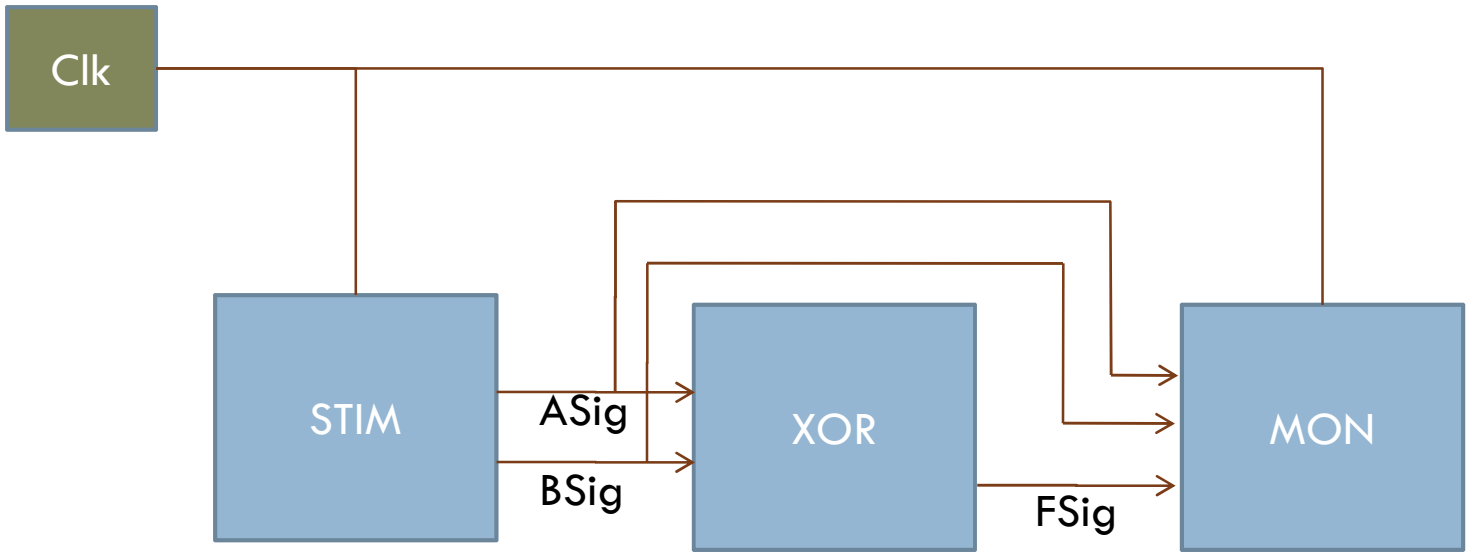
SYSTEMC

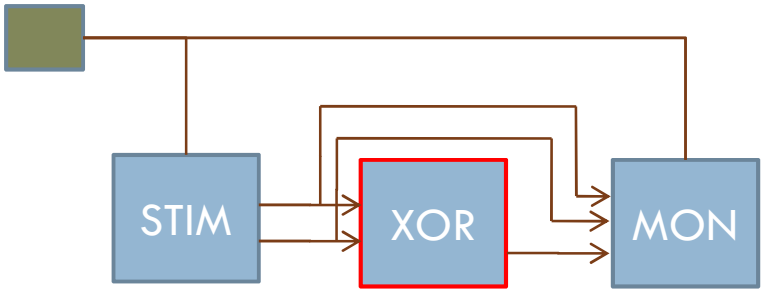
Cpr E 588

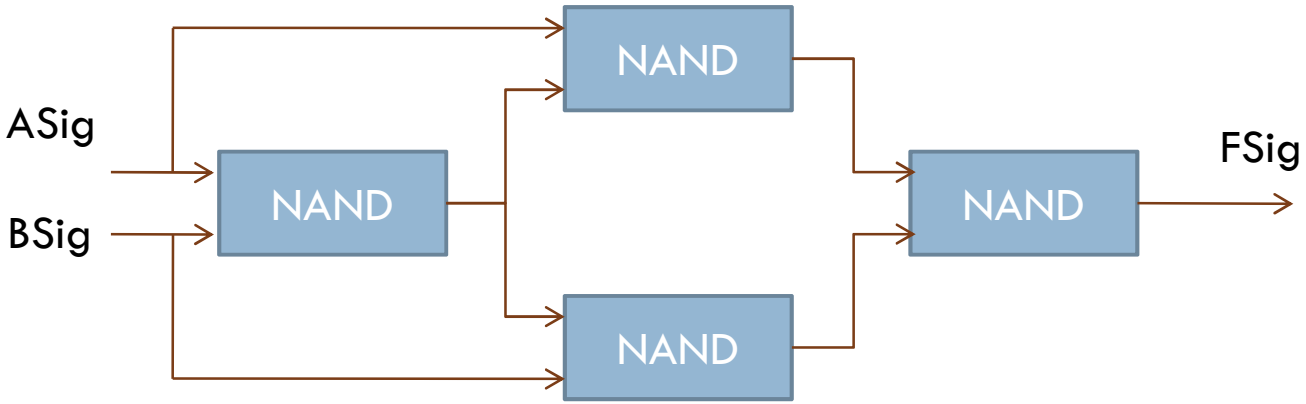
RTL Model



- DUT Simulation
  - Simulation Clock, Stimulus, Monitoring
- Hierarchy
- sc_signal: evaluate – update.

XOR

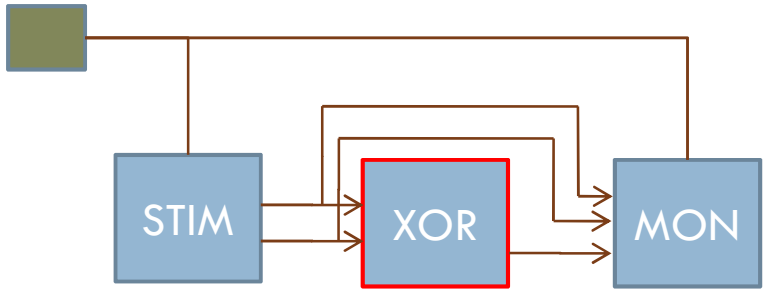




*Constructor without a process.

NAND

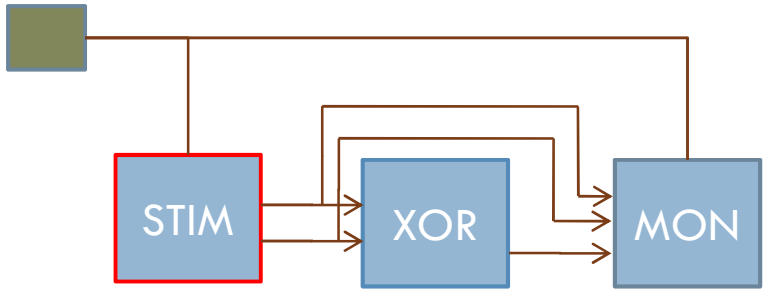
```
SC_MODULE(nand2)
{
  sc_in<bool> A, B;
  sc_out<bool> F;

  void do_nand2() {
    F.write( !(A.read() && B.read()) );
  }

  SC_CTOR(nand2) {
    SC_METHOD(do_nand2);
    sensitive << A << B;
  }
};
```



- sc_in: specialized port for sc_signals.
  - `sc_port<sc_signal_in_if<bool > >`
