LOTOS Specification and Model Checking for a Cellular FIFO-type Memory (Overview)

Katsumi WASAKI
wasaki@cs.shinshu-u.ac.jp

December 22th, 2003
Overview

- Objective
- Motivation
- Basic/Full-LOTOS
- Model Checking
- A Cellular FIFO-type memory
- Property
- INRIA/VASY CADP toolbox
- Self-recovering capability
- FSMview
Objective

- Verification of the concurrent systems
- FIFO-type memory resource management
- Queuing controller based cellular automaton
- Using “Model Checking” technology
- Specification of the FIFO memory controller by using Full-LOTOS
- Generating to LTS (Labelled Transition System) formatted BCG (Binary Coded Graph) file
- Reducing the state spaces (strong equivalence, branching equivalence, observational equivalence)
- Check the important properties for the self-recovering property of cellular FIFO memory
Motivation (history of methodologies)

- Temporal-logic model checking algorithms by Clarke and Emerson (1980's)
- New symbolic representation for the state transition graphs as ordered binary decision diagrams (OBBDs) by McMillan (1987)
- Possible to verify more than $10^{20} \sim 10^{120}$ state spaces (1991)
- SMV (Symbolic Model Checking) tool developed by McMillan and Clarke @CMU (1993)
- VIS (Verification Interacting with Synthesis) tool developed @Berkeley (1996)
- CADP (Caesar/Aldebaran Development Package) toolbox developed by VASY @INRIA (1996)
Motivations (previous works)

- A Fault-tolerant FIFO-type memory entitled: “A Self-recovering FIFO Memory based on the concept of Cellular Automaton” with self-recovering capability
- A high speed/realtime CODEC system design based on Extended Petri-net model and its executer
- A formal verification for a part of circuit module for ALU unit of micro processor by using Mizar proof checker
- A formal verification for a bus arbiter process entitled: “A Formal Verification of 'FutureBUS' Computer Bus Arbiter based on CTL Model Checking” by using VIS tool (Verilog-HDL model)
Motivations (parallel computing)

- Shared-memory type parallel processing (SMP)
- Message-passing type parallel processing (PVM, MPI) for “coarse grained” problems
- Distributed-MIMD super computer (ex. Cray/Xp)
- Workstation/PC Cluster system (“poor” super computer)
- Interconnecting network(i.e. Cross-bar switch) vs. Giga-bit Ethernet switch
- Approaches to the communication overhead caused network latency
LOTOS vs. E-LOTOS

  -- A formal description technique based on the temporal ordering of observational behaviour

- ACTONE : abstract data typing for Basic-LOTOS
  -- Basic-LOTOS+data type = (Full-) LOTOS

  -- adding the facilities of modularity, built-in data type, timing constraints, exceptions, parallel over parameters, etc... however, there are few tools...
Model Checking

- Several important advantages for verification of circuits and protocols
  -- the procedure is completely “automatic”.
- The user provides:
  1) the high level representation of the model (state machine, process algebra, HDL, etc...)
  2) represent the property to be checked (modal temporal logic, ex. CTL, Action-CTL)
- Generate state spaces of FSM based BDDs, BCGs
- State space reducing
- Explore the state spaces to check that satisfies the property by using exhaustive or on-the-fly
Unwind from FSM to Binary Decision Diagram (BDD)
A Cellular FIFO-type memory

- FIFO-type memory resource management
- Queuing controller based cellular automaton
- Self-recovering and fault-tolerant capability
Queuing control scheme
Neighborhoods of $Cell(i)$
State machine for read operation of $Cell(i)$

Condition: $(\text{Read}, \text{Flag}_i, \text{Flag}_{i+1})$

State 0: Initialize
State 1: Wait Read Operation
State 2: Detect Read Operation and Not Open ‘GATE’
State 3: Detect Read Operation and Open ‘GATE’
  Clear $\text{Flag}_i$
Functional block diagram of memory
Properties of basic operation

- An input data which is queuing the memory formerly, it can be taken out later as output data
- Queuing order by “First-In, First-Out” absolutely
- When the queue has no data, Empty Flag (EF) signal is asserted
- When the queue has a series of data filled the memory up, Full Flag (FF) signal is asserted
- Write operation is failed, if FF is asserted
- Read operation is failed, if EF is asserted
Property of error recovering (proposed)

- When a bit error has occurred on the shift-register “Flag” temporality, it will be entirely recovered after a book of finite operations of writing, (exclusive) or reading respectively.

- Case (1) : \#k bit has accidentally turned to “1”

- Case (2) : \#k bit has accidentally turned to “0”
Error recovering process: Case(1)

(a) for reading  

(b) for writing
Error recovering process: Case (2)

(a) for reading

(b) for writing
Property of error recovering (generalized)

- From starting any bit errors has occurred on the shift-register “Flag”, it will be entirely recovered after a book of finite operations of writing or reading. (i.e., non-deterministic choice of write/read)

- Case (3): any bit have accidentally turned to “1” or “0” (there are $2^n$ patterns)

--- not by-hands out, by using model checking!
INRIA/VASY CADP toolbox

- CADP (Caesar/Aldebaran Development Package)
  - Caesar : LOTOS compiler, LTS generation
  - Aldebaran : State space reducing, model checking
- A Software Engineering Toolbox for Protocols and Distributed Systems
- OPEN/CAESAR framework
  - manipulating LTS state space by using BCG
  - OPEN/C library
- Represent the property by using “Evaluator” (on-the-fly Model Checker)
- Web  http://www.inrialpes.fr/vasy/cadp/
FSMview

- A fancy graphical visualization of LTS formatted BCG file
- A lot of examples for VLTS (Very Large Transition System) Benchmark suite
- The resulting visualization:
  -- succeeded in showing global symmetries in the state transition graphs
  -- as well as showing similarities between similar sections of different (but related) graphs
- Web  http://www.win.tue.nl/~fvham/fsm/vlts/