SPECIFICATION AND VERIFICATION OF DIGITAL SYSTEMS

F.K. Hanna and N. Daeche

Rapporteur: Mr. A.M. Koelmans
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 & VERIFICATION
OF DIGITAL SYSTEMS

KEITH HANNA    NEIL DAECHS
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 VERIFY.
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
  - UNCOMPASS MATHS
  - EXPRESS INTUITIVE NOTIONS NATURALLY
  - COMPUTER CHECKABLE REASONING

- CANDIDATE:
  
  HIGHER-ORDER LOGIC
SPECIFYING "DIGITAL" WAVEFORMS

"IN REALITY, ALL WAVEFORMS ARE ANALOGUE."
high

low

assertions

\[ \text{constant}(i, \text{high, } w) \]
\[ \text{constant}(j, \text{low, } w) \]
**SPECIFYING DIGITAL "BEHAVIOURS"**

**QUESTION** WHAT PROPERTY DO WE REQUIRE OF \( \langle w, w, w \rangle \) IN ORDER THAT THEY CHARACTERISE "AND" BEHAVIOUR?

**FEATURES**
- PARTIAL
- A "SPEC" — NOT A "DESCRIPTION"
- ALLOWS FOR UNCERTAIN KNOWLEDGE
PARAMETERISED BY 4 DURATIONS

"TIMING PARAMETERS"

\[ \vec{d} = (d_1, d_2, d_3, d_4) \]
EXPRESSED AS AN AXIOMATIC DEFINITION

\[ \text{AND-BEHAV}(\alpha)(w_1, w_2, w) = \]

\[ \forall i: \text{interval}, \]

\[ \text{constant}(i, \text{low}, w_1) \lor \text{constant}(i, \text{low}, w_2) \]

\[ \Rightarrow \]

\[ \text{constant}(\text{shift}(d_1, d_2, i), \text{low}, w) \]

\[ \land \]

\[ \text{constant}(i, \text{high}, w_1) \land \text{constant}(i, \text{high}, w_2) \]

\[ \Rightarrow \]

\[ \text{constant}(\text{shift}(d_3, d_4, i), \text{high}, w) \]

FEATURES

- "TEMPORAL" AND "FUNCTIONAL" ASPECTS ARE INSEPARABLE!

- THE "\( \land \)" IS ALMOST LOST!
STRUCTURES

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

\[
\text{import} \subseteq \text{port} \\
\text{from} : \text{import} \to \text{port} \\
\text{\[ \text{from ip} = p \]
}
\]

\[
W : \text{port} \to \text{wf} \quad \text{DESCRIBES THE WAVEFORM AT A PORT}
\]

Axiom

\[
\vdash \forall \text{ip} : \text{import.} \\
W \left( \text{from ip} \right) = W \text{ ip}
\]

Q12

\[
\vdash W \circ \text{from} = W
\]
**GATES**

![Diagram of a gate with inputs and output]

\[ g : \text{GATE} \]

\[ \text{in}_1 : \text{GATE} \rightarrow \text{input} \]

\[ \text{in}_2 : \text{GATE} \rightarrow \text{input} \]

\[ \text{out} : \text{GATE} \rightarrow \text{port} \]

\[ \text{ANDGATE} \subseteq \text{GATE} \]

\[ \text{tp} : \text{ANDGATE} \rightarrow \text{data}^4 \]

**AXIOM**

\[ \forall g : \text{ANDGATE}. \]

\[ \text{AND.BEHAV} (\overline{d}) (w_1, w_2, w) \]

WHERE

\[ \overline{d} = \text{tp} g \]

\[ w_1 = (W \circ \text{in}_1) g \]

\[ w_2 = (W \circ \text{in}_2) g \]

\[ w = (W \circ \text{out}) g \]
CASE STUDY: EDGE-TRIGGERED, D-TYPE FLIPFLOP

DATA IN D → Q DATA OUT
CLOCK C → SN7474

TYPICAL MANUFACTURER'S SPEC.
BEHAVIOURAL DEFINITION

\[
\begin{array}{c}
\text{CLOCK SPEC.} \\
\begin{array}{cccc}
a_1 & a_2 & a_3 & d \\
a_4 & \\
e & \\
\end{array}
\end{array}
\]

AND

\[
\begin{array}{c}
\text{DATA SPEC.} \\
\begin{array}{cc}
a_5 & a_6 \\
\end{array}
\end{array}
\]

