Metropolis

Metropolis Project Team
University of California Berkeley
Cadence Berkeley Laboratories
Metropolis Framework

Design of Function Processes

Design of Communication Media

Design of Architecture Components

**Metropolis Infrastructure**
- Model of computation
- Design methodology
  - Abstraction levels
  - Refinement
- Base tools
  - Design imports
  - User interface
  - Simulation

**Metropolis Point Tools:**
- Synthesis/Refinement
- Analysis/Verification
Metropolis: Model of Computation

- **System function**: a network of processes
  - process: sequential function + **ports**

- **Do not commit to particular communication semantics**
  - **ports**: interconnected by **communication media**
  - **communication media**: define communication semantics
    - e.g. queues, shared memory, …, generic, ...

- **Do not commit to particular firing rules of processes**
  - a special construct to define interaction between processes and media
Communication

- **Communication medium:**
  - **state:** snapshot of the medium
  - **interfaces:** read, write, status-check, ...
  - **properties:** # of writers, transaction, arbitration, ...

- An interface may be supported by more than one media.
- Interface functions at different abstraction levels to support refinement.

- Language to define communication media
- Library of pre-defined media
interface reader {
    void read(data, rate);
    int num();  // # of elements
}

interface writer {
    void write(data, rate);
    int num();
}

medium bfifo reader writer {  // bounded FIFO
    int num;  // # of elements
    int depth;  // the depth of the fifo
    ...

    int num() {
        return num;
    }
    void read(data, rate) {
        ...
    }
    void write(data, rate) {
        ...
    }
}
Process

- **Ports:**
  - Each port is specified with an interface it can access to. All and only the functions of the interface can be used through the port.

- **Sequential program:**
  - Interaction with communication media
    ```
    await(cond){ st1; st2; ... stk;}
    “if cond is TRUE, then atomically execute {st1; ... stk;}.”
    • Atomic operations
    • Micro steps
  - Non determinism
  - Bounded loops
  - Parameters
interface reader {
    void read(data, rate);
    int num();
}

interface writer {
    void write(data, rate);
    int num();
}

process filter {
    reader port1;
    writer port2;

    await(port1.num() > 7) {
        port1.read(V, 8);
    }

    bounded_loop(i, 0, 4, 1) { // for(i=0; i<4; i=i+1)
        V[i] = V[7 - i];
    }

    ...
}
interface reader {
    void read(data, rate);
    int num();
}

interface writer {
    void write(data, rate);
    int num();
}

process filter {
    reader port1, port3;
    writer port2;

    c = 1;
    await (port1.num() > 7 || port3.num() > 0) {
        if (port3.num() > 0) port1.read(c, 1);
        if (port1.num() > 7) port1.read(V, 8);
    }

    bounded_loop(i, 0, 4, 1) {
        V[i] = c * V[7 - i];
    }

    ...
}
Network of Processes

- Define the structure of a network
  - Instantiate processes: set parameters
  - Instantiate communication media: set parameters
  - Specify connections

- Specify constraints on the network
  - Scheduling constraints

A network may be hierarchical; a process may be a subnet of processes.
application my_design {
    process F = new filter();
    medium Fifo1 = new bfifo(8, int);  // bfifo(depth, type)
    medium Fifo3 = new bfifo(1, int);

    connect(F.port1, Fifo1);
    connect(F.port3, Fifo3);

    process P = new producer();
    process C = new controller();

    ...}
Metropolis Framework

Metropolis Infrastructure
- Model of computation
- **Design methodology**
  - Abstraction levels
  - Refinement
- Base tools
  - Design imports
  - User interface
  - Simulation

Design of Function Processes
Design of Communication Media
Design of Architecture Components

Metropolis Point Tools:
- Synthesis/Refinement
- Analysis/Verification
Design Methodology

- Functional Decomposition
- Behavior Adaptation
- Communication Media Insertion
  MoC Wrapping
- Communication Refinement
  Channel Adaptation
- Mapping and Optimizations
Functional Decomposition

- **Functional Decomposition**
  - at the highest abstraction level, a system is a single process
  - it is refined into a set of concurrent processes

- **Process:**
  - relation between an input domain and an output co-domain
  - only behavior, no communication
  - denotational specification
Functional Decomposition (ex.)

MPEG Decoder

VLD \rightarrow IDCT \rightarrow MC \rightarrow DISPLAY
Behavior Adaptation

- Behavior adapters
  - match different domains, so that processes can understand each other
  - relation between two domains
  - not part of original system specification: needed because of the particular decomposition
  - needed independently of how the communication is performed
Communication and MoC

- Communication medium
  - each link needs a communication medium
  - does not affect or change the relation inside processes

- MoC wrapper
  - used to establish a firing rule and a communication semantics for each process
  - only the Moc wrapper is modified if a medium is changed
Communication and Moc (ex.)
**Refinement**

- any communication medium can be refined into an arbitrary netlist, as long as the interface is not changed

**Channel adapters**
- used to preserve properties of a given interface
- example:
  - lossless communication realized with a lossy medium (retransmission + acknowledge)
Optimization

- map each element (processes, adapters, media) onto architecture
- merge processes, adapters and media into a single process, when applicable
- provide an imperative description for each process
Mapping and Optimization (ex.)
Metropolis Framework

Design of Function Blocks

Design of Communication Media

Design of Architecture Components

Metropolis Infrastructure
- Model of computation
- Design methodology
  - Abstraction levels
  - Refinement
- Base tools
  - Design imports
  - User interface
  - Simulation

Metropolis: Synthesis/Refinement
- Compile-time scheduling of concurrency
- Communication-driven hardware synthesis
- Protocol interface generation

Metropolis: Analysis/Verification
- Static timing analysis of reactive systems
- Invariant analysis of sequential programs
- Refinement verification
- Three-valued simulation
- Delay estimation using object code