A Career in Research: Mike Gordon and Hardware Verification

Lawrence C Paulson
The world of computing in 1975

16MB a colossal amount of memory

minicomputers, e.g. 16KB memory

mainframe disk, about 5MB
The world of theory, 1975–80

C. A. R. Hoare. An axiomatic basis for computer programming
Communications of the ACM 12 (10), Oct. 1969

Dana Scott. Outline of a mathematical theory of computation.

operational semantics just emerging
denotational semantics and fixed-point theory

type theory emerging process algebras
Edinburgh LCF (1975–1979)

*Edinburgh LCF: A Mechanised Logic of Computation.* 

- The first real *proof assistant* (for computation theory)
- Introducing ML (the first *polymorphic* functional language)
With so many nascent fields, what did Mike decide to do?

Software is being solved, so let’s verify hardware.

He talked to the hardware experts at Edinburgh and sketched out some theory

... and designed his own computer!
A model of register transfer systems with applications to microcode and VLSI correctness.

Fig. XIV: The host machine for implementing the computer
<table>
<thead>
<tr>
<th>address</th>
<th>ROM</th>
<th>Microinstruction</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td>ready, idle; button→1,0 ; knob+1</td>
<td>branch to 1 if button pressed, otherwise loop; decode knob position</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>rsr, wpc ; 0</td>
<td>switches→PC</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>rsr, wacc ; 0</td>
<td>switches→ACC</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>rps, wmar ; 7</td>
<td>PC→MAR</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>ready ; button→0,6</td>
<td>begin fetch-decode-execute cycle</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>rps, wmar ; 8</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td>racc, write ; 0</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td>read, wir ; 9</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td>; opcode→10</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td>0</td>
<td>decode</td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td>halt</td>
</tr>
<tr>
<td>11</td>
<td></td>
<td>rir, wpc ; 6</td>
<td>JMP; IR→PC</td>
</tr>
<tr>
<td>12</td>
<td></td>
<td>; acc→0 + 11, 17</td>
<td>JZRO:</td>
</tr>
<tr>
<td>13</td>
<td></td>
<td>racc, warg ; 19</td>
<td>ADD; ACC→ARG</td>
</tr>
<tr>
<td>14</td>
<td></td>
<td>racc, warg ; 22</td>
<td>SUB; ACC→ARG</td>
</tr>
<tr>
<td>15</td>
<td></td>
<td>rir, wmar ; 24</td>
<td>LD; IR→MAR</td>
</tr>
<tr>
<td>16</td>
<td></td>
<td>rir, wmar ; 25</td>
<td>ST; IR→MAR</td>
</tr>
<tr>
<td>17</td>
<td></td>
<td>rps, inc ; 18</td>
<td>PC+1→BUF; increment program counter</td>
</tr>
<tr>
<td>18</td>
<td></td>
<td>rbuf, wpc ; 5</td>
<td>BUF→IR</td>
</tr>
<tr>
<td>19</td>
<td></td>
<td>rir, wmar ; 20</td>
<td>IR→MAR</td>
</tr>
<tr>
<td>20</td>
<td></td>
<td>read, add ; 21</td>
<td>ARG+MEM(MAR)→BUF</td>
</tr>
<tr>
<td>21</td>
<td></td>
<td>rbuf, wacc ; 17</td>
<td>BUF→ACC</td>
</tr>
<tr>
<td>22</td>
<td></td>
<td>rir, wmar ; 23</td>
<td>IR→MAR</td>
</tr>
<tr>
<td>23</td>
<td></td>
<td>read, sub ; 21</td>
<td>ARG-MEM(ARG)→BUF</td>
</tr>
<tr>
<td>24</td>
<td></td>
<td>read, wacc ; 17</td>
<td>MEM(MAR)→ACC</td>
</tr>
<tr>
<td>25</td>
<td></td>
<td>racc, write ; 17</td>
<td>ACC→MEM(MAR)</td>
</tr>
</tbody>
</table>

Fig. XVI Microinstruction in control units’ ROM: the microprogram macrocode
internal specifications
even the industrial design!

1. load PC
2. load ACC
3. store
4. run

Switches

PC display lights

ACC display lights

button

O ready

O idle
Mike wrote 21 pages on the computer alone

75 pages on a theory of hardware verification, from gates to computers

Already many key insights by 1981, e.g. to treat combinatorial (stateless) devices like sequential ones.
Three ideas for modelling devices

- 1981: recursive domain equations (*too complicated*)
- 1983: Logic for Sequential Machines, or LSM: Mike’s hardware formalism, loosely based on CCS (*too ad-hoc*)
- 1985: higher-order logic! And all devices as *relations*.

