USING A CUSTOM-BUILT HDL FOR PRINTED CIRCUIT BOARD DESIGN CAPTURE

Brent Nelson
Brad Riching
Josh Mangelson
Dept. of Electrical and Computer Engineering
Brigham Young University

PCB West Presentation
September 2012
Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy’s National Nuclear Security Administration under contract DE-AC04-94AL85000. SAND Number : 2012-6638 C
Who Are We?

• Dr. Brent Nelson (Professor)
• Brad Riching (MS student)
• Richard Black (BS student)
• Joshua Mangelson (BS student)

– Dept. of Electrical and Computer Engineering
– Brigham Young University
Overview

• Why HDL’s for PCB design capture?

• PHDL – An HDL for PCB design capture
  – *The language*
  – *The tool flow*

• Examples of PHDL board designs

• Examples of 3rd party support tools and utilities

• Links to PHDL open source community

• Future Work
Part 1

MOTIVATION: HDL’S FOR PCB DESIGN CAPTURE
Graphical Schematic Entry

• Imitates manual drawing
  – Intuitive
  – Spatial information + human visual system  → understanding
  – Current industrial practice

• Would seem to be the best method!

  QED…
HDL’s for PCB Design

- **Design capture**
  - Capture designer’s intent
  - Define design components and connectivity
  - Currently done with schematic tools

- **Physical design**
  - Physical design rules
  - Electrical considerations (voltage, current, capacitance, inductance)
  - Best done using current layout tools & methods
Note: this entire page is mainly the symbol for a single 784 pin FPGA.

It is one of 20-30 pages in the overall design.
Drawing a Schematic Net

• subroutine drawSchematicNet() {
  foreach pin in net {
    1. pan to page and area of pin
    2. zoom in on pin
    3. attach wire stub to pin
    4. add textual name to wire stub
    5. zoom back out
    6. repeat
  }
}

• How many mouse clicks per pin?

• How many nets in a design?

• How do you do a design review?
  – From printed schematics…
  – From a schematic design tools…

So, just how again is this graphical approach helping me?
Design Change Tracking

• Design changes between two versions?

```bash
% diff a.sch b.sch
Binary files a.sch and b.sch differ
%
```

That’s not very helpful...

Need a mechanism for comparing design versions and documenting individual design changes.

Source code control systems (SCCS’s) have done this for years for software.
Design Sharing

• Sharing a design amongst a team
  – Divide and conquer

• Design reuse
  – How do you reuse portions of a schematic between designs?
Schematic Tools: Proprietary Source

• Schematic design files:
  – Protected, binary formats
  – Specialized tools just to view source…
    • Edit, manipulate, analyze?

• If you have the wrong version of proprietary CAD tool…
  – Cannot even open the schematic to view

• Discourages/prevents 3rd party tools
INTRO TO PHDL
What is PHDL?

Schematic Capture produces netlist

PCB Layout interprets netlist

Bill of Material, other ancillary data
What is PHDL?

Schematic Capture produces netlist

PCB Layout interprets netlist

PHDL Source Code

PHDL compiler

Bill of Material, other ancillary data
PHDL: A First Example

- Our example circuit:
PHDL: A First Example

- Four types of devices

- Plus some wires
device Resistor {

  attr REPREFIX = "R";
  attr FOOTPRINT = "M0805";
  attr LIBRARY = "complib";
  attr VALUE = "120";
  pin a = {1};
  pin b = {2};
}

// Comments are allowed like this.

// Compiler will auto-assign full refDes's
device SevenSeg {
    attr REFPREFIX = "LD";
    attr FOOTPRINT = "SA08-21";
    attr LIBRARY = "myLib";

    // Multi-bit pins are allowed
    pin[1:8] segments =
        {2,15,13,11,
         5,3,14,10};
    pin[2:0] anode = {4,12,17};
}

Defining Devices: Multi-Bit Pins

SA08-21
// Define the design and give it a name

design sevenSeg {

// Define the wires
net gnd, vcc;

// Define a multi-bit wire
net[1:8] segs, r2sw;

// Define the design and give it a name
design sevenSeg {

    // Define the wires
    net gnd, vcc;

    // Define a multi-bit wire
    net[1:8] segs, r2sw;

    R1 120 S1
    R2 120 S2
    R3 120 S3
    R4 120 S4
    R5 120 S5
    R6 120 S6
    R7 120 S7
    R8 120 S8
// Define the design and give it a name
design sevenSeg {

    // Define the wires
    net gnd, vcc;

    // Define a multi-bit wire
    net[1:8] segs, r2sw;
}
// Define the design and give it a name
design sevenSeg {

    // Define the wires
    net gnd, vcc;

    // Define a multi-bit wire
    net[1:8] segs, r2sw;

    R1 120  S1
    R2 120  S2
    R3 120  S3
    R4 120  S4
    R5 120  S5
    R6 120  S6
    R7 120  S7
    R8 120  S8

    LD1
    SA08-21
    G1
    A B C D E F G DP
    2 15 13 11
    3 10 14 12
    5
    +
    -
// Define the design and give it a name
design sevenSeg {

    // Define the wires
    net gnd, vcc;

    // Define a multi-bit wire
    net[1:8] segs, r2sw;
}
inst source of Battery {
    pos = vcc;
    neg = gnd;
}

// Create an instance of the
// "Battery" device
// Wire it up:
// Port "pos" -> net "vcc"
// Port "neg" -> net "gnd"
inst segment of SevenSeg {
segments = segs;
anode = <vcc>;
}

// Instance the seven
// segment chip

// Tie pins segments[1:8] ->
// net segs[1:8]

// Tie all the “anode” pins
// to “vcc”.

