DHDL
The D Hardware Design Language

Luís Marques
<luis@luismarques.eu>
What is hardware?
What is hardware?

You compile this code with DMD:

```d
uint mul(uint a, uint b)
{
    return a * b;
}
```

**Question 1:**
Is this multiplication implemented in hardware? (i.e., is it hardware accelerated?)
What is hardware?

Question 2: What about this one?

```c
uint mul(uint a, uint b) {
    uint sum = 0;
    while(a != 0) {
        if((a & 1) != 0)
            sum += b;
        b <<= 1;
        a >>= 1;
    }
    return sum;
}
```
What is hardware?

You compile this code with DMD DMC

```cpp
#include <iostream>

uint mul(uint a, uint b)
{
    return a * b;
}

int main()
{
    uint a = uniform(0, uint.max);
    uint b = uniform(0, uint.max);
    std::cout << a << ' ' << b << std::endl;
    return 0;
}
```

...and you run it on an original Intel 8086.

**Question 3:**
Is this multiplication implemented in hardware?
What is hardware?

- Each 8086 instruction is executed as $n$ $\mu$ instructions.
- What if I tell you that the $\mu$ code for the `mul` instruction executes this algorithm?
- **Question 4:** Is the `mul` instruction implemented in HW?

```plaintext
uint mul(uint a, uint b) {
    uint sum = 0;
    while(a != 0) {
        if((a & 1) != 0)
            sum += b;
        b <<= 1;
        a >>= 1;
    }
    return sum;
}
```
What is hardware?

• What if there is no μcode, but the multiplication circuit works through a series of additions and shifts?

• **Question 5**: Does it make a difference? Is it (still?) hardware?
What is hardware?

- Software
  - Dynamic allocation of resources
  - Generic
  - Flexible
  - Serial

- Hardware
  - Fixed resources for each function
  - Specialized
  - Hardcoded
  - Parallel
Why Hardware Matters
Why Hardware Matters

As we move from one process to the next, the increase in the number of transistors switching and the frequency with which they switch dominate the decrease in load capacitance and voltage, leading to an overall growth in power consumption and energy. The first microprocessors consumed less than a watt and the first 32-bit microprocessors (like the Intel 80386) used about 2 watts, while a 3.3 GHz Intel Core i7 consumes 130 watts. Given that this heat must be dissipated from a chip that is about 1.5 cm on a side, we have reached the limit of what can be cooled by air.

Given the equation above, you would expect clock frequency growth to slow down if we can't reduce voltage or increase power per chip. Figure 1.11 shows that this has indeed been the case since 2003, even for the microprocessors that were the highest performers each year. Note that this period of flat clock rates corresponds to the period of slow performance improvement range in Figure 1.1.

Between 1978 and 1986, the clock rate improved less than 15% per year while performance improved by 25% per year. During the "renaissance period" of 52% performance improvement per year between 1986 and 2003, clock rates shot up almost 40% per year. Since then, the clock rate has been nearly flat, growing at less than 1% per year, while single processor performance improved at less than 2% per year.
Why Hardware Matters

• Clock frequency has stopped increasing

• Solutions:
  • Multiple cores
  • GPUs
  • FPGAs +parallel +specialized +harder
  • ASICs
Why Hardware Matters

• The datacenter doesn't lie

• E.g., Amazon

• GPU instances

• FPGA instances (f1)
Creating Hardware
Creating Hardware

```d
import core.stdc.stdio;

void main()
{
    int a, op, b, v;

    while(true)
    {
        a  = getchar();
        op = getchar();
        b  = getchar();
        if(op == '+') v = a+b-'0';
        if(op == '-') v = a-b+'0';
        putchar(v);
        putchar('
');
    }
}
```

$ rdmd -version=v1 calc.d
3+4
7
8-5
3
Creating Hardware

```d
import core.stdc.stdio;

void main()
{
    int a, op, b, v, ip;

    while(true)
    {
        if(ip == 0) { a = getchar(); ++ip; }
        if(ip == 1) { op = getchar(); ++ip; }
        if(ip == 2) { b = getchar(); ++ip; }
        if(ip == 3) { if(op == '+') v = a+b-'0'; ++ip; }
        if(ip == 4) { if(op == '-') v = a-b+'0'; ++ip; }
        if(ip == 5) { putchar(v); ++ip; }
        if(ip == 6) { putchar('\n'); getchar(); ip=0; }
    }
}
```

$ rdmd -version=v2 calc.d
3+4
7
8-5
3
Creating Hardware

```c
int a, op, b, v, ip;

while (true)
{
  ip0: if (ip == 0) { a = getchar(); ++ip; goto d; }
  ip1: if (ip == 1) { op = getchar(); ++ip; goto d; }
  ip2: if (ip == 2) { b = getchar(); ++ip; goto d; }
  ip3: if (ip == 3) { if (op == '+') v = a+b-'0'; ++ip; goto d; }
  ip4: if (ip == 4) { if (op == '-') v = a-b+'0'; ++ip; goto d; }
  ip5: if (ip == 5) { putchar(v); ++ip; goto d; }
  ip6: if (ip == 6) { putchar('\n'); getchar(); ip=0; goto d; }
}

d: final switch(uniform(0, 7))
{
  case 0: goto ip0;
  case 1: goto ip1;
  case 2: goto ip2;
  case 3: goto ip3;
  case 4: goto ip4;
  case 5: goto ip5;
  case 6: goto ip6;
}
```

```
$ rdmd -version=v3 calc.d
3+4
7
8-5
3
```
import core.stdc.stdio;

void main()
{
    int a, op, b, v, ip;

    while(true)
    {
        if(ip == 0) { a = getchar(); ++ip; }
        if(ip == 1) { op = getchar(); ++ip; }
        if(ip == 2) { b = getchar(); ++ip; }
        if(ip == 3) { if(op == '+') v = a+b-'0'; ++ip; }
        if(ip == 4) { if(op == '-') v = a-b+'0'; ++ip; }
        if(ip == 5) { putchar(v); ++ip; }
        if(ip == 6) { putchar('
'); getchar(); ip=0; }
    }
}