IMPLIES

\[
\begin{array}{c}
\text{OUTPUT SPEC.} \\
\begin{array}{ccc}
a_7 & \\
& a_8 \\
\end{array}
\end{array}
\]

\[
\text{DFF BEHAV } \left( \overline{a} \right) \left( \omega_c \omega_d \omega_q \right)
\]

\[
= \text{...}
\]

\[
\text{...}
\]

\[
(\sim 12 \text{ (ncs})
\]
AN IMPLEMENTATION OF A D-FLIPFLOP

It looks simple enough, but its operation is extraordinarily complicated!

impl_behavior (\vec{a}) (w_1, w_2, w_3, w_4, ... w_k) =
NAND_behavior (\vec{a}) (w_1)
\land
NAND_behavior (\vec{a}) (w_2, w_3)
\land
\ldots
\land
NAND_behavior (\vec{a}) (w_2, w_4, w_5)
VERIFICATION

\[ \text{IMPL-BEHAV} (\overline{\mathbf{a}}) \xrightarrow{\text{IMPLIES}} \text{DIFF-BEHAV} (\overline{\mathbf{a}}) \]

\[ \forall w_c, w_d, w_e, w_f, w_g, w_h, w_i, w_j, w_k, w_l, w_m, w_n, w_o, w_p, w_q \cdot \]
\[ \text{IMPL-BEHAV} (\overline{\mathbf{a}}) (w_c, w_d, w_e, w_f, w_g, w_h, w_i, w_j, w_k, w_l, w_m, w_n, w_o, w_p) \rightarrow \]
\[ \text{DIFF-BEHAV} (\overline{\mathbf{a}}) (w_c, w_d, w_e, w_f, w_g) \]

G O A L
$R(\overline{a}, \overline{n}) \rightarrow$

**Question**

What is $R(a, a_2, a_3, a_4, a_5, a_6, a_7, a_8, n, n_2, n_3, n_4)$?
FORMAL VERIFICATION OF
DIGITAL SYSTEMS

OVERVIEW

• Recap on case study (D-84-85)
• Choice of logic
• Axiomatisation
• Computational implementation of a logic
• Construction of D-84-85 proof
• Conclusions.
RECAP

\[ \text{NAND BEHAVIOR(} \bar{d}) (\omega, \omega_L, \omega) \]
\[ = \forall \text{: interval.} \]
\[ \text{constant}(\bar{t}, \text{low}, \omega) \text{, } \bar{t} \ldots \]

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

\[ \bar{a} = (a, a_L, a_0, a_{-1}, a_{-2}, a_{-3}, a_{-4}) \]
THE LOGIC ("VERITAS")

- HIGHER-ORDER
- TYPED
  - AVOID INCONSISTENCY
  - HIGHER-LEVEL e.g. m : nat
d : dur
w : wf
q : AND, Gates
- POLYMORPHIC
  - USES "TYPES AS VALUES" SCHEME

EG equal : \( T \times \mathbb{U} \times \mathbb{U} \times \mathbb{U} \rightarrow \mathbb{U} \)

\( \text{eg} \) (equal nat) : nat \times nat \rightarrow bool
\( \text{eg} \) (equal wf) : wf \times wf \rightarrow bool

EG list : \( \mathbb{U} \times \mathbb{U} \rightarrow \mathbb{U} \)
cons : \( T \times \mathbb{U} \times \mathbb{U} \rightarrow \mathbb{U} \)

N.B. TRADE-OFF
"EXPRESSION" LOGIC — GOOD FOR SPECS.
"AUSTERE" LOGIC — GOOD FOR INFERENCE.
ELEMENTARY ARITHMETIC

THEORY  Natural Numbers

SYMBOL

nat : U0

SYMBOL

0 : nat ;
suc : nat \rightarrow nat ;
+ : nat \times nat \rightarrow nat ;
\geq : nat \times nat \rightarrow bool

AXIOM

Peano, : \forall n : nat . \neg (0 = suc n) ;

Peano_2 : 
\forall P : nat \rightarrow bool .
\ P 0 \rightarrow 
\ (\forall n : nat . \ P n \rightarrow P (suc n)) \rightarrow 
\ \forall n : nat . \ P n ;

THEOREM PROVING

COMPUTATIONAL ASSISTANCE:
- GUARANTEE FREEDOM FROM ERRORS
- MAINTAIN DATABASE OF THEORIES, THEOREM, ET.
- AUTOMATE LOW-LEVEL INFERENCE

PRINCIPLES OF ML/LCF

OBJECT LOGIC (PP)
METALANGUAGE (ML)
- TYPED, H-O, IMPERATIVE

SYNTACTIC CATEGORIES:
ABSTRACT TYPES
- TERM
- THEOREM
t

VALUES IN ML
- TERM
- THEOREM
- THM
[ WELL-FORMEDNESS UNFORCED!]

