SPECIFICATION AND VERIFICATION OF DIGITAL SYSTEMS

F.K. Hanna and N. Daeche

Rapporteur: Mr. A.M. Koelmans

Specification and Verification of Digital Systems

Keith Hanna
University of Kent

These two talks relate to work carried out over the last few years by my colleague Neil Daeche and myself.

Specification of Digital Systems

This talk will describe how predicate logic may advantageously be used as a 'hardware description language' for specifying (that is, describing *some* but not necessarily all, characteristics of) waveforms, device behaviours (gates, flipflops, etc) and structures (or circuit diagrams). It will concentrate particularly on techniques for specification that are relevant to the lower levels of abstraction where the analogue characteristics of digital waveforms cannot be ignored, and at which many real problems (glitches, hazards, etc) tend to occur. In particular, it will propose the use of a structured, polymorphically sorted, higher-order predicate logic as a natural basis for a hardware description language.

Formal Verification of Digital Systems

The principles involved in formal verification of digital systems will be examined, and the kind of computational support (for allowing sound, user-guided inferencing) required will be outlined. The main part of the talk will be concerned with a case study involving the verification of an edge-triggered D-type flipflop. Although at first sight, it seems to be a relatively straightforward circuit, on closer examination the reason 'why' it works turns out to be unexpectedly complex. One interesting feature of the approach used to construct the proof is that the conditions (a complex set of inequalities involving the timing parameters of the various gates) under which the circuit actually correctly realises the required behaviour emerge as a by-product of undertaking the proof.

References

"Specification and Verification using Higher-Order Logic: A Case Study", published in "Formal Aspects of VLSI Design", Milne and Subrahmanyam (editors), North Holland, 1986.

"Purely Functional Implementation of a Logic", published in "Lecture Notes in Computer Science, Vol 230 (8th Intnl. Conf. on Automated Deduction, 1986)", Springer Verlag, 1986.

SPECIFICATION & VERIFICATION OF DIGITAL SYSTEMS

KEITH HANNA NEIL DAECHE

UNIVERSITY OF KENT

I.

- SPECIFYING DIGITAL WAVEFORMS
- SPECIFYING DIGITAL BEHAVIOURS
- SPECIFYING DIGITAL STRUCTURES
- CASE STUDY: D-FLIPFLOP SPEC.

II.

- HIGHER-ORDER LOGIC
  - SPECIFICATION & IMPLEMENTATION
- CASE STUDY: D-FLIPFLOP VERIFIC.

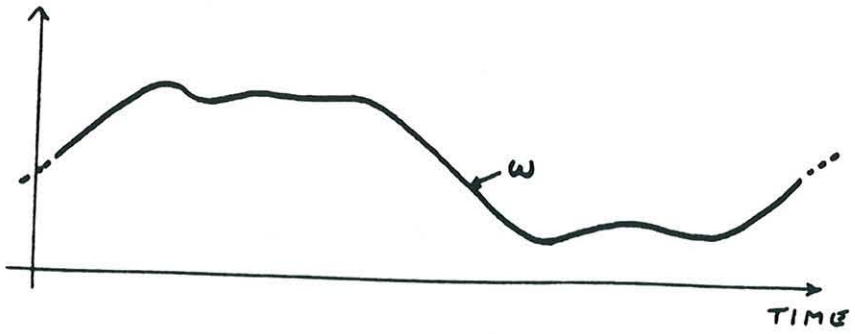
DESCRIBING DIGITAL SYSTEMS

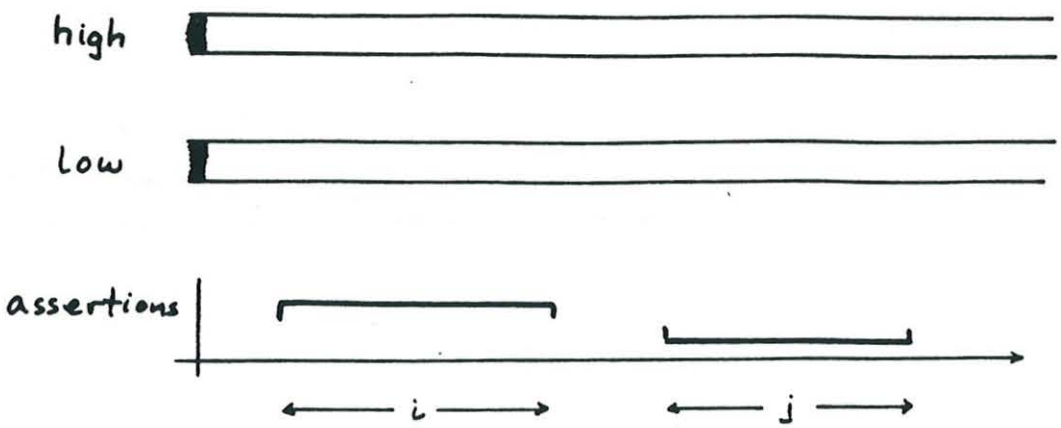
- MANY (HIGHLY OPTIMISED) NOTATIONS ("HDL's") DEVELOPED FOR DESCRIBING/SIMULATING DIG. SYSTEMS
- BUT NOT FOR REASONING ABOUT THEM !
- FEATURES FOR AN IDEAL HDL :-
  - PARTIAL DESCRIPTIONS
  - ENCOMPASS MATHS
  - EXPRESS INTUITIVE NOTIONS NATURALLY
  - COMPUTER CHECKABLE REASONING
- CANDIDATE :