$ rdmd -version=v2 calc.d
3+4
7
8-5
3
import core.stdc.stdio;

void main()
{
    int a, op, b, v, ip;

    while(true)
    {
        ip = (ip + 1) % 7;
        if(ip == 0) a = getchar();
        if(ip == 1) op = getchar();
        if(ip == 2) b = getchar();
        if(ip == 3) if(op == '+') v = a+b-'0';
        if(ip == 4) if(op == '-') v = a-b+'0';
        if(ip == 5) putchar(v);
        if(ip == 6) putchar('
'); getchar();
    }
}
void main()
{
    int a, op, b, v, ip;

    while(true)
    {
        ip = (ip + 1) % 5;
        if(ip == 0) a = getchar();
        if(ip == 1) op = getchar();
        if(ip == 2) b = getchar();
        if(ip == 3) if(op == '+') v = a+b-'0';
        if(ip == 4) if(op == '-') v = a-b+'0';
    }
}
void main()
{
    int a, op, b, v, ip;

    while(true)
    {
        ip = (ip + 1) % 4;
        if(ip == 0) a = getchar();
        if(ip == 1) op = getchar();
        if(ip == 2) b = getchar();
        if(ip == 3) op == '+' ? v = a+b-'0' : v = a-b+'0';
    }
}
Creating Hardware

```cpp
void main()
{
    int a, op, b, v, ip;

    while(true)
    {
        ip = (ip + 1) % 4;
        if(ip == 0) a = input();
        if(ip == 1) op = input();
        if(ip == 2) b = input();
        if(ip == 3) op == 0 ? v = a+b : v = a-b;
    }
}
```
void main()
{
    int a, op, b, v, ip;

    while(true)
    {
        ip = (ip + 1) % 4;
        if(ip == 0) a  = input();
        if(ip == 1) op = input();
        if(ip == 2) b  = input();
        if(ip == 3) op == 0 ? v = a+b : v = a-b;
    }
}
Creating Hardware

```c
void main()
{
    int a, op, b, v, ip;

    while(true)
    {
        ip = (ip + 1) % 4;
        if(ip == 0) a = input();
        if(ip == 1) op = input();
        if(ip == 2) b = input();
        if(ip == 3) op == 0 ? v = a+b : v = a-b;
    }
}
```

**Question 6:** How many subcircuits are there in here?
Question 6: How many subcircuits are there in here?
Creating Hardware

```cpp
import std.random;
import stdcore.stdio;

void main()
{
    int a, op, b, v, ip;

    while (true)
    {
        ip = (ip + 1) % 4;
        if(ip == 0) a  = input();
        if(ip == 1) op = input();
        if(ip == 2) b  = input();
        if(ip == 3) op == 0 ? v = a + b : v = a - b;
    }
}
```
void main()
{
    int a, op, b, v, ip;

    while(true)
    {
        ip = (ip + 1) % 4;
        if(ip == 0) a = input();
        if(ip == 1) op = input();
        if(ip == 2) b = input();
        if(ip == 3) op == 0 ? v = a+b : v = a-b;
    }
}
void main()
{
    int a, op, b, v, ip;
    while(true)
    {
        ip = (ip + 1) % 4;
        if(ip == 0) a = input();
        if(ip == 1) op = input();
        if(ip == 2) b = input();
        if(ip == 3) op == 0 ? v = a+b : v = a-b;
    }
}
void main()
{
    int a, op, b, v, ip;

    while(true)
    {
        ip = (ip + 1) % 4;
        if(ip == 0) a = input();
        if(ip == 1) op = input();
        if(ip == 2) b = input();
        if(ip == 3) \textcolor{orange}{op == 0} ? v = a+b : v = a-b;
    }
}
Creating Hardware

```cpp
import std.random;
import core.stdc.stdio;

void main()
{
    int a, op, b, v, ip;

    while(true)
    {
        ip = (ip + 1) % 4;
        if(ip == 0) a = input();
        if(ip == 1) op = input();
        if(ip == 2) b = input();
        if(ip == 3) op == 0 ? v = a+b : v = a-b;
    }
}
```
void main()
{
    int a, op, b, v, ip;

    while(true)
    {
        ip = (ip + 1) % 4;
        if(ip == 0) a = input();
        if(ip == 1) op = input();
        if(ip == 2) b = input();
        if(ip == 3) op == 0 ? v = a+b : v = a-b;
    }
}
Creating Hardware

```cpp
import std.random;
import core.stdc.stdio;

void main()
{
    int a, op, b, v, ip;

    while(true)
    {
        ip = (ip + 1) % 4;
        if(ip == 0) a = input();
        if(ip == 1) op = input();
        if(ip == 2) b = input();
        if(ip == 3) op == 0 ? v = a+b : v = a-b;
    }
}
```
void main()
{
    int a, op, b, v, ip;

