LASSO - IRO
Université de Montréal

Hierarchical Annotated Action Diagrams (HAAD): Interface-Oriented Specification and Verification Method

E. Cerny

with

K. Khordoc, A. Tarnauceanu, B. Berkane, P. Girodias

in cooperation with

Nortel Semiconductors (Ottawa)

Outline

- **Introduction**
- **Structure of HAAD**
- **Example**
- **Interface Timing Verification**
- **Problems with Verification Methods**
- **Causality and Compatibility**
- **Verification Algorithms**
- **Constraint Satisfaction and Relational Interval Arithmetic**
- **Interface Timing Verification with Constraint Logic Programming and RIA**
- **Conclusions**

Introduction

- VHDL models of real-time controllers complicated if input timing margins to be checked and output timing implemented
- Accurate models Intellectual Property blocks needed without revealing internal structure
- Often, specifications are partly timing diagrams (interface protocols, I/O timing), partly procedures (data processing, decisions), partly high-level schematics (structure), and partly natural language (link / explain the preceding forms).
- Proposed unified approach:

Hierarchical	**A**nnotated	**A**ction **D**iagrams
Composition - form more complex behaviors from simpler ones	Attachment of procedures to actions - describes what data processing and when	Like Timing diagrams - describes basic protocol fragments and timing constraints

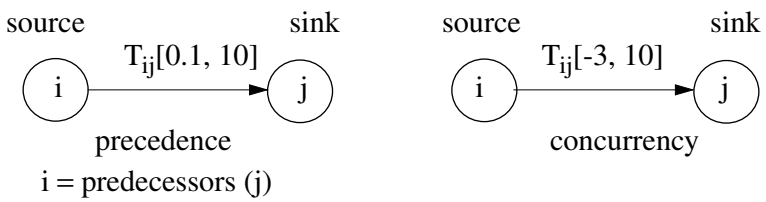
Use: Specification, documentation, behavioral VHDL model generation, static verification of realizability and compatibility of interface specs, model checking of non-annotated HAAD via Timed Automata

- Intermediate HAAD language, LISP-like syntax

Structure of HAAD - Leaf Action Diagrams (AD)

Basic behavioral templates over actions on ports and internal signals:

- Ports: VHDL port type and mode (in, out, inout), internal signals
- Waveform on a port: Sequence of actions related by timing relations
- Action: Some event (change of value or just update), direction in or out
- **Timing relation** (TR) between occurrence times of actions:
  - *Precedence*: $T_{min} \le t_j - t_i \le T_{max}, T_{min} > 0$ $(t_j - t_i \in [T_{min}, T_{max}])$
  - *Concurrency*: $T_{min} \le t_j - t_i \le T_{max}, T_{min} \le 0$ $(t_j - t_i \in [T_{min}, T_{max}])$



Structure of HAAD - Leaf AD

- **Timing Constraints** (TC): combinations of TRs having common sink action

  - *Conjunctive*: (any type of TR), simultaneously satisfied: Declarative definition

    $\forall \mathbf{i} \in$ **predecessors(j),** $\mathbf{max_i(t_i + T_{ij.min}) \le t_j \le min_i(t_i + T_{ij.max})}$

    $$t_3 \in t_1 + [1, 10] \cap t_2 + [1, 20]$$

  - *Latest* (precedence type TR): Causal dependence, e.g., 0-to-1 on inputs of AND gate

    $\forall \mathbf{i} \in$ **predecessors(j),** $\mathbf{max_i(t_i + T_{ij.min}) \le t_j \le max_i(t_i + T_{ij.max})}$

    $$t_3 \in max\,(t_1 + [1, 10],\ t_2 + [1, 20])$$

  - *Earliest*: (precedence type TR): Causal dependence, e.g., 1-to-0 on inputs of AND gate

    $\forall \mathbf{i} \in$ **predecessors(j),** $\mathbf{min_i(t_i + T_{ij.min}) \le t_j \le min_i(t_i + T_{ij.max})}$

    $$t_3 \in min\,(t_1 + [1, 10],\ t_2 + [1, 20])$$

- All TC satisfied simultaneously.

Structure of HAAD - Leaf AD

- **Intent of constraints**

Assume:	Assumed timing over input actions (environment)
	I -> I, O -> I constraints
Commit:	Timing of produced output actions
	I -> O, O -> O constraints
Requirements:	Timing to be implied by the behavior
	E.g., safety constraints on contention of shared bus

Restrictions for VHDL model generation:

- Only Latest and Earliest constraints in Commitments
- Each AD has a single Begin and a single End (virtual) action

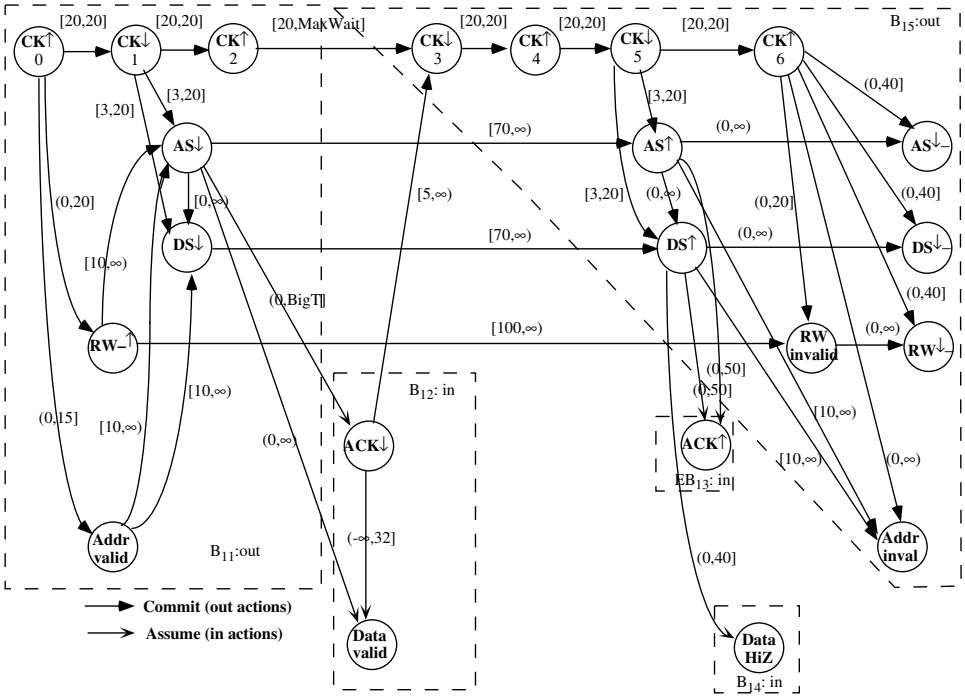
In development

- All types of constraints (causal partition over actions related by conjunctive constraints)
- Begin - End on each port - more flexible composition, pipelined behavior