  HIGHER-ORDER LOGIC

SPECIFYING "DIGITAL" WAVEFORMS

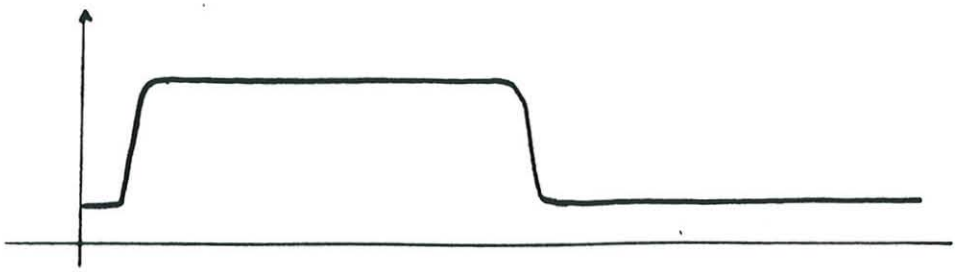
- IN REALITY, <u>ALL</u> WAVEFORMS ARE <u>ANALOGUE</u>.





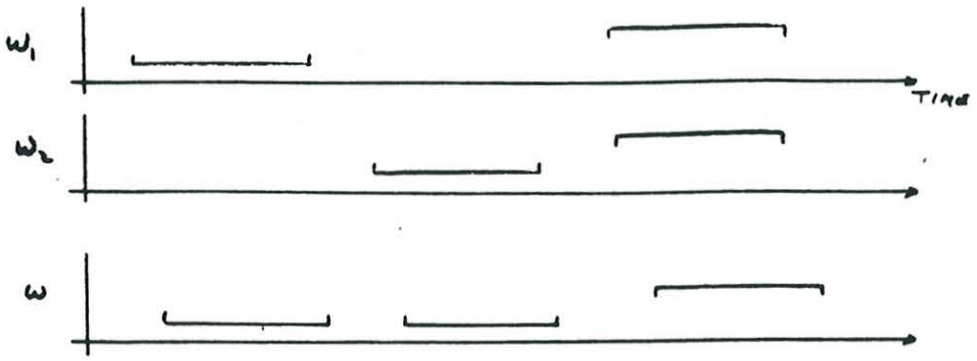
$\vdash$ constant (i , high , w)

$\vdash$ constant (j , low, w)



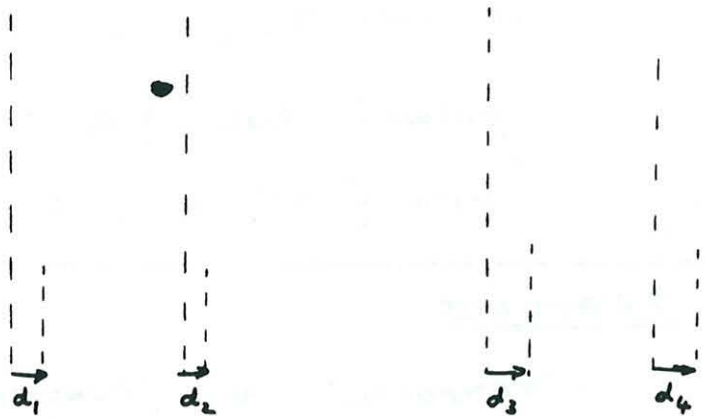
SPECIFYING DIGITAL "BEHAVIOURS"

QUESTION WHAT PROPERTY DO WE REQUIRE OF $\langle w_1\ w_2\ w \rangle$ IN ORDER THAT THEY CHARACTERISE "AND" BEHAVIOUR ?