    while(true)
    {
        ip = (ip + 1) % 4;
        if(ip == 0) a = input();
        if(ip == 1) op = input();
        if(ip == 2) b = input();
        if(ip == 3) op == 0 ? v = a+b : v = a-b;
    }
}
void main()
{
    int a, op, b, v, ip;
    while(true)
    {
        ip = (ip + 1) % 4;
        if(ip == 0) a = input();
        if(ip == 1) op = input();
        if(ip == 2) b = input();
        if(ip == 3) op == 0 ? v = a+b : v = a-b;
    }
}
Creating Hardware

```c
void main()
{
    int a, op, b, v, ip;

    while(true)
    {
        ip = (ip + 1) % 4;
        if(ip == 0) a = input();
        if(ip == 1) op = input();
        if(ip == 2) b = input();
        if(ip == 3) op == 0 ? v = a+b : v = a-b;
    }
}
```
Creating Hardware

```cpp
void main()
{
    int a, op, b, v, ip;

    while (true)
    {
        ip = (ip + 1) % 4;
        if (ip == 0) a = input();
        if (ip == 1) op = input();
        if (ip == 2) b = input();
        if (ip == 3) op == 0 ? v = a + b : v = a - b;
    }
}
```
void main()
{
    int a, op, b, v, ip;

    while(true)
    {
        ip = (ip + 1) % 4;
        if(ip == 0) a = input();
        if(ip == 1) op = input();
        if(ip == 2) b = input();
        if(ip == 3) op == 0 ? v = a+b : v = a-b;
    }
}
void main()
{
    int a, op, b, v, ip;

    while(true)
    {
        ip = (ip + 1) % 4;
        if(ip == 0) a = input();
        if(ip == 1) op = input();
        if(ip == 2) b = input();
        if(ip == 3) op == 0 ? v = a+b : v = a-b;
    }
}
```c
void main()
{
    int a, op, b, v, ip;

    while(true)
    {
        ip = (ip + 1) % 4;
        if(ip == 0) a = input();
        if(ip == 1) op = input();
        if(ip == 2) b = input();
        if(ip == 3) op == 0 ? v = a+b : v = a-b;
    }
}
```
void main()
{
    int a, op, b, v, ip;

    while(true)
    {
        ip = (ip + 1) % 4;

        if(ip == 0) a = input();
        if(ip == 1) op = input();
        if(ip == 2) b = input();
        if(ip == 3) op == 0 ? v = a+b : v = a-b;
    }
}
Creating Hardware

```cpp
void main()
{
    int a, op, b, v, ip;

    while(true)
    {
        ip = (ip + 1) % 4;
        if(ip == 0) a = input();
        if(ip == 1) op = input();
        if(ip == 2) b = input();
        if(ip == 3) op == 0 ? v = a+b : v = a-b;
    }
}
```
void main()
{
    int a, op, b, v, ip;

    while(true)
    {
        ip = (ip + 1) % 4;
        if(ip == 0) a = input();
        if(ip == 1) op = input();
        if(ip == 2) b = input();
        if(ip == 3) op == 0 ? v = a+b : v = a-b;
    }
Creating Hardware

```cpp
import std.random;
import core.stdc.stdio;

void main()
{
    int a, op, b, v, ip;

    while(true)
    {
        ip = (ip + 1) % 4;
        if(ip == 0) a  = input();
        if(ip == 1) op = input();
        if(ip == 2) b  = input();
        if(ip == 3) op == 0 ? v = a+b : v = a-b;
    }
}
```
void main()
{
    int a, op, b, v, ip;
    
    while (true)
    {
        ip = (ip + 1) % 4;
        switch (ip)
        {
            case 0: a = input();
            case 1: op = input();
            case 2: b = input();
            default: if (op == 0) v = a + b; else v = a - b;
        }
    }
}
Question 7: There’s an important optimization left to do in this circuit. What is it?
Question 7

• Optimization: \(-x = \neg x + 1\); \(a+b = a+(\neg b)+1\)

\[
\text{if}(\text{ip} == 3) \text{ op } == 0 \ ? \ v = a+b \ : \ v = a-b;
\]

\[
\text{if}(\text{ip} == 3) \ v = a + (\text{op} == 0 \ ? \ b : \neg b + 1);
\]
Question 8

- **Question 8**: do you know what an exclusive or (XOR) is?

  [  ] Yes
  [  ] No
  [  ] Is this a trick question?
Question 8

- **Question 8**: do you know what an exclusive or (XOR) is?

[ ] Yes
[ ] No
[ ] Is this a trick question?

<table>
<thead>
<tr>
<th>INPUT</th>
<th>OUTPUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

```c
if (ip == 3) v = a + (op == 0 ? b : ~b + 1);
```
Bus splitter
(1x4 bits to 4x1bit)

XOR

Bus joiner
Creating Hardware

- Approach 1: Describe the algorithm, derive the circuit from the description (high level synthesis)

```c
int a, op, b, v;

while(true)
{
    a = getchar();
    op = getchar();
    b = getchar();
    if(op == '+') v = a + b - '0';
    if(op == '-') v = a - b + '0';
    putchar(v);
    putchar(\n); getchar();
}
```
Creating Hardware

• Approach 2: Write an event-based simulation of your design. Let the tools infer what hardware would be consistent with that simulation.