Structure of HAAD - Leaf AD

MC 68360 processor Read cycle

see also pp14 & 16



Structure of HAAD - Hierarchical AD

- **Composition** of lower-level ADs inspired by Process Algebras
- **Synchronization** between ADs
  - Matching Begin - End actions
  - Single output - multiple input causal rendez-vous over common names

    Causal - output timing determined by source, not jointly
- **Operators**
  - Parallel
  - Sequential
  - Delayed deterministic choice
  - Loop
  - Exception
  - Top-level loop for commit delay exhaustive enumeration
- **Instantiation** and **port mapping**

Structure of HAAD - Annotations

- VHDL syntax
- Variable and constant declarations in AD
- AD Body contains
  - VHDL (state) variables, constants
  - Action variables attached to actions, store value when action occurs
  - These objects can be passed by reference to lower-level ADs
  - Scoping rules - local visibility only
- Boolean functions (predicates) and procedures attached to actions
  - Selection predicates on choice branches
  - Loop exit predicate on loop begin action
  - Loop count increment on loop end action
  - Action value validation predicate
  - Output action value assignment through action variable name
  - Output port assignment at action occurrence

VHDL Model Generation

- Compile from HAAD textual spec. to VHDL (IEEE STD 1076-1987)
  - Top-level HAAD corresponds to a VHDL entity
  - Package of predefined internal types and procedures
  - User defined types, procedures, and functions in a user package
- Instantiate and connect with other VHDL components (HAAD, gate, other...)

VHDL Process organization

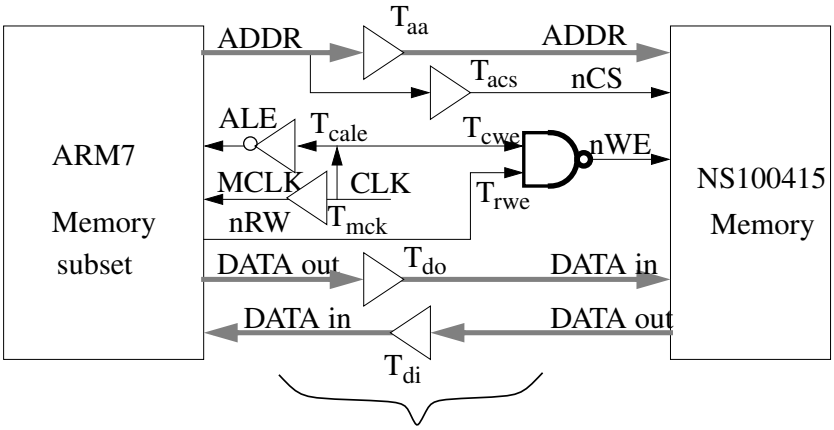
```
PROCESS BEGIN
  WHILE progress AND NOT error LOOP
    compute_time_out;
    WAIT ON all_signals FOR time_out;
    catch_possible_actions; --input and output when occurred
    process_outputs;         -- identify occurred output actions
    validate_port_activity; -- timing and value check
    action_var_side_effects;-- assign action variables
    call_action_pred;        -- exec. predicates
    call_action_proc;        -- exec. procedure
    generate_port_activity; -- assign outputs to ports
  END LOOP;
  Check termination conditions; -- done?
END PROCESS;
```

Example

(Sponsored by Cadence, Inc.)

Modeling and Interfacing NS100415 Static memory and ARM7 Processor

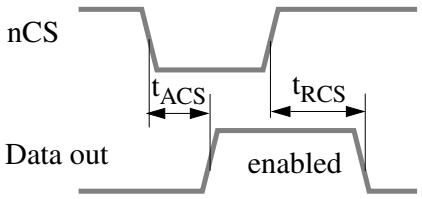
- Information in data sheets -> Interpretation and integration into AD templates
- Construction of HAAD models, VHDL generation
- Instantiation with interface glue logic

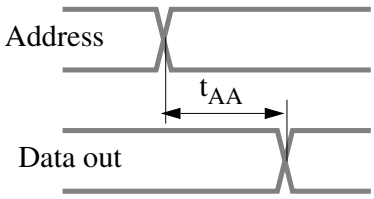


Gate-level VHDL model

Example

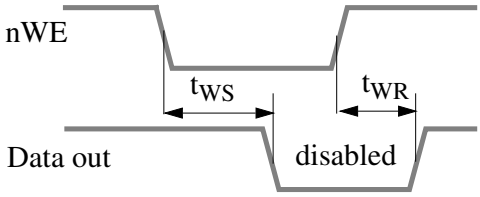
NS100415 Memory Data-sheet specification - Read Mode





t_{AA} = max 10 ns
t_{ACS} = max 5 ns
t_{RCS} = max 5 ns
t_{WS} = max 5 ns
t_{WR} = max 10 ns
Min values assumed 0^+