FEATURES

- PARTIAL
  - A "SPEC." — NOT A "DESCRIPTION"
  - ALLOWS FOR UNCERTAIN KNOWLEDGE



- PARAMETERISED BY 4 DURATIONS
  — "TIMING PARAMETERS"

$$\vec{d} = \langle d_1 \; d_2 \; d_3 \; d_4 \rangle$$

EXPRESSED AS AN AXIOMATIC DEFINITION

ANDBEHAV $(\vec{d})$ $(\omega_1\ \omega_2\ \omega)$

=

$\forall\, i$: interval.

constant(i, low, ω_1) $\vee$ constant(i, low, ω_2)

$\rightarrow$

constant(shift(d_1 d_2 i), low, ω)

$\wedge$

constant(i, high, ω_1) $\wedge$ constant(i, high, ω_2)

$\rightarrow$

constant(shift(d_3 d_4 i), high, ω)

FEATURES

- "TEMPORAL" AND "FUNCTIONAL" ASPECTS ARE INSEPARABLE !
- THE "$\wedge$" IS ALMOST LOST !

STRUCTURES

AIM: TO USE THE TYPE-DISCIPLINE OF THE LOGIC TO GUARANTEE THAT ONLY "WELL-FORMED" CIRCUITS ARE DESCRIBABLE.
[ASSUME "IDEALISED" TTL TECHNOLOGY]

$inport \subseteq port$

$from : inport \to port$

$from\ ip = p$

$W : port \to wf$ DESCRIBES THE WAVEFORM AT A PORT

AXIOM

$\vdash \forall ip : inport.$
$W\ (from\ ip) = W\ ip$

OI2

$\vdash W \circ from = W$

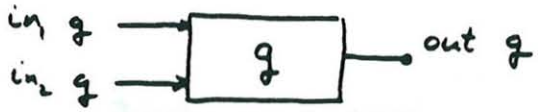
GATES

g : GATE

in_1 : GATE → inport

in_2 : GATE → inport

out : GATE → port



ANDGATE ⊆ GATE

tp : ANDGATE ⟶ dur

AXIOM

$\vdash \forall g$: ANDGATE.

ANDBEHAV $(\vec{d})$ $(\omega_1\ \omega_2\ \omega)$

WHERE

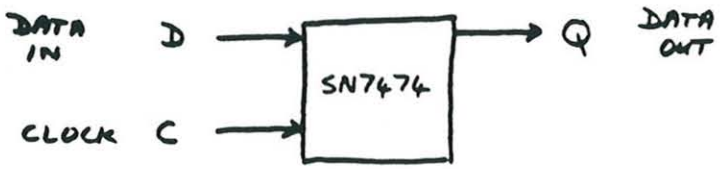
$\vec{d} = tp\ g$

$\omega_1 = (W \circ in_1)\ g$

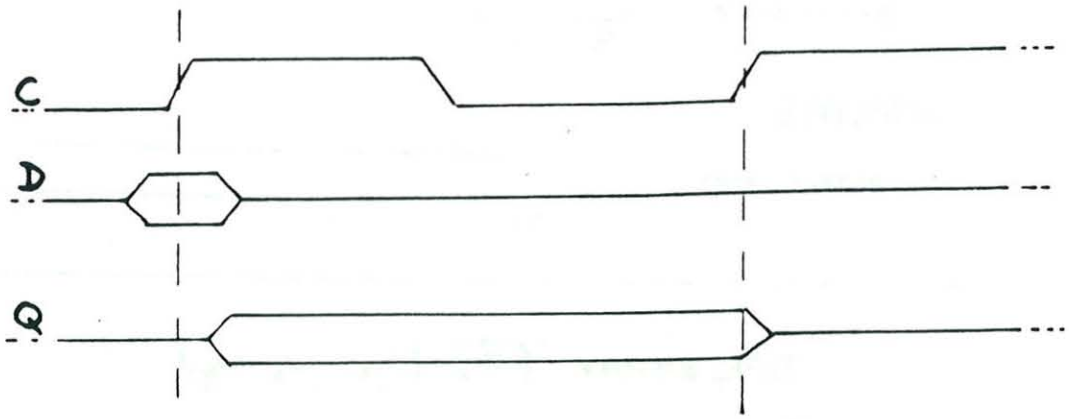
$\omega_2 = (W \circ in_2)\ g$

$\omega = (W \circ out)\ g$

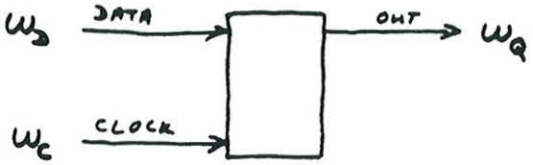
CASE STUDY: EDGE-TRIGGERED D-TYPE FLIPFLOP



TYPICAL MANUFACTURER'S SPEC.



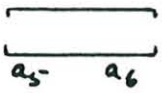
BEHAVIOURAL DEFINITION



CLOCK SPEC.