• In practise, only the simulations for which there is an obvious mapping to hardware work. That’s the so-called synthesizable subset.
Creating Hardware

- Approach 3: Describe the circuit (RTL generator)

```cpp
void main()
{
    auto a = new Reg();
    auto op = new Reg();
    auto b = new Reg();
    auto ip = new Reg();
    auto adder = new Adder;
    adder.connect(a, b);
    auto dec = new Decoder();
    dec.connect(ip);
    // ... etc
}
```
Creating Hardware

- Approach 1: High-level synthesis
  Approach 2: Event-based
  Approach 3: RTL builder

- VHDL & Verilog: Follow mostly approach 2

- DHDL (and Chisel, SpinalHDL, etc.): Follows mostly approach 3

- Not pure: You don’t have to specify events / gates for combinational circuits, just write \((a + b) \times 7\) etc.
DHDL

- What is DHDL?
  - A hardware design language
  - D Language extension + D library
  - Falls more on the RTL generator side (currently)

- Why DHDL?
  - Modern design
Modern Design

- `int sum(int[] values);`
- `int mul(int[] values);`
- `auto accumulate(alias op, T)(T[] values);`

Level 1: one tool does multiple jobs

Level 0: one tool does one job

Level 1: generates multiple tools, each of which do one job
Modern Design

Question 9:
At what level do hardware designers work?

Level -1
(sometimes...
Modern Design

- Level -1:

- Level 0: one tool does one job
  (e.g. max computes maximums)

- Level 1: one tool does multiple jobs
  (e.g. accumulate computes sums, maximums, etc.)
Modern Design

- Level -1: multiple tools needed to do one job
  \( \text{sum\_part1()}, \text{sum\_part2()} \)

- Level 0: one tool does one job
  (e.g. max computes maximums)

- Level 1: one tool does multiple jobs
  (e.g. accumulate computes sums, maximums, etc.)
HDL Woes

• VHDL was created to create simulations of hardware

• After debugging the design, the hardware was built by converting to a schematic, where you wired simple gates, etc.

• Why not make the conversion automatic?

• Hardware engineers were used to schematic entry as a design method. VHDL was nailed with that hammer.

• Lots of little blocks without obvious relationship
HDL Woes

- void sum_part1()
  {
    thread
  }

- void sum_part2()
  {
    thread
  }
HDL Woes

CBandDatat_LatchPROC9F: process(MDLE, CB_In, Reset_Out_N)
begin
  if Reset_Out_N = '0' then
    CBLatch_F_1       <= "0000";
  elsif MDLE = '1' then
    CBLatch_F_1       <= CB_In(3 downto 0);
  end if;
end process;

CBandDatat_LatchPROC10F: process(MDLE, CB_In, DParIO_In, Reset_Out_N)
begin
  if Reset_Out_N = '0' then
    CBLatch_F_2       <= "0000";
  elsif MDLE = '1' then
    CBLatch_F_2(6 downto 4) <= CB_In(6 downto 4);
    CBLatch_F_2(7)          <= DParIO_In;
  end if;
end process;

CBLatch_F <= CBLatch_F_2 & CBLatch_F_1;

BullEn_PROC: process(Mem_Test, ByteSel)
begin
  BullEn = <not (ByteSel(0) and ByteSel(1) and ByteSel(2) and ByteSel(3)) or Mem_Test;
end process;

IUChk_Out_Gen: process (IUDataLatch_F, ChkGen_Data, BullEn, CB_bull)
  Variable IUGen_Chk : std_logic_vector(7 downto 0);
begin
  IUGen_Chk(6 downto 0) := ChkGen (IUDataLatch_F);
  IUGen_Chk(7)          := ChkGen_Data(32);
  CB_Out_Int <= mux2 (BullEn, IUGen_Chk, CB_bull);
end process;
HDL Woes

• The argument is:
  • VHDL was originally created for simulation
  • Synthesis was added later
  • Problems arose from that
  • But then at least things related to simulation should be pretty good, right?
entity hello_world is
end hello_world;

architecture behaviour of hello_world is
  signal foo : std_logic;
begin
  process begin
    foo <= '1';
    wait for 1 ns;
    report std_logic'image(foo);
    foo <= '0';
    wait for 1 ns;
    report std_logic'image(foo);
    wait;
  end process;
end behaviour;

$ ghdl -a hello.vhdl && ghdl -e hello_world
$ ./hello_world
hello.vhdl:13:8:@1ns:(report note): '1'
hello.vhdl:16:8:@2ns:(report note): '0'
HDL Woes

entity hello_world is
end hello_world;

architecture behaviour of hello_world is
  signal foo : std_logic_vector(7 downto 0);
begin
  process begin
    foo <= "11111111";
    wait for 1 ns;
    report std_logic_vector'image(foo);
    foo <= "00000000";
    wait for 1 ns;
    report std_logic_vector'image(foo);
    wait;
  end process;
end behaviour;

$ ghdl -a hello.vhdl && ghdl -e hello_world
hello.vhdl:13:32: prefix of 'image attribute must be a scalar type
hello.vhdl:13:32: found array type
"std_logic_vector" defined at ../../../src/ieee/std_logic_1164.v93:69:30
HDL Woes

Google search for "vhdl print std_logic_vector"
How to Display std_logic_vector Value? - Google Groups

Jan 15, 2001 - comp.lang.vhdl ... How can I display the value of std_logic_vector? ... a) Both compilers complain data_out is incorrect type for REPORT.

