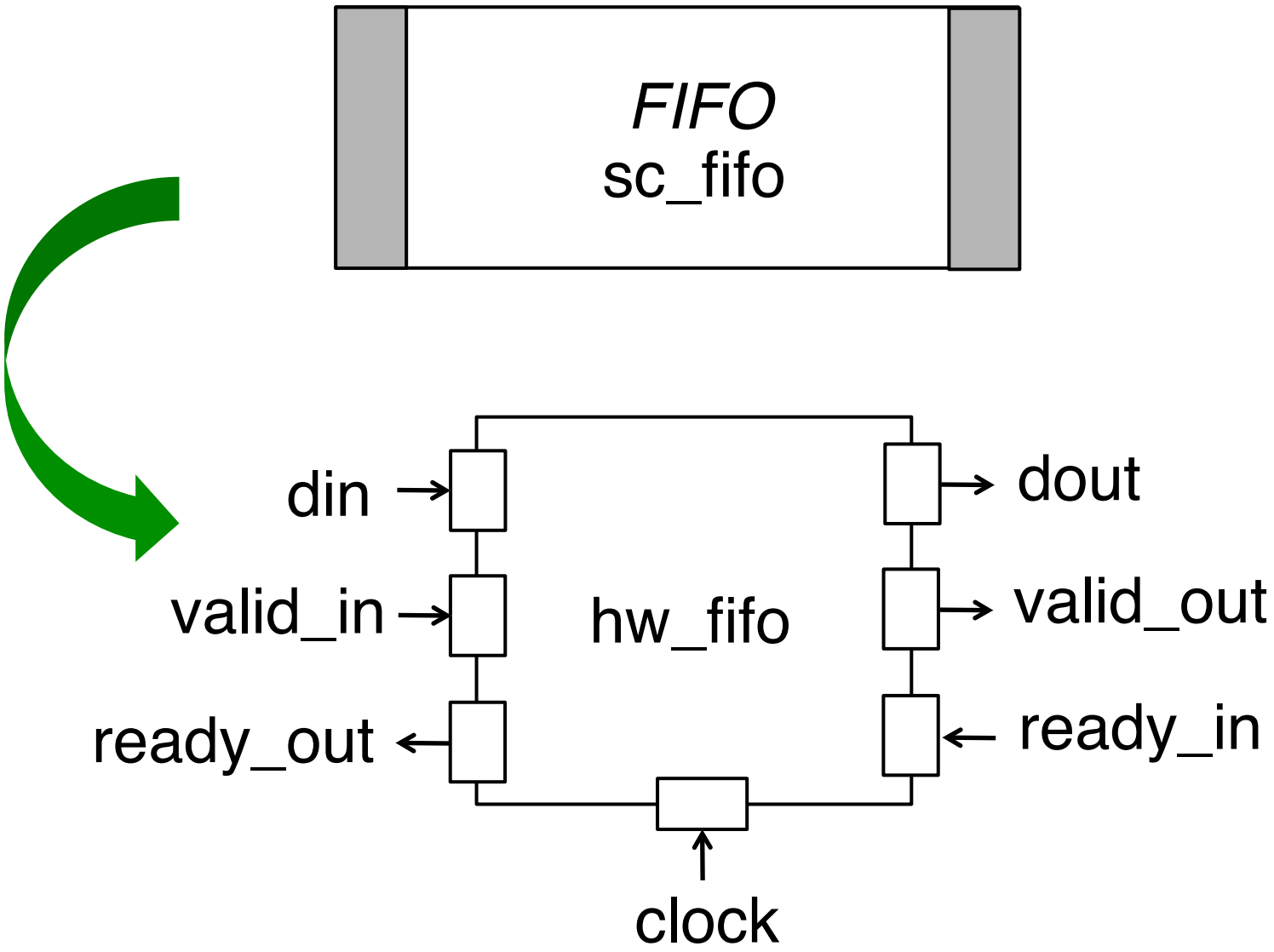


adapters:
translate interface methods to operations on ports

Refinement: Example



Refinement: HW FIFO



Refinement: HW FIFO

```
template<T> class hw_fifo : public sc_module
{
public:
   sc_in<bool> clock;

   sc_in<T>  din;
   sc_in<T>  valid_in;
   sc_out<T> ready_out;

   sc_out<T> dout;
   sc_out<T> valid_out;
   sc_in<T>  ready_in;

   ... // constructor and process
protected:
   unsigned _size, _first, _items;
   T* _data;
}
```