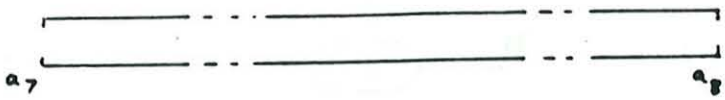
AND

DATA SPEC.



IMPLIES

OUTPUT SPEC.

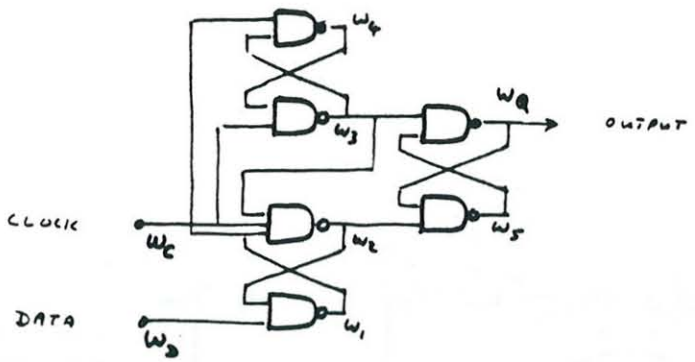


DFF_BEHAV $(\vec{a})$ $(w_c\ w_d\ w_q)$

=

(~ 12 lines)

AN IMPLEMENTATION OF A D-FLIPFLOP



NAND GATE
TIMING PARAMETERS

$\vec{n} = \langle n_1\ n_2\ n_3\ n_4 \rangle$

IT LOOKS SIMPLE ENOUGH, BUT ITS OPERATION IS EXTRAORDINARILY COMPLICATED !

IMPL_BEHAV $(\vec{n})$ $(w_C\ w_D\ w_Q\ w_1\ w_2 \ldots w_5)$

=

NAND_BEHAV $(\vec{n})$ $(w_D\ w_2\ w_1)$

^

NA_ _ _ _

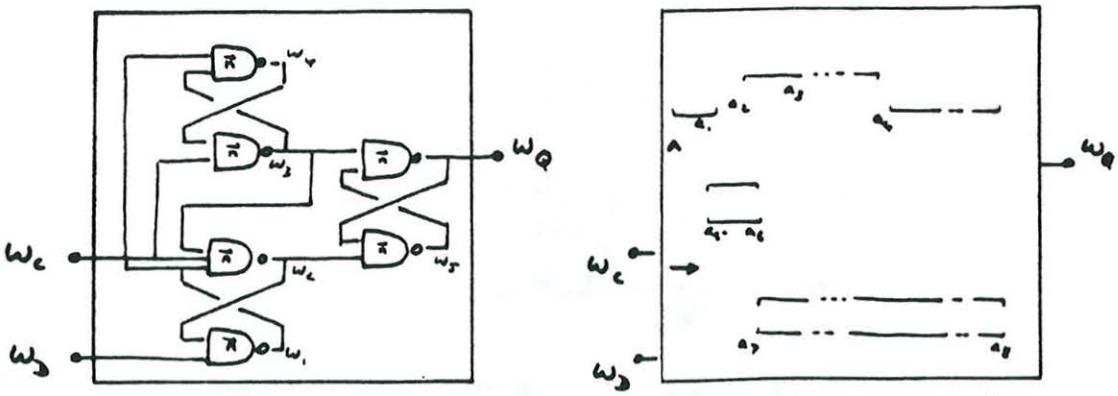
^

⋮

^

NAND_BEHAV $(\vec{n})$ $(w_3\ w_5\ w_Q)$

VERIFICATION



IMPL_BEHAV $(\vec{n})$ DFF_BEHAV $(\vec{a})$

IMPLIES ⟹

GOAL

$$\forall\, w_C\ w_D\ w_Q\ w_1\ w_2\ w_3\ w_4\ w_5 .$$

IMPL_BEHAV $(\vec{n})$ $(w_C\ w_D\ w_Q\ w_1\ w_2\ w_3\ w_4\ w_5)$

$\rightarrow$

DFF_BEHAV $(\vec{a})$ $(w_C\ w_D\ w_Q)$

$$R(\vec{a}, \vec{n}) \rightarrow$$

<u>QUESTION</u>

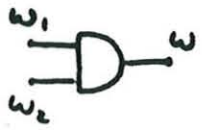
WHAT IS $R(a_1\, a_2\, a_3\, a_4\, a_5\, a_6\, a_7\, a_8\, n_1\, n_2\, n_3\, n_4)$?