Creating The Design: Instancing Devices
Creating The Design: Instance Array

```plaintext
inst(1:8) swArray of Switch {
    combine(a) = r2sw;
    combine(b) = segs;
}
```

// Make an array of 8
// “Switch” instances

// The instances will be
// numbered from 1 to 8

// For arrays, use (...)’s

// For nets and pins,
// use [...]’s
inst(1:8) swArray of Switch {
  combine(a) = r2sw;
  combine(b) = segs;
}

// Take all the “a” pins,
// combine them left to right
// into a bus, and wire them
// to the “r2sw” net

// Thus,
swArray(1:8).a -> r2sw[1:8]
inst(1:8) swArray of Switch {  
    combine(a) = r2sw;  
    combine(b) = segs;  
}

// Take all the “a” pins,  
// combine them left to right  
// into a bus, and wire them  
// to the “r2sw” net  

// Thus,  
swArray(1:8).a -> r2sw[1:8]
Creating a Design: Setting Inst Attributes

\[
\begin{align*}
\text{inst} (1:8) & \text{ rArray of Resistor} \\
& \{ \\
& \quad \text{this}(1:4) . \text{VALUE} = "100"; \\
& \quad \text{this}(8) . \text{VALUE} = "75"; \\
& \quad \text{combine}(a) = \text{r2sw}; \\
& \quad b = \text{gnd}; \\
& \}
\end{align*}
\]

// Set some "value" attributes to "100".
// Set one "value" attribute to "75".

// Remember, the Resistor device has a default "value" of "120"
inst(1:8) rArray of Resistor
{
    VALUE = "120";
    combine(a) = r2sw;
    b = gnd;
}

// Take each "b" pin and
// individually tie it to
// the "gnd" net.
device Resistor {
    attr REFPREFIX = "R";
    attr FOOTPRINT = "M0805";
    attr LIBRARY = "complib";
    attr VALUE = "120";
    pin a = {1};
    pin b = {2};
}

device Switch {
    attr REFPREFIX = "SW";
    attr FOOTPRINT = "MS243";
    attr LIBRARY = "complib";
    pin a = {1};
    pin b = {2};
}

device Battery {
    attr REFPREFIX = "G";
    attr FOOTPRINT = "1V60R";
    attr LIBRARY = "complib";
    attr VALUE = "9V";
    pin pos = {2};
    pin neg = {1};
}

device SevenSeg {
    attr REFPREFIX = "LD";
    attr FOOTPRINT = "SA08-21";
    attr LIBRARY = "myLib";
    pin[1:8] segments =
        {2, 15, 13, 11, 5, 3, 14, 10};
    pin[1:3] anode = {4, 12, 17};
}
design ssControl {
    net gnd, vcc;
    net[1:8] segs, r2sw;
    inst source of Battery {
        pos = vcc;  neg = gnd;
    }
    inst segment of SevenSeg {
        segments = segs;
        anode = <vcc>;
    }
    inst(1:8) swArray of Switch {
        combine(a) = r2sw;
        combine(b) = segs;
    }
    inst(1:8) rArray of Resistor {
        this(1:7).VALUE = "100";
        this(8).VALUE = "75";
        combine(a) = r2sw;
        b = gnd;
    }
}
Compilation Flow

$java -jar phdlcomp.jar srcFolder [switches]

PHDL Compiler

Output files

- Netlist
- Bill of Materials
- Component List
- Layout Directions
- XML
- Tool-specific Scripts

Eclipse flow: Compiler runs every time you save your design.
A Netlist

!PADS-POWERPCB-V9.0-MILS! NETLIST FILE FROM PADS LOGIC V9.3

*PART*
G1 complib@1V60R
LD1 complib@MS243
R1 complib@M0805
R2 complib@M0805
R3 complib@M0805
R4 complib@M0805
R5 complib@M0805
R6 complib@M0805
R7 complib@M0805
R8 complib@M0805
SW1 complib@MS243
SW2 complib@MS243
SW3 complib@MS243
SW4 complib@MS243
SW5 complib@MS243
SW6 complib@MS243
SW7 complib@MS243
SW8 complib@MS243

!*
*CONNECTION*
*SIGNAL* GND
G1.1 R1.2
R1.2 R2.2
R2.2 R3.2
R3.2 R4.2
R4.2 R5.2
R5.2 R6.2
R6.2 R7.2
R7.2 R8.2
*SIGNAL* R2SW[1]
SW1.1 R1.1
*SIGNAL* R2SW[2]
SW2.1 R2.1
*SIGNAL* R2SW[3]
SW3.1 R3.1
*SIGNAL* R2SW[4]
SW4.1 R4.1
*SIGNAL* R2SW[5]
SW5.1 R5.1
*SIGNAL* R2SW[6]
SW6.1 R6.1
*SIGNAL* R2SW[7]
SW7.1 R7.1
*SIGNAL* R2SW[8]
SW8.1 R8.1