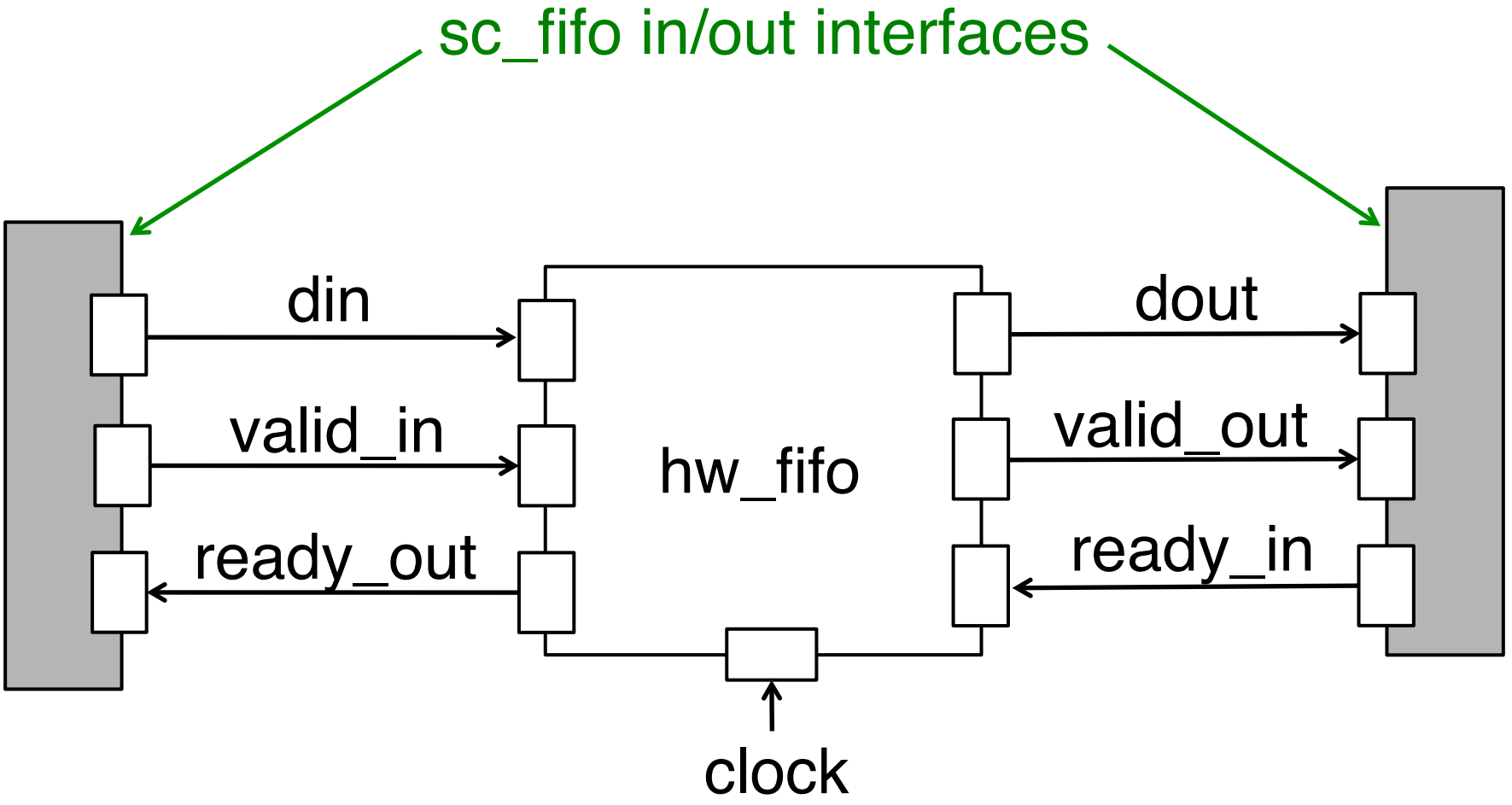
Refinement: HW FIFO

```
template<class T> class hw_fifo : public sc_module
{
public:
   // constructor and process
   SC_HAS_PROCESS(hw_fifo);
   hw_fifo(sc_module_name name, unsigned size)
      : sc_module(name), _size(size) {
     assert(size > 0);
     _first = _items = 0;
     _data = new T[_size];
     SC_METHOD(main);  sensitive << clock.pos();
     ready_out.initialize(true);
     valid_out.initialize(false);
   }

   ~hw_fifo() { delete _data; }
}
```

Refinement: HW FIFO

```
void hw_fifo::main()
{
   // write operation
   if (valid_in.read() && ready_out.read()) {
      _data[(_first + _items) % _size] = din;
      _items++;
   }
   // read operation
   if (ready_in.read() && valid_out.read()) {
      _items--;
      _first = (_first + 1) % _size;
   }

   ready_out = (_items < _size);
   valid_out = (_items > 0);
   dout      = _data[_first];
}
```

Refinement: HW FIFO Wrapper



Refinement: HW FIFO Wrapper

```
template<class T> class hw_fifo_wrapper :
      public sc_module,
      public sc_fifo_in_if<T>,
      public sc_fifo_out_if<T>  {
public:
   sc_in<bool> clock;

protected:
   sc_signal<T> wr_data;
   sc_signal<T> wr_valid;
   sc_signal<T> wr_ready;

   sc_signal<T> rd_data;
   sc_signal<T> rd_valid;
   sc_signal<T> rd_ready;

   hw_fifo<T> hw_fifo_inst; // hw fifo instance
}
```