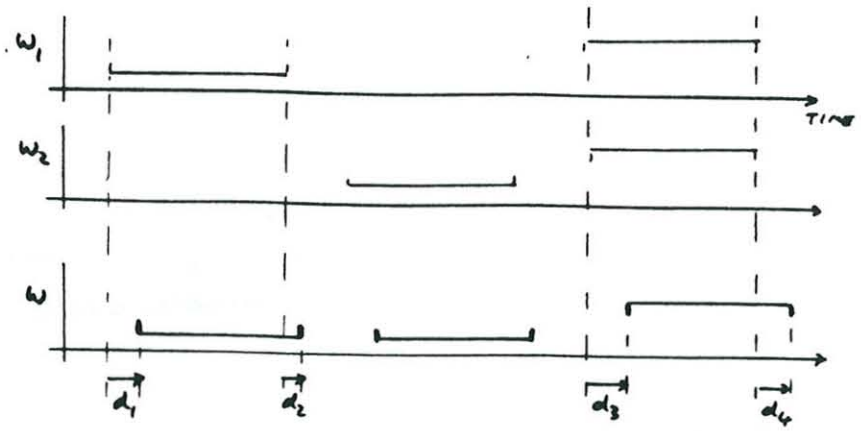
FORMAL VERIFICATION OF DIGITAL SYSTEMS

OVERVIEW

- RECAP ON CASE STUDY (D-flipflop)
- CHOICE OF LOGIC
- AXIOMATISATION
- COMPUTATIONAL IMPLEMENTATION OF A LOGIC
- CONSTRUCTION OF D-flipflop PROOF
- CONCLUSIONS.

RECAP



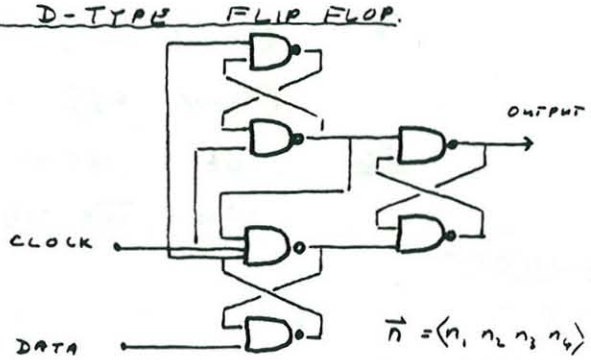


NAND_BEHAV $(\vec{d})$ $(w_1\ w_2\ w)$

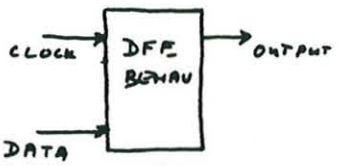
= $\forall$ i: interval.

constant (i, low, w_1) $\vee$ . . .

IMPLEMENTATION (CLAIMED !) OF A D-TYPE FLIP FLOP.



$\vec{n} = \langle n_1\ n_2\ n_3\ n_4 \rangle$



$\vec{a} = \langle a_1\ a_2\ a_3\ a_4\ a_5\ a_6\ a_7\ a_8 \rangle$

THE LOGIC ("VERITAS")

- HIGHER-ORDER

- TYPED
  - AVOID INCONSISTENCY
  - HIGHER-LEVEL e.g.
    m : nat
    d : dur
    w : wf
    g : AND_GATE