From Write Mode



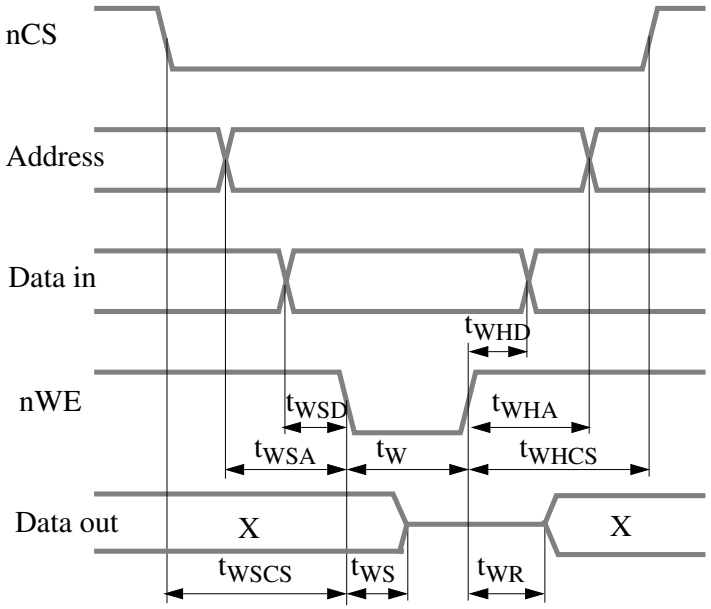
Acts like interrupt on read operation

Example

NS100415 Memory Data-sheet specification - Write Mode

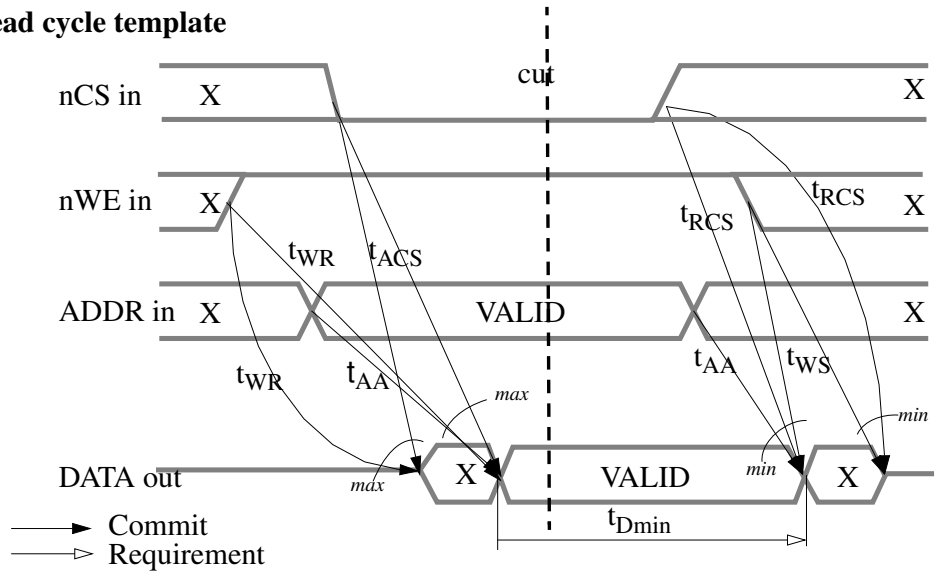
t_{WSCS} = min 1 ns
t_{WHCS} = min 2 ns
t_{WSA} = min 1 ns
t_{WHA} = min 2 ns
t_{WSD} = min 1 ns
t_{WHD} = min 2 ns
t_W = min 7ns
Assumed max is +∞

Affecting Read mode
t_{WS} = max 5 ns
t_{WR} = max 10 ns
Assumed min is 0 ns



Example

Read cycle template



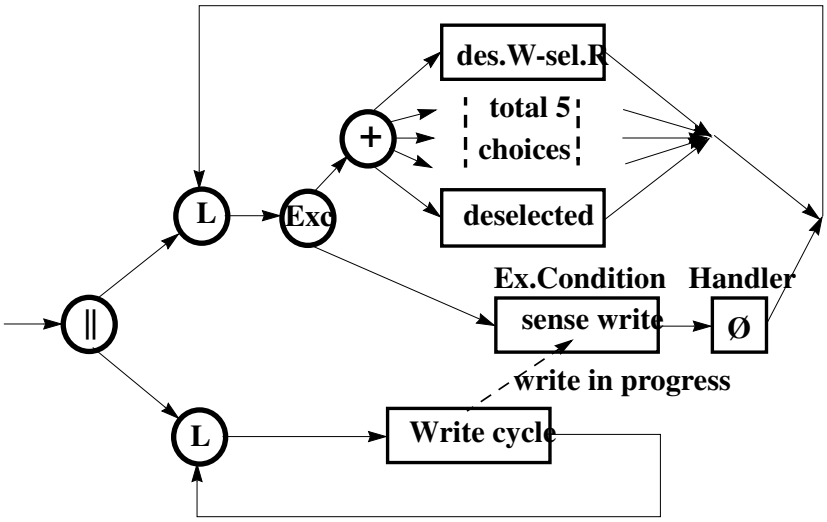
- Constraints with the same label are 100% correlated
- Interval values of the constraints (in ns):

t_{WR} = (0, 10] t_{WS} = (0, 5] t_{AA} = (0, 10]
t_{ACS} = (0, 5] t_{RCS} = (0, 5] t_{Dmin} = (0, _]

Example

- Interactions between read and write cycles
- Cycles when not selected
- Reduce number of templates - Address (and CS) and Data updates in each cycle even if value does not change (transactions)
- Write can interrupt a read

Memory HAAD hierarchy

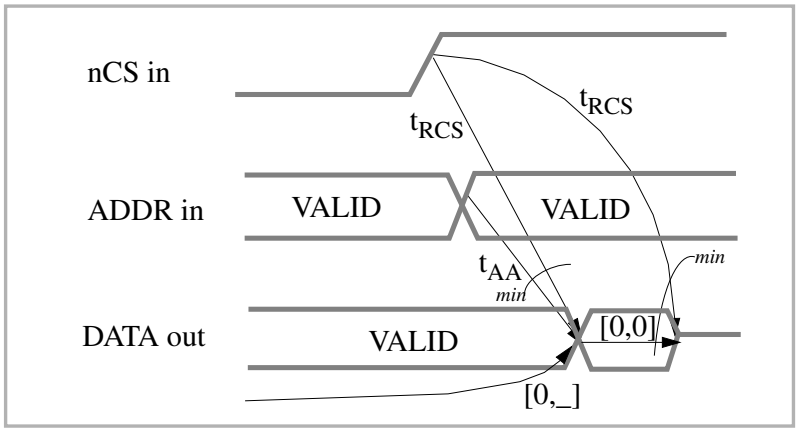


Annotations:

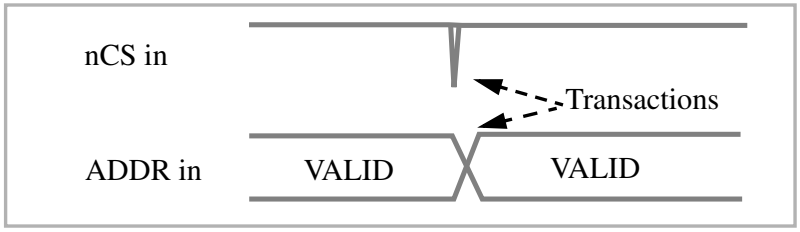
- Memory array, Write and Read Procedures into/from this array
- Automatic action variables for DataIn, DataOut, and Address

Example

A leaf AD: Selected R - Deselected R or W



A leaf AD: Deselected - Deselected



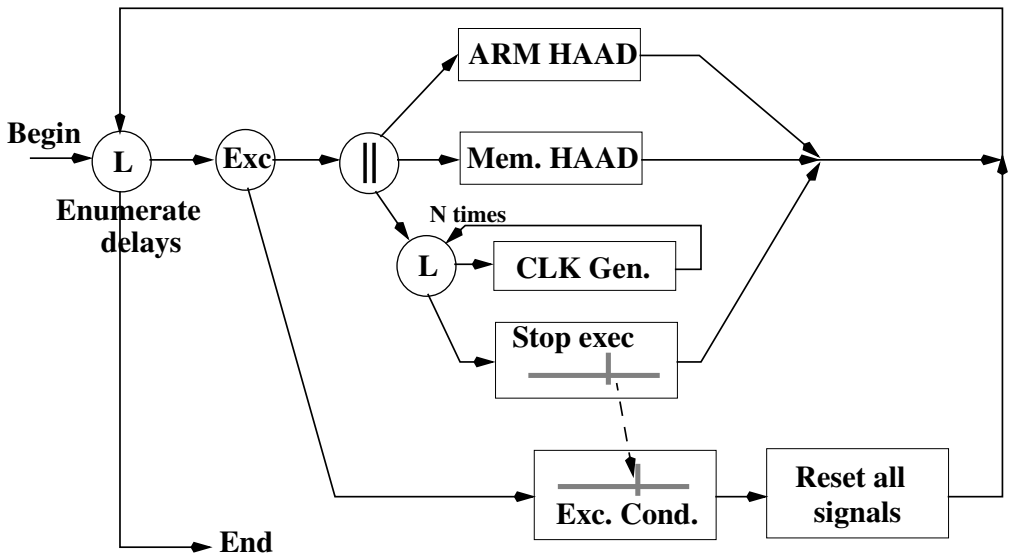
Example

HAAD encoding of a leaf AD:

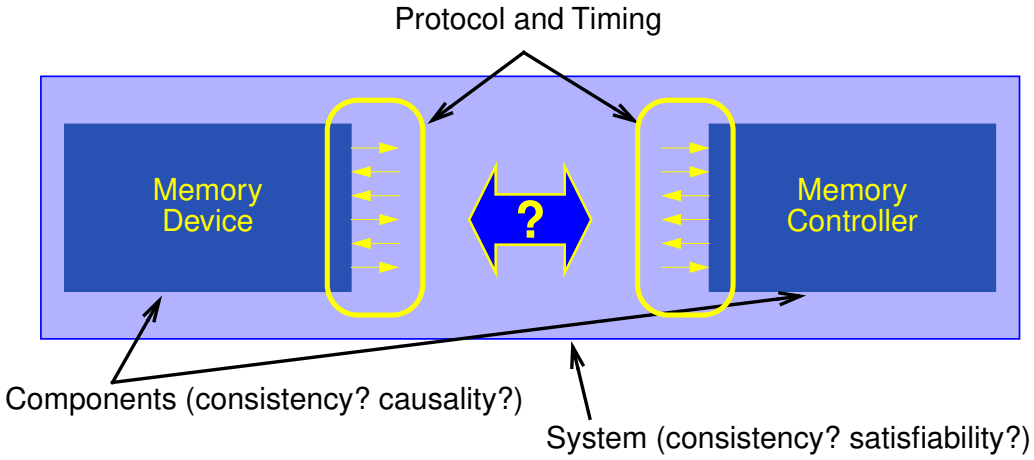
```
(DEFBEHAVIOR WR_non_select
 (PORTS    (PORT nCS IN "std_logic" MESSAGE)
  .... )
 (PARAMS   (PARAM InAddr INOUT "address_data_bus")
  .... )
 (LEAF
  (CARRIER-SPEC nCS
   (INITIAL-SPEC (VALID)) (ACTION-SPEC 'e1  (CONSTANT "'0'")))
  ....
  (CARRIER-SPEC DATAout
   (INITIAL-SPEC (CONSTANT "\"ZZZZZZZZZZZZZZZ\""))
   (ACTION-SPEC 'e5  (VALID initialize))
   (ACTION-SPEC 'e6  (VALID OutData)
    (PROCEDURE-CALL "read_data_from_memory"
                              InAddr OutData read_memory)))
  (LATEST
   (PRECEDENCE 'e1 'e5 (CNAME 'Twr1)
                       (CMIN 2) (CMAX 5) (INTENT COMMIT))
   (PRECEDENCE 'e2 'e5 (CNAME 'Tacs1)
                       (CMIN 5) (CMAX 10) (INTENT COMMIT)) )
  ....
  )
```


Example

Combined ARM - Memory HAAD model



Interface Timing Verification



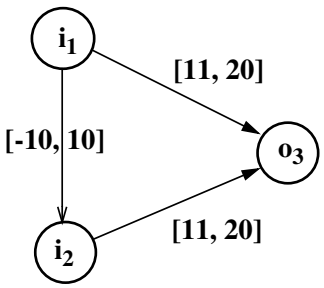
- **Consistency**: specification free of contradictions
- **Satisfiability**: specification makes valid assumptions
- **Causality**: the future does not control the past

Interface Timing Verification

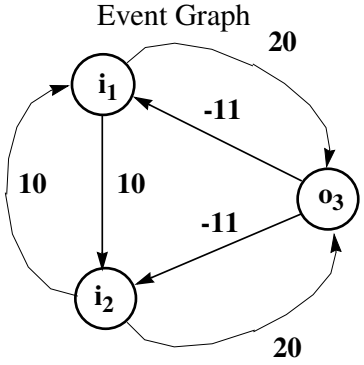
Problems with Verification Methods

- ***Consistency***
  - Occurrence time can be assigned to all actions without violating any constraint
  - Conjunctive constraints only - no negative cycles in Event Graph