Refinement: HW FIFO Wrapper

```
hw_fifo_wrapper::hw_fifo_wrapper(
              sc_module_name name, unsigned size)
{
   hw_fifo_inst(name, size);

   // port binding for hw_fifo_inst
   hw_fifo_inst.clock(clock);
   hw_fifo_inst.din(wr_data);
   hw_fifo_inst.valid_in(wr_valid);
   hw_fifo_inst.ready_out(wr_ready);

   hw_fifo_inst.dout(rd_data);
   hw_fifo_inst.valid_out(rd_valid);
   hw_fifo_inst.ready_in(rd_ready);
}
```

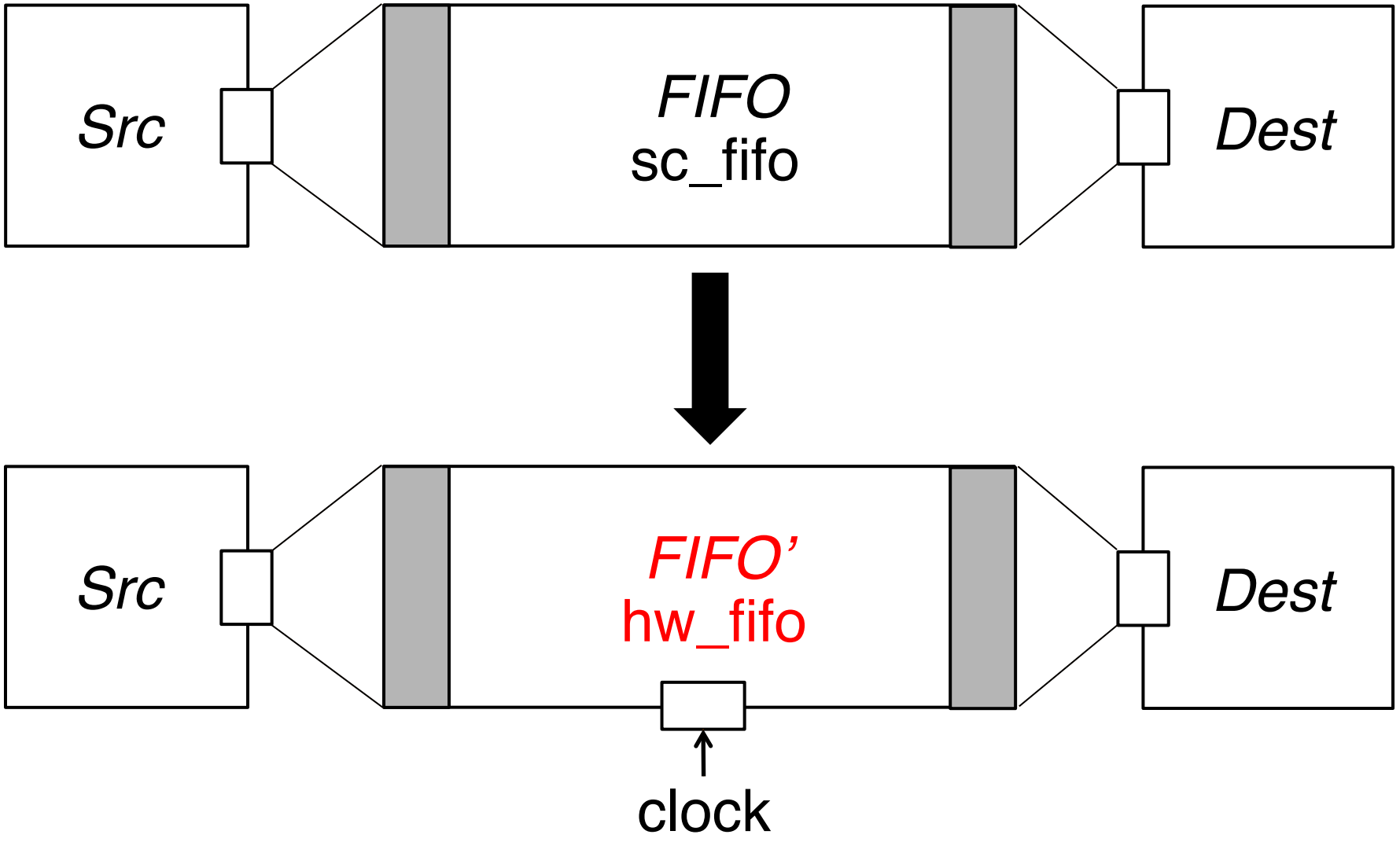
Refinement: HW FIFO Wrapper

```
virtual void hw_fifo_wrapper::write(const T& data)
{
   wr_data = data;
   wr_valid = true;

   while (wr_ready != true) {
      wait (clock->posedge_event());
   }
   wr_valid = false;
}
```

Refinement: HW FIFO Wrapper

```
virtual T hw_fifo_wrapper::read()
{
   rd_ready = true;

   while (rd_valid != true) {
      wait (clock->posedge_event());
   }
   rd_ready = false;
   return rd_data.read();
}

virtual void hw_fifo_wrapper::read(T& d)
{
   d = read();
}
```

Refinement: Example



Reading Guide

- ***SystemC-1*** book: section 9.1 – 9.3.