- POLYMORPHIC
  - USES ITT "TYPES AS VALUES" SCHEME

EG equal : $\Pi s{:}U0.\ s \times s \to bool$

eg (equal nat) : $nat \times nat \to bool$

(equal wf) : $wf \times wf \to bool$

EG list : $U0 \to U0$

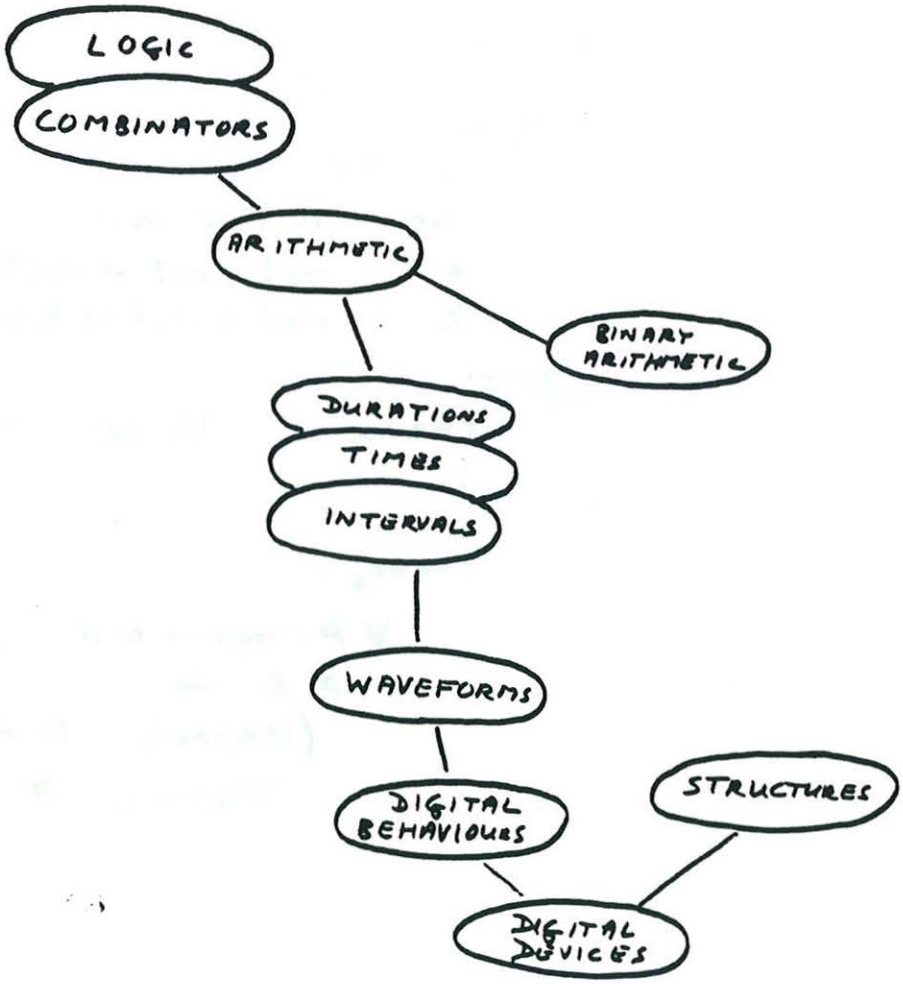
cons : $\Pi s{:}U0.\ s \times (list\ s) \to (list\ s)$

N.B. TRADE OFF

"EXPRESSIVE" LOGIC — GOOD FOR SPECS.

"AUSTERE" LOGIC — GOOD FOR INFERENCING

TYPICAL VERITAS STRUCTURED THEORY PRESENTATION



E.G. ELEMENTARY ARITHMETIC

THEORY Natural_Numbers

SYMBOL

nat : U0

SYMBOL

0 : nat ;

suc : nat → nat ;

+ : nat x nat → nat ;

≥ : nat x nat → bool

AXIOM

Peano$_1$: $\forall n{:}nat.\ \neg(0 = suc\ n)$;

⋮

Peano$_8$:

$\forall P : nat \to bool.$

$P\ 0 \to$

$(\forall n{:}nat.\ P\ n \to P\ (suc\ n)) \to$

$\forall n{:}nat.\ P\ n$;

⋮

THEORGM PROVING

COMPUTATIONAL ASSISTANCE:

- GUARANTEE FREEDOM FROM ERRORS
- MAINTAIN DATABASE OF THEORIES, THEOREMS, etc.
- AUTOMATE LOW-LEVEL INFERENCING

PRINCIPLES OF ML/LCF

OBJECT LOGIC (PPλ)	METALANGUAGE (ML)
	— TYPED, H-O, IMPERATIVE
SYNTACTIC CATEGORIES: TERM THEOREM etc	ABSTRACT TYPES : TERM : THEOREM etc
INSTANCES OF TERM THEOREM etc	VALUES IN ML tm : TERM thm : THEOREM [WELL-FORMEDNESS ENFORCED!]
THEORIES (AXIOMS, CONSTANTS)	STATE OF THE ML INTERPRETOR

ADVANTAGE : SEPARATION OF CONCERNS OF

- SOUNDNESS
- HEURISTICS

METHODOLOGY

FORMAL VERIFICATION IN AN "ENGINEERING" CONTEXT.

USE: TO BACKUP CLAIMS OF ABSOLUTE CERTAINTY OF CORRECTNESS

(OF SMALL, SIMPLE SYSTEMS !)

ASSERTIONS:

- THEOREM PROVER OUGHT TO BE ENGINEERED TO SAME (BETTER ?) STANDARDS AS INTENDED OBJECT SYSTEM.

  $\Rightarrow$

  THEOREM PROVER ITSELF HAVE FORMAL SPEC.

- THEOREM PROVER SPEC
  AXIOMATISATION
  INTERPRETATION
  PROOFS

  } SHOULD BE IN PUBLIC DOMAIN.

"ALGEBRAIC" APPROACH TO THEOREM PROVERS

ALLOWS A PURELY FUNCTIONAL IMPL. OF A T.P.

⇒ CONCISE, "EXECUTABLE SPEC" OF A T.P.

APPROACH

- GIVE SIGNATURES (= LIST OF SYMBOL / AXIOM DECLARATIONS) EQUAL STATUS TO TERMS.
- CONSTRUCT TERMS FROM SIGNATURES
  ⇒ TERMS AS CONTEXT-FREE, PURE VALUES
  ⇒ ALLOWS PURELY FUNCTIONAL META-LANGUAGE.
- USE "PROPOSITIONS AS TYPES" ANALOGY
  - TREAT DERIVATIONS LIKE TERMS

AXIOM	—	SYMBOL
MODUS PONENS	—	APPLICATION
DISCHARGE	—	ABSTRACTION

  - THE TYPE OF A DERIVATION IS THE THEOREM IT ESTABLISHES.