AD



Event Graph



Conjunction of normalized one-sided constraints

$t_3 - t_1 \leq 20$

$t_1 - t_3 \leq -11$

$t_3 - t_2 \leq 20$

$t_2 - t_3 \leq -11$

$t_2 - t_1 \leq 10$

$t_1 - t_2 \leq 10$

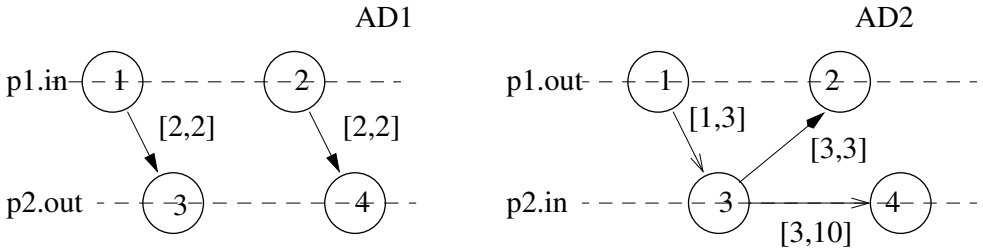
- All-pairs shortest distance algorithm (Jacobi: $O(n^4)$; Warshall-Floyd: $O(n^3)$)

Interface Timing Verification

Problems with Verification Methods

- ***Compatibility*** (Timing satisfiability in [BrGaMa91])
  - Verify that the timing produced by one AD satisfies the assumptions and requirements made on it by the other AD's [BrGaMa91]
  - Max. action separations produced by assume constraints of one device AD must be at least as great as max. separations of the same actions produced by the commit constraints of the other device AD

Problem: a false negative

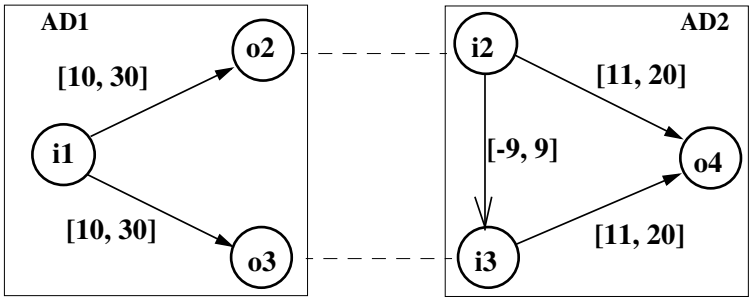


Must **COMPOSE commits** of all AD's ...

Interface Timing Verification

Problems with Verification Methods

But simple composition may not work:



- Compose commits of AD1 and AD2, check assumptions by resulting action separations computed: $t_2 - t_3 \in [-9,9]$

  assumption: [-9, 9] includes this interval -> SUCCESS

  false positive!!
- Device 1 implemented independently from Device 2
- Separation between o1 and o2 produced by Device 1 could be anywhere in [-20, 20]

Causality

- Consistency and simple composition / verification of assume constraints not sufficient!
- Must verify *causality* in the presence of linear conjunctive constraints
  - declarative nature of constraints
- Symmetry: assumptions (model of environment) and commitment (model of device)
- The decision that an action should occur at time **t** in a device as determined by the commit constraints must not depend on the occurrence times of actions performed by the environment at time > **t**, and vice-versa. It may depend, however, on all past actions
- Latest and earliest constraints by definition causal (max and min operations)
  - occurrence time of future actions cannot affect past occurrence times

Causality

A specification has a **causal** interpretation if there exists a **causal partition of actions**:

- Each block only contains actions of the same direction
- Local constraints of a block are all constraints having the sink action in the block
- Occurrence times of actions determined from
  - Occurrence time of Triggers - actions located in other block that are source of local constraints
  - Local constraints (non-deterministic selection of occurrence time)
  - Execution of a block occurs when all its trigger actions (in other blocks) occur (generalized Petri-net transition)
- Occurrence time of block actions must be in the future of its triggers (causal)

(See example of block structure on page 7)

Causality

Block Machine: semantics of execution

- Time t increases monotonically from 0
- A_i has a time-stamp variable $t(A_i)$, initially $t(A_i) = \infty$
- A_i *occurs* when $t = t(A_i)$
- B *enabled* when $t = \max(t(Trig_1), ..., t(Trig_k))$
- $F_j(T_{TrigBj})$ computes set of possible occ. times given the trigger times
- Execution of Block Machine:

```
t(A0) := 0; t := 0⁻;              -- Origin action A0
while ∃i, t(Ai) > t do            -- ∃ unoccurred action?
    wait until an Ak occurs;      -- Action occurrence may trigger
    ∀j, Bj is enabled do          -- a block?
        if Fj(TTrigBj) = Ø then deadlock-- no solution?
            else
                Choose TBj ∈ Fj(TTrigBj) -- occ. time assignment
end if; end do; end while;
```

- Block Machine is ***live*** if it does not ***deadlock***

Causality

Sufficient Conditions for causal AD specification: Causal Partition

- C1: In and out actions do not share a block of P
- C2: All trigger actions of B are effectively in the past of all actions in B
- C3: The corresponding Block Machine is live