- sc_signal:
  - evaluate/update channel, like a signal in VHDL or reg in Verilog
  - The value do not changes (see example)
- Static 'sensitivity': defined at registration, the only sensitivity allowed for SC_METHOD

STIM

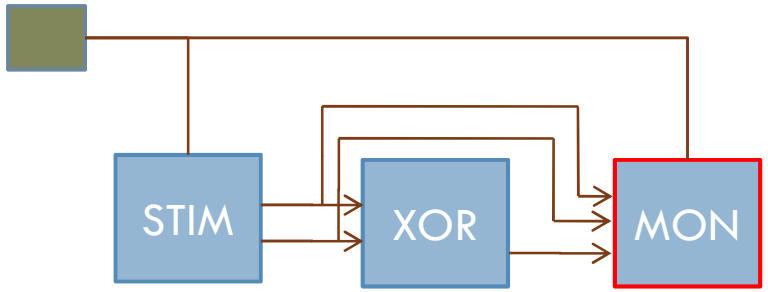
```
SC_MODULE(stim)
{
  sc_out<bool> A, B;
  sc_in<bool> clk;

  void StimGen() {
    A.write(false);
    B.write(false);
    wait();
    A.write(false);
    B.write(true);
    wait();
    A.write(true);
    B.write(false);
    wait();
    A.write(true);
    B.write(true);
    wait();

    sc_stop();
  }

  SC_CTOR(stim) {
    SC_THREAD(StimGen);
    sensitive << clk.pos();
  }
};
```