⇒

A LOGIC APPEARS AS A PARTIAL, FREE ALGEBRA.

```
DATATYPE
    term  =  Symbol  sig  int
          |  App  term  term
          |  Abstr  term
          |  ...
    deriv =  Axiom  sig  int
          |  Mp  deriv  deriv
          |  Disch  deriv
          |  ...
    sig   =  Empty
          |  Extend  sig  string  term
```

- wf-symbol
- wf-app
- wf-abstr

- wf-axiom
- wf-mp
- wf-disch

- wf-extend

WITH

```
wf-symbol sg i = (i ≥ 0) andalso (i < len sg)
wf-app tm1 tm2 = ...
```

ADVANTAGES

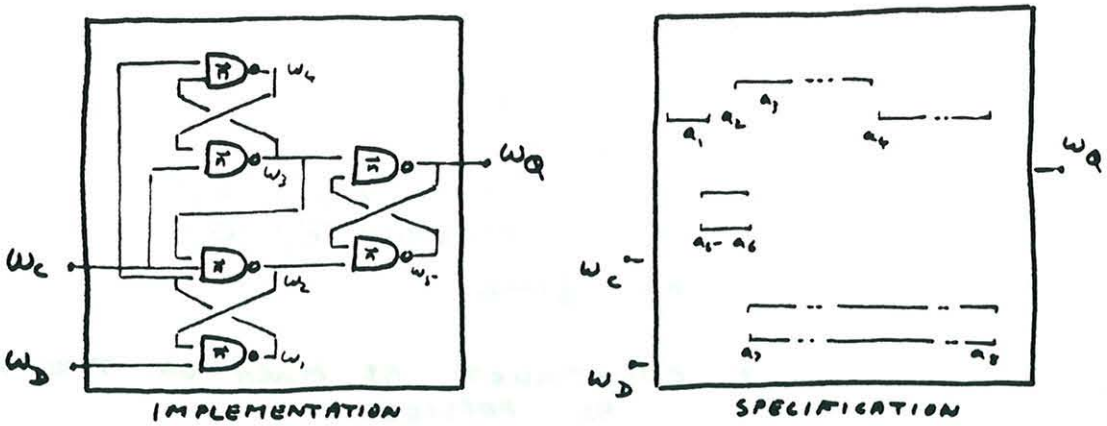
- MATHEMATICALLY SIMPLE.
- ADVANTAGES OF "PATTERN MATCHING"

```
CASE dv OF
    Axiom sg i => ...
    Mp dv1 dv2 => ...
END CASE
```

- FUNCTIONAL PROGRAMMING + PATTERN MATCHING
  ⇒ CONCISE, ELEGANT, "EXEC. SPEC" OF LOGIC
  AND OF TOOLSET
  (PARSERS, EDITORS, SIMPLIFIERS, ...)

PROOF TREE

D-FLIPFLOP CASE STUDY (cont.)



AIM : PROVE THE THEOREM AND DETERMINE R.

$?\vdash R(\vec{a}\ \vec{n}) \rightarrow$

$\forall w_C\ w_D\ w_Q\ w_1\ w_2\ w_3\ w_4\ w_5 \cdot$

IMPL_BEHAV $(\vec{n})$ $(w_C\ w_D\ w_Q\ w_1\ w_2\ w_3\ w_4\ w_5)$

$\rightarrow$

DFF_BEHAV $(\vec{a})$ $(w_C\ w_D\ w_Q)$

$R(a_1\ a_2\ a_3\ a_4\ a_5\ a_6\ a_7\ a_8\ n_1\ n_2\ n_3\ n_4) = ??$

APPROACH

- START WITH

  $\forall \vec{w}.$
  IMPL_BEHAV $(\vec{n})$ $(\vec{w})$
  $\rightarrow$ DFF_BEHAV $(\vec{a})$ $(\vec{w})$

  AS GOAL.