VHDL BLOG: Print to STDOUt using REPORT Statement in VHDL

Aug 12, 2013 - variable B : std_logic := '1'; report "B value is" & std_logic'image(B); output > B is 1. (C) To print std_logic_vector type values to STDOUT ...

How do I use VHDL write() function to print std_logic_vector type ...

May 23, 2012 - 3 posts - 2 authors
Can anyone tell me how to do it? I used to use report() function, but recently learned that write() can be used for printing things on a screen.

How to write ucf file in xilinx for vhdl code for an ... 6 posts 28 Apr 2014
[SOLVED] decimal numbers in vhdl - EDA Board 12 posts 3 Aug 2011
More results from www.edaboard.com

How to print std_logic_vector variable into hex string in VHDL ...

Aug 31, 2005 - 5 posts - 4 authors
How to print std_logic_vector variable into hex string in VHDL ... Is there any command like "report integer'image(myvalue)"; so that it will print ...

vhdl - Is there a way to print the values of a signal to a file from a ...

Jun 20, 2014 - First make functions that convert std_logic and std_logic_vector to string like: function to_bstring(sl : std_logic) return string is variable sl_str_v ...
You may want to use this function:

```vhdl
-- By PK debug mean for printing std_logic
function prtstd( v : std_logic_vector ) return integer is
variable s : string( 3 downto 1 );
variable r : string( v'left+1 downto v'right+1 );
begins
  for i in v'left downto v'right loop
    --report std_logic'image(v(i));
    s := std_logic'image(v(i));
    --string must start/stop at 1
    -- '1' we need only the second character
    r(i+1) := s(2);
  end loop;
  report "dbg by pk " & r & " dbg end";
return 0;
end prtstd;
```

Call it by:
```vhdl```,
junk := prtstd( FileDesc.CurrentWord );
```

In article <940isc$qfe$1...@nnrp1.deja.com>,

- show quoted text -
How to Display std_logic_vector Value? - Google Groups
https://groups.google.com/d/topic/comp.lang.vhdl/Zky1UFJlu4j ▼
Jan 15, 2001 - comp.lang.vhdl › ... How can I display the value of std_logic_vector? ... a) Both compilers complain data_out is incorrect type for REPORT.

VHDL BLOG: Print to STDOUT using REPORT Statement in VHDL
sagekingthegreat.blogspot.com/2013/08/report-statement-report-statement.html ▼
Aug 12, 2013 - variable B : std_logic := '1'; report "B value is" & std_logic'image(B); output > B value is 1. (C) To print std_logic_vector type values to STDOUT ...

How do I use VHDL write() function to print std_logic_vector type ...
www.edaboard.com › ... › PLD, SPLD, GAL, CPLD, FPGA Design ▼
May 23, 2012 - 3 posts - 2 authors
Can anyone tell me how to do it? I used to use report() function, but recently learned that write() can be used for printing things on a screen.

[ SOLVED ] decimal numbers in vhdl - EDA Board
12 posts 3 Aug 2011
More results from www.edaboard.com

How to print std_logic_vector variable into hex string in VHDL ...
https://www.thecodingforums.com › Archive › Archive › VHDL ▼
Aug 31, 2005 - 5 posts - 4 authors
How to print std_logic_vector variable into hex string in VHDL ... Is there any command like "report integer'image(myvalue)"; so that it will print ...

vhdl - Is there a way to print the values of a signal to a file from a ...
stackoverflow.com › .../is-there-a-way-to-print-the-values-of-a-signal-to-a-file-from-a-... ▼
Jun 20, 2014 - First make functions that convert std_logic and std_logic_vector to string like: function to_bstring(sl : std_logic) return string is variable sl_str_v ...
HDL Woes

As you have discovered, the 'image attribute is only declared for scalar types, not arrays or records: the usual approach is to create one's own library of test utilities including `to_string` or `image` functions in a package at the start of a design, and use it throughout.

It would be perfectly possible to standardise a library of these, and you will probably find many potential "test utility" packages but none has really caught on well enough to deserve becoming a standard.

That being said, you might find the following package a useful starting point.

- wat?
- Is this what the competition looks like? You have to create your own writeln?
HDL Woes

• More seriously, here are the fundamental problems:
  
  • VHDL & Verilog are event-based
  
  • Only a subset is synthesizable
  
  • Only a subset of that subset actually works in practise.
  
  • Which sub-sub-set? Oh, it depends on the tool vendor!
  
  • No OOP, FP or generics. Everything is verbose, repetitive and error prone
HDL Woes

- Only VHDL and Verilog are mainstream, but there are some competitors:
  - Bluespec, Lava, Hydra, ForSyDe, etc. (Haskell)
  - Chisel / SpinalHDL (Scala)
  - MyHDL (Python)
  - CLaSH
  - Lucid
Why DHDL

- DHDL
  - Chisel’s hardware construction model…
  - …without the Scala DSL tax
  - …with D’s modeling power
Why DHDL

- Scala DSL tax

```
if(a == false)
  b := 42;
else
  c := 0;
(DHDL)
```

```
when(a === false.B)
{
  b := 42.U
}
.otherwise
{
  c := 0.U
}
(Chisel)
```
Why DHDL

• Compiling DHDL
Why DHDL

• Compiling DHDL
• What we’ll see today
FPGA

- What’s an FPGA?
  - The largest VLIW CPU in the world
  - An interpreter for hardware
FPGA
FPGA
FPGA