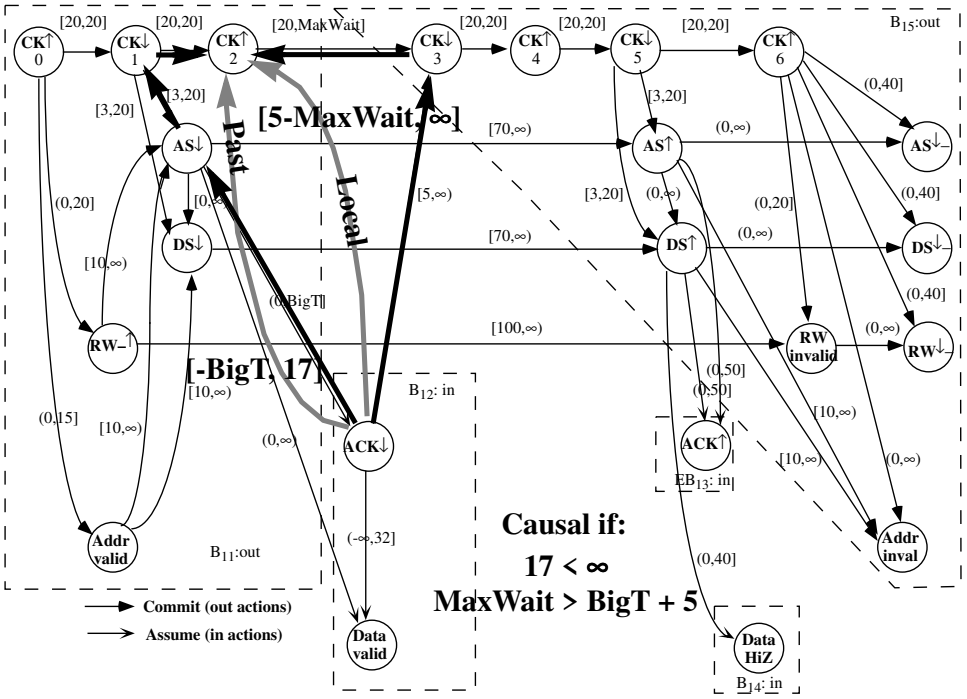
  Verify that separations of triggers of B computed over complete constraint graph are strictly tighter than those computed using the local constraints of B.

Properties of causal ADs

- In (assume constraints) and Out (commit constraints) sub-partitions are independent
- All causal partitions are trace equivalent
- Inconsistency of AD is a special case of non-causality

Causality

Example - verifying condition C3



Causality

Inclusion of latest / earliest constraints

- Latest / Earliest constraints are by nature causal
  - precedence from source actions
  - for any time of sources there is an occurrence time assignment to the sink action.
- Sink action is target only of one combination of
  conjunctive **or** earliest **or** latest
- Actions that are sinks of earliest / latest constraints form each a Block
- Causality conditions checked only on blocks with conjunctive constraints

Compatibility

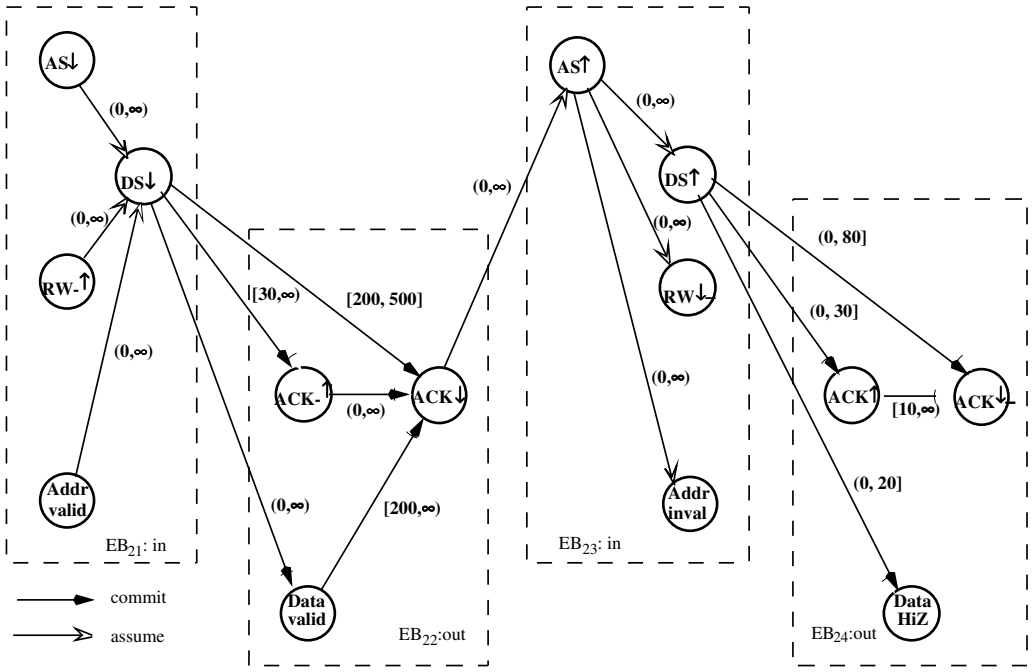
- Check consistency and assumptions on composition of all commit constraints

Properties:

- Compatibility independent of choice of causal partition (block machine) for each AD
- Compatibility check does not require the knowledge of the Causal Partitions

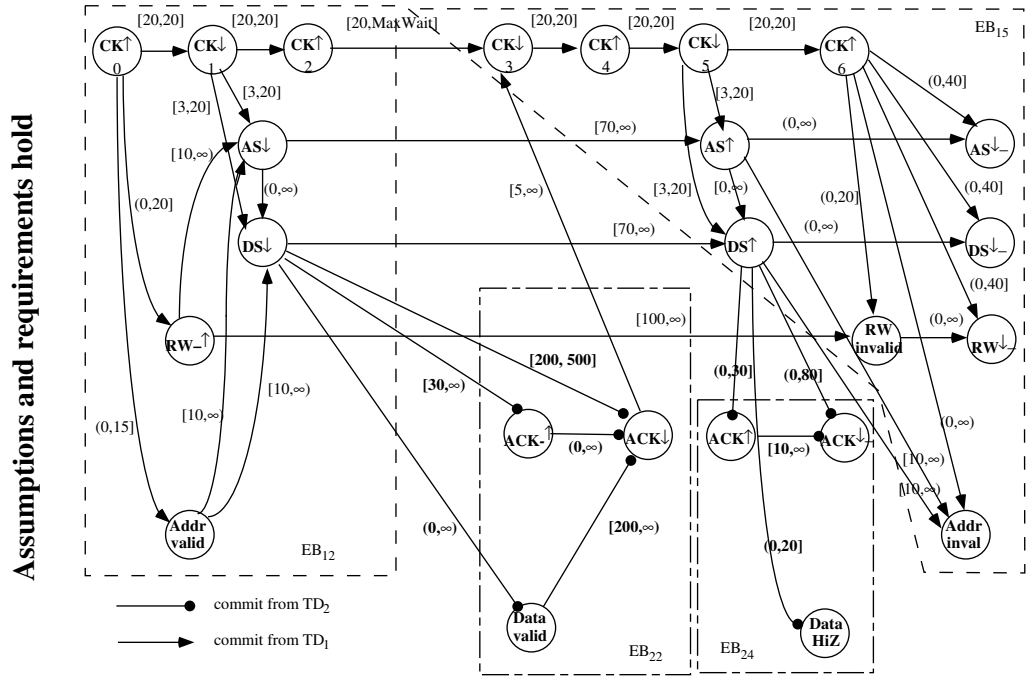
Compatibility

Example: MC68360 vs I/O interface (I/O interface Read cycle)