- Clk: special kind of signal.
  - `sc_clock TestClk("TestClock", 10, SC_NS, 0.5, 1, SC_NS);`
- Clock drives the stimulus, static sensitivity.
- Stimulus drives the rest of the simulation.
- Clocked SC_THREAD

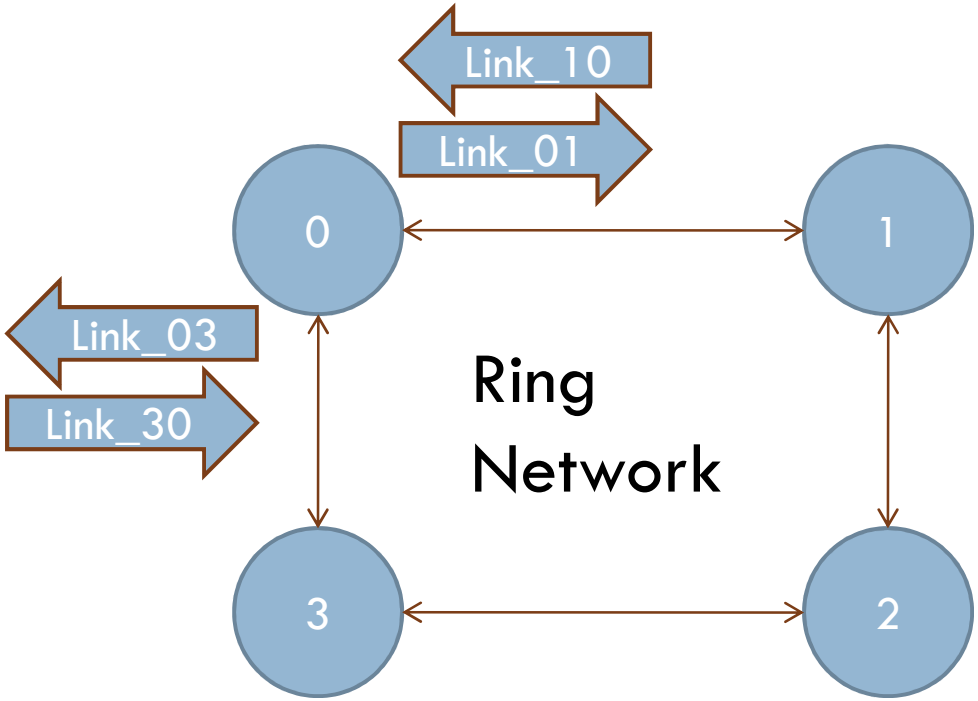
STIM

```
SC_MODULE(mon)
{

  sc_in<bool> A, B, F;
  sc_in<bool> Clk;

  void monitor() {
    cout << setw (10) << "Time";
    cout << setw (2)  << "A";
    cout << setw (2)  << "B";
    cout << setw (2)  << "F" << endl;

    while(true) {
          cout << setw (10) <<
sc_time_stamp();
             cout << setw (2) << A.read();
             cout << setw (2) << B.read();
             cout << setw (2) << F.read()
<< endl;
             wait();
    }
  }

  SC_CTOR(mon) {
    SC_THREAD(monitor);
    sensitive << Clk.pos();
  }
};
```



- Clocked SC_THREAD
- sc_time_stamp: current simulation time.

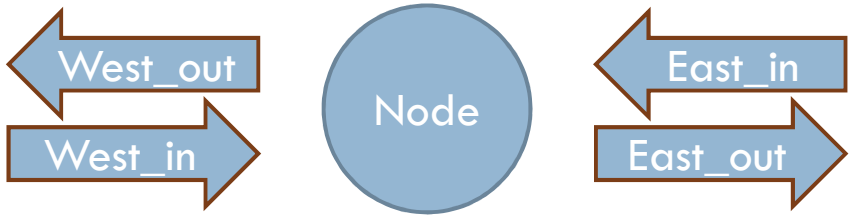
FIFO Communication (TLM)



*Two FIFOs per link

Node Model

- Port
  - `sc_fifo_out<packet> east_out;`
  - Custom data 'packet'
- FIFO Channel
  - `sc_fifo<packet> Link01("Link_01", 1);`
  - Depth of 1, blocking communication
- Dynamic sensitivity
  - No sensitivity
  - `west_out.write(pkt);`
  - `wait( east_in.data_written_event());`
- See behavior…




```
struct packet {
  char src;
  char dst;
  char data[2];
  packet& operator= (const packet& rhs)
  { … };
  bool operator== (const packet& rhs)
    const { return ( … ) };

  void Print(void)
  { … };
};
ostream& operator<< (ostream& os,
                     const packet& trans);
```