- CONSTRUCT AS MUCH OF THE PROOF AS POSSIBLE

- COLLECT UP DANGLING SUBGOALS

  EG $(a_7 + n_2 < n_3) \wedge (n_2 > 0)$

- CHOOSE A DEFINITION FOR $R(\vec{a}\ \vec{n})$ WHICH IS:
  - STRONG ENOUGH
  - SIMPLE ENOUGH

- "INSERT" IT AS AN AXIOM

- COMPLETE THE PROOF

RESULT

- YES ! IMPLEMENTATION IS CORRECT.
- 1,600 SUBGOALS — 2 WEEKS WORK.
- A "REASONABLE" DEFINITION FOR R IS:

$$
\begin{aligned}
R(\vec{a}, \vec{n}) = \quad & a_1 > 2n_1 + 2n_3 \ \wedge \\
& a_1 > a_5 + n_1 \ \wedge \\
& a_3 > 2n_1 + 2n_3 \ \wedge \\
& n_4 + n_2 \geq a_4 + n_1 \ \wedge \\
& a_5 > 2n_1 + n_3 \ \wedge \\
& a_6 > n_1 + n_3 \ \wedge \\
& a_7 > n_1 + 2n_3 \ \wedge \\
& n_4 + n_2 \geq a_4 + a_8 \ \wedge \\
& n_2 > 0
\end{aligned}
$$

E.G.

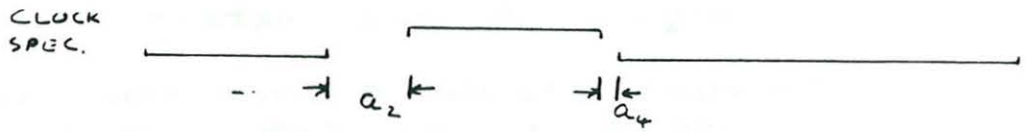
NAND GATES

$n_1, n_3 = 300$ ps

$n_2, n_4 = 200$ ps

RESULTANT D-FLIPFLOP

a_1	=	1400 ps
a_2	=	UNCONSTRAINED
a_3	=	1300 ps
a_4	=	100 ps
a_5	=	1000 ps
a_6	=	700 ps
a_7	=	900 ps
a_8	=	300 ps



CONCLUSIONS

BEHAVIOURAL SPECS

- SPEC. OF EVEN A SIMPLE GATE IS QUITE COMPLEX.
- "LOGICAL" AND "TEMPORAL" ASPECTS ARE INEXTRICABLY LINKED
- RELATIONS, NOT FUNCTIONS (WORLD INDETERMINATE, KNOWLEDGE INCOMPLETE!)

LOGIC

- POLYMORPHIC, HIGHER-ORDER LOGIC IS IDEAL AS A "SPECIFICATION LANGUAGE" FOR DIGITAL SYST.
  - ENCOMPASSES MATHS
  - "HIGHER-ORDER" ⇒ EXPRESSIVE/NATURAL
  - DEDUCTIVE CALCULUS ALREADY EXISTS!

METHODOLOGY

- FOR "REAL" APPLICATIONS, NEED TO SPECIFY THE THEOREM PROVER WITH EQUAL RIGOUR AS THE OBJECT SYSTEM.
- THEOREM PROVER, AXIOMATISATION, AND PROOFS SHOULD BE IN PUBLIC DOMAIN.

"ALGEBRAIC" SPEC OF A LOGIC

- RECOGNISE "SIGNATURES" AS HAVING EQUAL STATUS TO TERMS AND DERIVATIONS

  ⇒ LOGIC APPEARS AS A PARTIAL, FREE ALGEBRA.

  ⇒ PURELY FUNCTIONAL IMPLEMENTATION

  ⇒ CONCISE, ELEGANT, EXECUTABLE SPEC OF THEOREM PROVER.

D-TYPE, EDGE-TRIGGERED FLIPFLOP

- MODUS OPERANDI OF IMPLEMENTATION MUCH MORE COMPLEX THAN IT LOOKS !
- FORMAL VERIFICATION (V. SIMULATION) IS HARD WORK, BUT :
  - RESULTS VALID FOR ALL WAVEFORMS AND ALL TIMING PARAMETERS, AND
  - ONLY PRACTICABLE WAY TO FIND R
- IMPLEMENTATION FOUND TO BE CORRECT
  — BUT NOT "EDGE-TRIGGERED" !

DISCUSSION

Dr. Schneider asked, are you assuming that wires do not have any delays associated with them? Dr. Hanna answered, that is correct. This is obviously an idealisation.

Dr. Hanna was asked would it be possible to introduce a fully synchronised, global, discrete time base? Dr. Hanna answered, we regard this as an interesting problem. We use discrete points in time to specify assertions rather than functions.

Professor Backhouse asked, what about the higher order logic contents of your work? I don't understand this. Dr. Hanna answered, what I mean is the use of higher order functions, although if you understand basic predicate calculus you will hardly notice the higher order functions.

Professor Sintzoff asked, could the case study be implemented by the use of abstract data types?

Dr. Hanna answered, this would be rather complicated, and it would require interactive use. We like to keep things reasonably simple.

Dr. Hanna was asked, can you specify very big problems? Dr. Hanna answered, we have not tried it but I suspect it would be rather difficult.