Compatibility

Example: MC68360 vs I/O Interface (composed commit constraints)



Verification Algorithms

- Interface verification can be reduced to
  - computing maximal event separations or

- Existing algorithms: Different constraint combinations need different algorithms, difficult to add other constraints

Constraint Logic Programming and Relational Interval Arithmetic

-> unified approach

- CLP (BNR) Prolog, distributed by ALS Inc. (Cambridge, MA)

Benefits

- Natural representation of the problem (constraints & intervals)
- Constraint satisfaction (check for consistency, max. distance calculation)
- User interface simplified and memory management and search techniques built in
- Other constraints can be included (delay correlation, conditional delays)
- Programs easy to modify and to extend

Constraint Satisfaction

Variables + Domains + Constraints = CSP → similar to ITV decision problems (events & timing relations)

- A **solution** to a CSP is an assignment of values to all variables, which satisfies all the constraints

- A CSP is **globally consistent** if any assignment of values that respects the set of domains is a solution

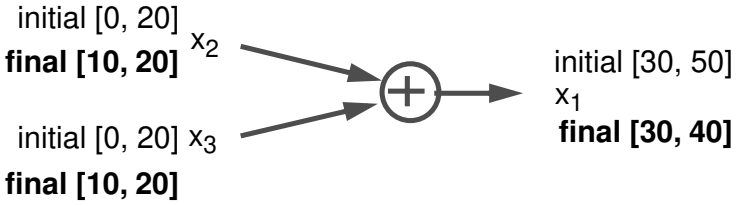
- Two CSPs *P* and *P´* are **equivalent** if they have the same set of solutions

- **Solving** a CSP *P* is the process of finding a globally consistent equivalent CSP P´ ($\mathcal{NP}$-complete problem)

- **Consistency techniques** → partial consistency...

Constraint Satisfaction - Relational Interval Arithmetic

- RIA as a consistency technique: **local propagation** of intervals
- Functional arithmetic vs. relational arithmetic
  - Expressions evaluated once vs. expressions repeatedly updated
  - Unidirectional flow of data vs. bi-directional flow of data



- "Inputs" to "outputs": tighter bounds on x_2 and x_3 tighten bound on x_1
- "Outputs" to "inputs": tighter bound on x_1 may tighten bounds on x_2 and x_3
- Bounds updated iteratively until a stable set is found - greatest fixpoint of system of equivalent interval equations

Constraint Satisfaction - Relational Interval Arithmetic

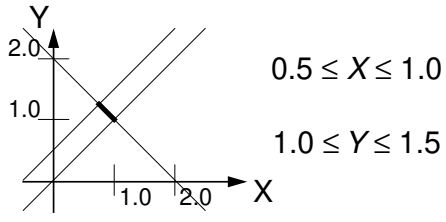
Problems:

- The interval values returned by RIA may be larger than the true intervals that characterize the exact solutions
- Example: *X*, *Y* are positive integers

$$X + Y = 2$$

$$Y \leq X + 1$$

$$Y \geq X,\ X \geq 0,\ Y \geq 0$$



$0.5 \leq X \leq 1.0$

$1.0 \leq Y \leq 1.5$

- Interval propagation $\rightarrow$ X = [0,2], Y = [0,2]!

Not globally consistent

Constraint Satisfaction - Relational Interval Arithmetic

Equivalent symbolic outputs - global consistency

- Systems of linear constraints

  CLP & RIA ≡ All-pairs shortest distance

- Systems of max-only constraints

  CLP & RIA ≡ McMillan & Dill max-only algorithm

- Systems of linear constraints and *max* constraints

  CLP & RIA ≡ McMillan & Dill linear and max algorithm

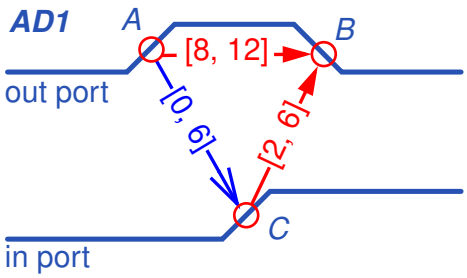
Systems of linear, *max* and *min* constraints? Delay correlation?

-> State verification as consistency question

-> If consistent, refine by case analysis within Prolog

Interface Timing Verification with CLP RIA

Event separation problem using Constraint Logic Programming with RIA




```
ad1(Ta, Tb, Tc) :-

  Dab : real(8, 12),
  Dac : real(0, 6),
  Dcb : real(2, 6),

  { Tb == Ta + Dab },
  { Tb == Tc + Dcb },
  { Tc == Ta + Dac }.
```

- Initially, $t_A = t_B = t_C = [-\infty, \infty]$
- **Reference point** *k* needed (*k* = A or *k* = B or *k* = C): `{ Ta == 0 }`
- After, $t_i = [-d'_{ki}, d'_{ik}]$
- Problem: $\boldsymbol{d'_{ik} = d_{ik} = maximize(t_k - t_i)}$ **or** $\boldsymbol{d'_{ik} > d_{ik}}$**?** (global consistency?)