INstances of
TERM
THEOREM

THEORIES
- AXIOMS, CONSTANTS
- STATE OF THE ML INTERPRETER

ADVANTAGE: SEPARATION OF CONCERNS OF
- SOUNDNESSES
- HEURISTICS
**METHODOLOGY**

**FORMAL VERIFICATION IN AN "ENGINEERING" CONTEXT.**

**USING 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.**

  $\implies$ **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 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**

\[
\begin{align*}
term &= \text{Symbol} \quad \text{sig} \quad \text{int} \\
&\quad \text{App} \quad \text{term} \quad \text{term} \\
&\quad \text{Abstr} \quad \text{term} \\

deriv &= \text{Axiom} \quad \text{sig} \quad \text{int} \\
&\quad \text{Mp} \quad \text{deriv} \quad \text{deriv} \\
&\quad \text{Disch} \quad \text{deriv} \\
\text{sig} &= \text{Empty} \\
&\quad \text{Extend} \quad \text{sig} \quad \text{string} \quad \text{term}
\end{align*}
\]
ADVANTAGES:

- Functional programming is concise, elegant, and easy to implement.
- Pattern matching cases are simple and easy to use.
- With symmetry of symbols (e.g., < = >), the expressions are always the same.
- With axioms and rules, the logic is clear and easy to understand.
**D-Flipflop Case Study (cont.)**

**Goal:** Prove the theorem and determine $R$.

\[ ? \vdash R(\overline{a} \overline{n}) \rightarrow \]
\[ \forall w_c, w_d, w_q, w_1, w_2, w_3, w_4. \]
\[ \text{impl-behav}(\overline{n})(w_c, w_d, w_q, w_1, w_2, w_3, w_4) \rightarrow \]
\[ \text{diff-behav}(\overline{a})(w_c, w_d, w_q) \]

\[ R(a, a_2, a_3, a_4, a_5, a_6, a_7, a_8, n, n, n) = ? \]
APPROACH

• START WITH
  \[ \forall \overline{w}. \]
  \[ \text{impl-behav} (\overline{a}) (\overline{w}) \rightarrow \text{diff-behav} (\overline{a}) (\overline{w}) \]
  AS GOAL.

• CONSTRUCT AS MUCH OF THE PROOF AS POSSIBLE

• COLLECT UP DANGLING SUBGOALS
  \[ \exists \overline{a}, \overline{w} \quad (a_1 + n_1 < n_2) \land (n_2 > 0) \]

• CHOOSE A DEFINITION FOR \( R(\overline{a}, \overline{w}) \) 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:

\[
R(\mathbf{a}, \mathbf{b}) = a_1 > 2n_1 + 2n_2 \land \\
   a_1 > a_5 + n_1 \land \\
   a_2 > 2n_1 + 2n_3 \land \\
   n_4 + n_5 > a_4 + n_1 \land \\
   a_5 > 2n_1 + n_3 \land \\
   a_6 > n_1 + n_3 \land \\
   a_7 > n_1 + 2n_2 \land \\
   n_4 + n_5 > a_4 + a_3 \land \\
   n_5 > 0
\]

E.G.

NAND GATES

\[ n_1, n_3 = 300 \text{ ps} \]
\[ n_2, n_4 = 200 \text{ ps} \]

RESULTANT D-FLIPFLOP

\[ a_1 = 1400 \text{ ps} \]
\[ a_2 = UNCONSTRAINED \]
\[ a_3 = 1300 \text{ ps} \]
\[ a_4 = 100 \text{ ps} \]
\[ a_5 = 1000 \text{ ps} \]
\[ a_6 = 700 \text{ ps} \]
\[ a_7 = 900 \text{ ps} \]
\[ a_8 = 200 \text{ ps} \]

CLOCK SPEC.

\[ -1 \quad \overline{a_2} \quad \overline{a_0} \]

103
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 SYS.
  - 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

1. 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 Flip-Flop

- 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 practically 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.