- We’ll be using a Papilio Pro in this talk
  - Spartan 6 LX9 FPGA
    - 1,430 slices
    - 32 x 18 Kb block RAM
    - 16 x DSP48A1 (18x18 mul)
    - 8 Megabytes SDRAM
    - 32 MHz crystal oscillator
    - 48 I/O pins
    - Papilio Wing form factor
Papilio Wings
<table>
<thead>
<tr>
<th>GND</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td>3V3</td>
<td>2</td>
</tr>
<tr>
<td>5V</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>7</td>
</tr>
</tbody>
</table>

Papilio Wings
Papilio Wings
Papilio Wings

- Audio
- Buttons+LEDs
- Joystick
- SD Card
- VGA
- Wii Controller
Papilio Wings

Breadboard

I/O Buffer (Logic Analyser)
Papilio MegaWings

LogicStart

RetroCade

Arcade
How to Win an FPGA and Influence Hardware

LogicStart

Papilio Pro

Wings
Hello World

```dhd1
helloworld.dhd1

circuit HelloWorld
{
    port in bool a;
    port out bool b;

    b := a;
}
```
Hello World

helloworld.dhdl

circuit HelloWorld
{
    port in bool a;
    port out bool b;

    b := a;
}

a

b
Hello World

helloworld.dhdl

circuit HelloWorld
{
  port in bool a;
  port out bool b;

  b := a;
}

unittest
{
  auto hello = peekPokeTester!HelloWorld;

  hello.a = false;
  hello.eval();
  assert(hello.b == false);

  hello.a = true;
  hello.eval();
  assert(hello.b == true);
}
Hello World

```dhdl
# helloworld.dhdl

Circuit HelloWorld
{
    port in bool a;
    port out bool b;

    b := a;
}
```

```d
# HelloWorld.d

class HelloWorld : Circuit
{
    Bool a;
    Bool b;

    this()
    {
        a = input("a", new Bool);
        b = output("b", new Bool);
        assign(b, a);
    }
}
```
Hello World

HelloWorld.fir

circuit HelloWorld :
  module HelloWorld :
    input reset : UInt<1>
    input clock : Clock
    input a : UInt<1>
    output b : UInt<1>

    b is invalid
    b <= a

HelloWorld.d

class HelloWorld : Circuit
{
  Bool a;
  Bool b;

  this()
  {
    a = input("a", new Bool);
    b = output("b", new Bool);

    assign(b, a);
  }
}
Hello World

HelloWorld.fir

circuit HelloWorld :
    module HelloWorld :
        input reset : UInt<1>
        input clock : Clock
        input a : UInt<1>
        output b : UInt<1>

        b is invalid

        b <= a

HelloWorld.v

module HelloWorld(
    input   reset,
    input   clock,
    input   a,
    output  b
);
    assign b = a;
endmodule
Hello World

VHelloWorld.h

```
// VL_MODULE (VHelloWorld) {
public:
    VL_IN8(reset, 0, 0);
    VL_IN8(clock, 0, 0);
    VL_IN8(a, 0, 0);
    VL_OUT8(b, 0, 0);
(...)
```

HelloWorld.v

```
module HelloWorld(
    input   reset,
    input   clock,
    input   a,
    output  b
);
    assign b = a;
endmodule
```

VHelloWorld.cpp

```
(...)
```

VHelloWorld (shared library)

```
(...)
```

dhdl\testing.d

```
(...)
```

auto r = ctRegex!r"\s*VL_((\[A-Z]+\[0-9]*))\((\[a-z]+),([0-9]+),";
Hello World

helloworld.dhdl

circuit HelloWorld
{
    port in bool a;
    port out bool b;

    b := a;
}

unittest
{
    auto hello = peekPokeTester!HelloWorld;

    hello.a = false;
    hello.eval();
    assert(hello.b == false);

    hello.a = true;
    hello.eval();
    assert(hello.b == true);
}
Hello World

circuit HelloWorld
{
    port in bool a;
    port out bool b;

    b := a;
}

module HelloWorld(
    input reset,
    input clock,
    input a,
    output b
);
    assign b = a;
endmodule

NET a LOC="P47"  |  IOSTANDARD=LVTTL;
NET b LOC="P112"  |  IOSTANDARD=LVTTL;
Hello World

GND 1

3V3 2

Button 1 3

Button 2 4

Button 3 5
Hello World

- GND 1
- 3V3 2
- Button 1 3
- Button 2 4
- Button 3 5

SPDT
Hello World
Hello World
Hello World
Hello World
Operators

operators.dhdl

circuit Operators
{
    port in bool a;
    port in bool b;
    port out bool c;

    b := a & b;
}

---

(a & b) circuit diagram

- Input ports: a, b
- Output port: c
- AND gate

a b c
Operators

```plaintext
operators.dhdl

circuit Operators {
    port in bool a;
    port in bool b;
    port out bool c;

    b := a & b;
}

unittest {
    auto op = peekPokeTester!Operators;
    op.a = false;
    op.b = true;
    op.eval();
    assert(op.c == false);
    op.a = true;
    op.eval();
    assert(op.c == true);
}
```
Operators

operators.dhdl

```plaintext
circuit Operators
{
    port in bool a;
    port in bool b;
    port out bool c;

    b := a & b;
}
```