Interface Timing Verification with CLP RIA

Constraint Logic Programming with RIA - Dealing with Partial Consistency

(1) Reduce problem to systems of linear and *max* constraints only

- **Implicit** decomposition of *min* constraints (using intervals of Booleans)

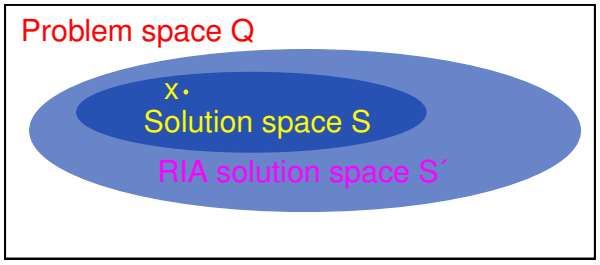
$$t_{d2} = min(t_{A2} + \delta_{A2d2}, t_{R2} + \delta_{R2d2})$$

$$B_1 = (t_{A2} + \delta_{A2d2} \leq t_{R2} + \delta_{R2d2}),\ B_2 = (t_{R2} + \delta_{R2d2} < t_{A2} + \delta_{A2d2})$$

$$t_{d2} = (t_{A2} + \delta_{A2d2}) * B_1 + (t_{R2} + \delta_{R2d2}) * B_2,\ B_1 = (1 - B_2)$$

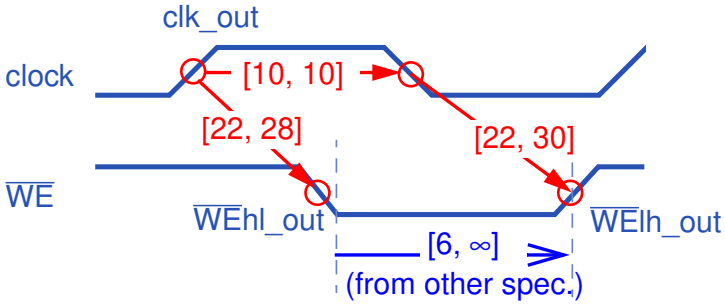
(2) Reason on the complement of the solution space

- Non-causality ⇒ inconsistency
- False negatives instead of false positives…



Interface Timing Verification with CLP RIA

Delay Correlation



- Without delay correlation:
  - $t_{WElh_out} - t_{WEhl_out} \in [4, 18]$
  - possible timing violation
- With delay correlation (50%):
  - $t_{WElh_out} - t_{WEhl_out} \in [8, 14]$
  - no timing violation possible

Interface Timing Verification with CLP RIA

Delay Correlation - Basic model:

- $\delta_i \in [l_i, u_i] \rightarrow \delta_i = l_i + \Delta_i \cdot z_i,\ z_i \in [0, 1]$
- $\forall i, j, |z_i - z_j| \leq 1 - \gamma$ (the degree of correlation is γ)

Examples:

- $\gamma = 1 \rightarrow \forall i, j, z_i - z_j = 0$ (100% correlated)
- $\gamma = 0 \rightarrow \forall i, j, |z_i - z_j| \in [0, 1]$ (no correlation)

Simpler formulation for RIA (**one** variable)

- $\boldsymbol{p \in [0, \gamma],\ \ x_i \in [0, (1 - \gamma)] \ \ \rightarrow \ \ z_i = p + x_i}$
- $\boldsymbol{\delta_i = l_i + \Delta_i \cdot (p + x_i)}$

Interface Timing Verification with CLP RIA

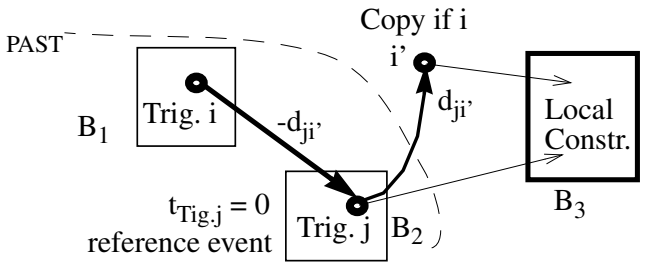
Delay Correlation - Example: 50% correlation

```
p(Tclk, TWEhl, TWElh, Gamma):-
     DCWElh: real(22, 30),
     DCK:    real(10, 10),
     DCWEhl: real(22, 28),
  { TWEhl == TCKlh + DCWEhl },
  { TCKhl == TCKlh + DCK },
  { TWElh == TCKhl + DCWElh},
  % delay correlation information
  Epsilon is (1.0 - Gamma),
  P: real(0.0, Gamma),
  XWEhl: real(0.0, Epsilon),
  XWElh: real(0.0, Epsilon),
  { DCWEhl == 22 + (6 * (P + XWEhl)) },
  { DCWElh == 22 + (8 * (P + XWElh)) }.

?-p(Tclk, TWEhl, TWElh, 0.5), { Tclk == 0 },
  { TWElh - TWEhl < 6 }.
NO
```

Interface Timing Verification with CLP RIA

- **Consistency** - Checked implicitly in causality check
- **Causality** - Verify by checking of consistency of a derived constraint system, one for each block and pairs of triggers, in order of predecessor blocks:
  if consistent for all pairs -> block is causal,
  else potentially non-causal, refine by interval splitting to desired accuracy



- **Compatibility** - Compose commit constraints, include complement of assumptions and check for consistency
- If consistent result obtained when global consistency not guaranteed - case analysis on booleans and interval splitting refines the potentially **false negative** answer

Interface Timing Verification with CLP RIA

Experimental Results: Memory device + memory controller

- Components consistent and causal
- Global system consistent
- Satisfiability depends on the degree of correlation:
  - 0% degree of correlation → 3 possible timing violations
  - 75% degree of correlation → no timing violations

	Nb. Ev.	Nb. Constr.	CPU times (ms)			
			Caus.	Cons.	Satisf.	Comp.
Mem. Device	15	22	635/1114†	619	N/A	N/A
Mem. Ctrl.	22	17	770/131871†	712	N/A	N/A
Dev. + Ctrl.	22	39	N/A	879	1192	2704

Conclusions

- Behavioral model hierarchy in HAAD
- Executable model generated in VHDL; Simulation with other VHDL models; Delay enumeration on Commit constraints for verifying interface implementations.
- Causality problem identified in AD specs with conjunctive constraints, sufficient conditions proposed
- Relational interval arithmetic is a local consistency technique that produces exact solutions for systems of linear, *min* and *max* constraints
- A powerful and flexible tool for static timing verification
- Practical cases easy (< 50 events and 100 constraints); No other “complete” method
- Efficient treatment of delay correlation
- Case analysis built in the Prolog system for refining approximate results
  - at worst false negative answers if incomplete case analysis used
- Lack of benchmarks ...
- Extensions in progress
  - Full HAAD static verification (mix of reachability analysis and constraint satisfaction)
  - Port-oriented composition (Begin - End)
  - RT and Gate-level timing verification