A concerted effort to *minimise* reliance on theory!
A counter using domains (1981)

\[\text{COUNT}(n) = \lambda(switch, in). (out = n + 1), \text{COUNT} (switch + in, n + 1)\]

Fig. II. Behaviour of COUNT(n)
A counter in LSM (1983)

When the counter is clocked, the new value of the state variable $n$ becomes $n+1$ (i.e. the old value plus one) if $false$ is being input on line $switch$, otherwise it becomes the value input on line $in$. We can express this by:

$$CLOCK(n) \rightarrow CLOCK(switch \rightarrow in \mid n+1)$$

In LSM the behaviour of the counter would be specified by the formula

$$COUNT(n) = \text{dev}\{switch, in, out\}. \{out=n\}; COUNT(switch \rightarrow in \mid n+1)$$
A CMOS full adder in HOL (1985)
The insight that devices are relations

\[ D(a, b, c, d) \equiv \exists p \ q. \ D_1(a, b, p) \land D_2(p, d, c) \land D_3(q, b, d) \]
HO Logic was a radical choice!

“Unlike first-order logic and some of its less baroque extensions, second and higher-order logic have no coherent well-established theory; the existent material consisting merely of scattered remarks quite diverse with respect to character and origin.” (Van Benthem and Kees Doets, 1983.)

And we need a type of $n$-bit words, so we need dependent types, right?

WRONG.
Verifying Mike’s computer

- In 1983, using LCF_LSM

“The entire specification and verification described here took several months, but this includes some extending and debugging of LCF_LSM … it would take me two to four weeks to do another similar exercise now. The complete proof requires several hours CPU time on a 2 megabyte Vax 750.”

- In 1986, with Jeff Joyce, Graham Birtwistle, using HOL

- And — under the name Tamarack — by many others!
It was even fabricated!
Verifying a *real* computer: VIPER

- designed by a UK defence lab for military purposes
- specified by a series of abstract layers
- equivalence of the *top three* proved by Avra Cohn
- controversy due to exaggerated claims made by the chip’s marketers
Hardware verification went from plan (1981) to realisation (1989) in eight years!
Some papers from that era

Mike Gordon. Proving a computer correct with the LCF LSM hardware verification system (1983).


“Verification involves a pair of models that bear an uncheckable and possibly imperfect relation to the intended design and to the actual device.”

—Avra Cohn, 1989
“a long term project on verifying combined hardware/software systems by mechanized formal proof”

Mike Gordon, “Mechanizing programming logics in higher order logic”, 1989

- Mike derived Hoare logic in HOL from the operational semantics of a programming language
- supporting the illusion by pretty-printing
- the first shallow embedding of one formalism within another
Whooshing to the year 2000...

Lots of PhDs on floating point arithmetic, process calculi, BDDs, etc., etc....
Verifying the ARM6 processor

• a “commercial off-the-shelf” design

• joint project with Graham Birtwistle at Leeds

• verification by Anthony Fox at Cambridge

• the ARM6 microarchitecture implements its ISA

• And we have a complete formal spec of this machine
Verification and assembly language

(Magnus Myreen, working with Mike)

- Hoare-style logics for assembly languages
- decompilation of assembler to HOL (for 3 machines!)
- proof-producing translation from HOL to assembler

“verifying combined hardware/software systems”
Culminating in CakeML

A dialect of Standard ML, with semantics formalised in HOL

The compiler backend can generate code for 5 different architectures

Source code can be written directly or translated from HOL

Bootstrapped via compilation within HOL4, yielding a “verified binary that provably implements the compiler itself”

The work of Ramana Kumar, Magnus Myreen, Scott Owens, etc.
How Mike accomplished so much

- he ignored “hot topics” to pursue an original plan
- … and talked to “the enemy” across the Department
- … to learn another subject really well
- while using his own knowledge of theory (denotational semantics and CCS)
Learn your application *thoroughly*

E.g. *cryptographic protocols*: such a *simple* field

*... so much flawed research*

*... leading to more bad research*

Because people *weren’t* learning from *security* experts!
Rely on robust tools and theory

- LCF provided a good verification engine
- Higher-order logic was old, solid theory (1940!)
- [Standard ML was emerging; Mike didn’t use it!]

*Type theory would take many years to settle down and would have been a distraction back in 1985.*
Where are we today?

- LCF architectures dominate the landscape
- … using higher-order logic or its extensions
- hardware verification is done extensively in industry
- academic research continues to push forward

But modern processors are still too complex to verify in full!
Mike Gordon
1948–2017