Operators.d

```plaintext
class Operators : Circuit
{
    Bool a;
    Bool b;
    Bool c;

    this()
    {
        a = input("a", new Bool);
        b = input("b", new Bool);
        c = output("c", new Bool);

        assign(c, a.and(b));
    }
}
```
Operators

Operators.fir

circuit Operators:
    module Operators:
        input reset : UInt<1>
        input clock : Clock
        output c : UInt<1>
        input a : UInt<1>
        input b : UInt<1>

        c is invalid

        node _T_14 = and(a, b)
        c <= _T_14

Operators.d

class Operators : Circuit
{
    Bool a;
    Bool b;
    Bool c;

    this()
    {
        a = input("a", new Bool);
        b = input("b", new Bool);
        c = output("c", new Bool);

        assign(c, a.and(b));
    }
}
Operators

Operators.fir

```plaintext

circuit Operators :
    module Operators :
        input reset : UInt<1>
        input clock : Clock
        output c : UInt<1>
        input a : UInt<1>
        input b : UInt<1>

        c is invalid

        node _T_14 = and(a, b)
        c <= _T_14
```

Operators.v

```plaintext

module Operators(
    input reset,
    input clock,
    output c,
    input a,
    input b
);

    wire _T_14;
    assign c = _T_14;
    assign _T_14 = a & b;
endmodule
```
Operators
State

```dhdl
blink.dhdl

circuit Blink
{
    port out bool a;

    reg bool state;

    state := !state;
    a := state;
}
```
circuit Blink
{
    port out bool a;

    reg bool state;

    a := state;
    state := !state;
}
circuit Blink
{
    port out bool a;
    reg bool state;
    a := state;
    state := !state;
}
circuit Blink
{
    port out bool a;
    reg bool state;
    a := state;
    state := !state;
}

State
circuit Blink
{
    port out bool a;
    reg bool state;
    a := state;
    state := !state;
}
circuit Blink
{
  port out bool a;
  reg bool state;
  a := state;
  state := !state;
}
circuit Blink
{
    port out bool a;
    reg bool state;
    a := state;
    state := !state;
}

Question 10: Will port a blink?
State

`blink.dhdl`

```dhdl
circuit Blink {
    port out bool a;
    reg bool state;
    a := state;
    state := !state;
}
```
State

```dhdl
circuit Blink
{
  port out bool a;
  reg bool state;
  a := state;
  state := !state;
}
```

```unittest
{  
  auto blink = peekPokeTester!Blink;

  foreach(i; 0..8)  
  {  
    assert(blink.a == i % 2);
    blink.step();
  }
}
```

**Question 11**: Will the LED (on port a) blink?
State
32 Mhz Crystal Oscillator

constraints.ucf

NET clock LOC="P94" | IOSTANDARD=LVTTL | PERIOD=31.25ns;
NET a LOC="P112" | IOSTANDARD=LVTTL;
enum clockRate = 32_000_000;

circuit Blink2
{
    port out bool a;

    reg bool state;
    reg uint counter = clockRate;

    a := state;

    if(counter != 0)
        counter := counter - 1;
    else
    {
        counter := clockRate
        state := !state;
    }
}

**Question 12:** Now, will the LED blink? (at 1 Hz)
State
State - width inference

```
blink2.dhdl

enum clockRate = 32_000_000;

circuit Blink2
{
    port out bool a;

    reg bool state;
    reg uint counter = clockRate;

    a := state;

    if(counter != 0)
        counter := counter - 1;
    else
    {
        counter := clockRate
        state := !state;
    }
}
```
State - width inference

blink2.dhdl

enum clockRate = 32_000_000;

circuit Blink2
{
    port out bool a;

    reg bool state;
    reg uint!25 counter = clockRate;

    a := state;

    if(counter != 0)
        counter := counter - 1;
    else
    {
        counter := clockRate
        state := !state;
    }
}
State - width inference

```dhdl
enum clockRate = 32_000_000;

circuit Blink2
{
    port out bool a;

    reg bool state;
    reg uint!(clockRate.log2Up) counter = clockRate;

    a := state;

    if(counter != 0)
        counter := counter - 1;
    else
    {
        counter := clockRate
        state := !state;
    }
}
```
IO - UART

byte 0   byte 2   byte 3   ...   bit 7
IO - UART

```
<table>
<thead>
<tr>
<th>bit 0</th>
<th>byte 0</th>
<th>byte 2</th>
<th>byte 3</th>
<th>...</th>
<th>bit 7</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bit 0</td>
<td>bit 1</td>
<td>bit 2</td>
<td>bit 3</td>
<td>bit 4</td>
<td>bit 5</td>
</tr>
</tbody>
</table>
```
circuit UARTTx(int symbolTime)
{
    port out bool tdx;
    port in bits!8 enq;
    port in bool valid;
    port out bool ready;

    reg bits buffer = 0b111111111;
    reg uint state = 10;
    reg uint counter = symbolTime;

    txd := buffer[0];

    ...  

    ready := state == 0;
}
if(state == 0) {
    if(valid) {
        buffer := enq ~ 0;
        counter := time;
        state := 10;
    }
} else {
    if(count == 0) {
        buffer := 1 ~ buffer[9..1];
        counter := time;
        state := state-1;
    } else {
        counter := counter-1;
    }
}
circuit UARTTx(int symbolTime)
{
    port out bool tdx;
    port in bits!8 enq;
    port in bool valid;
    port out bool ready;

    reg bits buffer = 0b111111111;
    reg uint state = 10;
    reg uint counter = symbolTime;

    txd := buffer[0];

    ...

    ready := state == 0;
}
circuit UARTTx(int symbolTime)
{
    port out bool tdx;
    port in Decoupled!(bits!8) enq;

    reg bits buffer = 0b11111111;
    reg uint state = 10;
    reg uint counter = symbolTime;

    txd := buffer[0];

    ...

    enq.ready := state == 0;
}
if(state == 0) {
    if(enq.valid) {
        buffer := enq.payload ~ 0;
        counter := time;
        state := 10;
    }
} else {
    if(count == 0) {
        buffer := 1 ~ buffer[9..1];
        counter := time;
        state := state-1;
    } else {
        counter := counter-1;
    }
}
IO - UART

```dhdl
port UARTPeripheral
{
    port in rxd;
    port out txd;
}

port UARTCtl
{
    port in Decoupled!(bits!8) enq;
    port out Decoupled!(bits!8) deq;
}

circuit UART(int symbolTime)
{
    port in UARTCtl ctl;
    port in UARTPeripheral pins;
    auto tx = new UARTTx!symbolTime;
    auto rx = new UARTRx!symbolTime;
    tx.txd <> pins.txd
    // ...
}
```
circuit TestSerial
{
    port in UARTPeripheral uartPins;

    auto uart = new UART!230400;
    uart.ctl.enq.valid := true;
    uart.ctl.enq.payload := 'D';
    uartPins.txd = uart.pins.txd;
}
IO - UART
IO - 7 segment
FPGA IO - 7 segment

```
sevensegment.dhdl

switch(digit)
{
    case 0x0: pins.segments := 0b1000000; break;
    case 0x1: pins.segments := 0b1111001; break;
    case 0x2: pins.segments := 0b0100100; break;
    case 0x3: pins.segments := 0b0110000; break;
    case 0x4: pins.segments := 0b0011001; break;
    case 0x5: pins.segments := 0b0010010; break;
    case 0x6: pins.segments := 0b0000010; break;
    case 0x7: pins.segments := 0b1111000; break;
    case 0x8: pins.segments := 0b0000000; break;
    case 0x9: pins.segments := 0b0011000; break;
    case 0xA: pins.segments := 0b0001000; break;
    case 0xB: pins.segments := 0b0000111; break;
    case 0xC: pins.segments := 0b1000110; break;
    case 0xD: pins.segments := 0b0100001; break;
    case 0xE: pins.segments := 0b0000110; break;
    case 0xF: pins.segments := 0b0001110; break;
}
```
FPGA IO - 7 segment

sevensegment.dhdl

switch(activeDigit)
{
    case 0: pins.anodes := 0b00000001; break;
    case 1: pins.anodes := 0b00000010; break;
    case 2: pins.anodes := 0b00000100; break;
    case 3: pins.anodes := 0b00001000; break;
    case 4: pins.anodes := 0b00010000; break;
    case 5: pins.anodes := 0b00100000; break;
    case 6: pins.anodes := 0b01000000; break;
    case 7: pins.anodes := 0b10000000; break;
}

FPGA IO - 7 segment

sevensegment.dhdl

switch(activeDigit)
{
    case 0: pins.anodes := 0b00000001; break;
    case 1: pins.anodes := 0b00000010; break;
    case 2: pins.anodes := 0b00000100; break;
    case 3: pins.anodes := 0b00001000; break;
    case 4: pins.anodes := 0b00010000; break;
    case 5: pins.anodes := 0b00100000; break;
    case 6: pins.anodes := 0b01000000; break;
    case 7: pins.anodes := 0b10000000; break;
}
IO - VGA
IO - VGA

[Diagram of VGA connector with labels for each pin:]
- red[0] (1)
- red[1] (2)
- red[2] (3)
- green[0] (4)
- green[1] (5)
- green[1] (6)
- blue[0] (7)
- blue[1] (8)
- vert. sync (9)
- hor. sync or C. S. (10)
- ground (GND) (11)

[Resistance values:]
- 2k
- 1k
- 510
- 820
- 390
- 82

[Nodes:]
- Node 1
- Node 2
- Node 3
- Node 4
- Node 5
- Node 6
- Node 7
- Node 8
- Node 9
- Node 10
- Node 11
- Node 12
- Node 13
IO - VGA

Visible area
IO - Audio
IO - Audio

• Delta-sigma DAC

```dhdsl
port out bool audio;
reg uint!8 sample;
reg uint!9 deltaSigma;
deltaSigma := deltaSigma + sample;
audio := deltaSigma[8];
```
Demo

• Let’s build an arcade with these components
  • What game should we use?
  • Something somewhat related to D…
Demo

```
C:\arcadedbg>arcadedbg.exe --elf
WRITE 1000 - 1F8F0 (1000 - 1F8F0)
WRITE *
WRITE 20000 - 2BCB0 (20000 - 2BCB0)
WRITE *
CLEAR 2BCB0 - 34F5C
WRITE *
WRITE 35000 - 13E800 (2C000 - 135800)
WRITE *
```

Image of a demo showing a hexadecimal number displayed on an LED display.
Demo
Demo

- Not shown for lack of time:
  - SDRAM Controller
  - RISC-V CPU
  - Cache

- LDC2 compiled with RISC-V backend
  - Lots of issues here…
Thank you