*SIGNAL* SEGS[1]
LD1.2 SW1.2
*SIGNAL* SEGS[2]
LD1.15 SW2.2
*SIGNAL* SEGS[3]
LD1.13 SW3.2
*SIGNAL* SEGS[4]
LD1.11 SW4.2
*SIGNAL* SEGS[5]
LD1.15 SW5.2
*SIGNAL* SEGS[6]
LD1.13 SW6.2
*SIGNAL* SEGS[7]
LD1.14 SW7.2
*SIGNAL* SEGS[8]
LD1.10 SW8.2
*SIGNAL* VCC
G1.2 LD1.4
LD1.4 LD1.12
LD1.12 LD1.17

*END*
A Bill of Materials

• comma-separated file
  – *Import into Excel to view*

QUANTITY, NAME, REFDES, LIBRARY, FOOTPRINT, VALUE
1, Battery, G1, complib, 1V60R, 9V
1, SevenSeg, LD1, complib, MS243,
8, Switch, SW1; SW2; SW3; SW4; SW5; SW6; SW7; SW8, complib, MS243,
8, Resistor, R1; R2; R3; R4; R5; R6; R7; R8, complib, M0805, 100
Targeted Design Flows

Mentor Graphics PADS

EAGLE PCB

Others coming soon…
Part 3

PHDL: DIGGING A LITTLE DEEPER...
Array indexing

- Can instantiate using any indexing desired
- Uses notion of left-to-right ordering

```
net[1:8] segs;
// Leftmost wire → “segs[1]”
// Rightmost wire → “segs[8]”

inst(1:8) swArray of Switch {
    ...
}
// Leftmost Switch → “this(1)”
// Rightmost Switch → “this(8)”

<OR>

inst(7:0) swArray of Switch {
    ...
}
// Leftmost Switch → “this(7)”
// Rightmost Switch → “this(0)”
```
```
net gnd;
net[1:8] net1, net2;

inst(1:8) res of Resistor {
   combine(b) = net1;
}
```
net gnd, vcc;
net[1:8] net1, net2;

inst(1:8) res of Resistor {
    this(1:4).a = gnd;
    this(5:8).a = vcc;
    combine(this(1, 3, 5, 7).b) = net1[7, 5, 3, 1];
    combine(this(8, 2, 4, 6).b) = net2[1:4]
}

// In all cases, indexing
// is viewed left-to-right
net n1, n2, x1, x2;

inst(1:8) res of Resistor { 
    combine(this(1:4).b) = 
    n1 & x2 & x1 & n2;
}

// The RHS of the above assignment
// is a "concatenation"
PHDL Packages (1)

- Declare devices to be in a package
- Must use package name when instantiating.
- Allows same device in multiple device library files without name collision.

```phdl
package myParts {
    device Resistor {
        attr REFPREFIX = "R";
        ...
    }
}

design sevenSeg {
    inst(1:8) rArray of myParts.Resistor {
        ...
    }
}
```
PHDL Packages (2)

- Declare devices to be in a package
- Import the package contents.
- Avoids having to use qualified names.
Subdesigns have port definitions. Subdesigns can be instanced like a device but with ‘subinst’ keyword. Subdesigns can be array instanced just like devices.
You can reach down into hierarchy using “.” notation to change lower level attributes. No limit to levels deep you can go.
Can make array of subdesigns...

```phdl
subdesign rc {
    port gnd, in, out;

    inst res of Resistor {
        a = in;
        b = out;
    }

    inst cap of Capacitor {
        pos = out;
        neg = gnd;
    }
}
```

design myCircuit {
    net[0:1] i, o;
    net gnd, vcc;

    subinst(0:1) rcl of rc {
        combine(in) = i;
        combine(out) = o;

        gnd = gnd;

        this(0).res.VALUE = "66";
        this(1).res.VALUE = "100";
    }

    inst P1 of Connector {
        p[0:2] = i[0] & gnd & o[1];
        p[3] = open;
    }
}
Part 4

A REAL PHDL BOARD: FPGA-BASED MOTOR CONTROL
FPGA-based motor controller (2-axes)

- *Spartan3 400K 144-pin QFP implements:*
  - 32-bit position, vel. and accel. registers per axis
  - Programmable PID filters, sampling intervals
  - Trapezoidal velocity profile generators
  - Packet router over RS232 to host PC application

- *Supporting hardware*
  - 500+ CPR encoder feedback resolution
  - PWM brushless and brushed motor motor drives
  - The usual JTAG, Flash ROM, GPIO, etc.
Motor Controller Board

Final Layout

Design Entry:

~1200 lines of PHDL

30 devicedecls

672 nets
Motor Controller Board

Back From Manufacturing
Motor Controller Board

Assembled

Top

Bottom
Part 5

SOURCE CODE CONTROL SYSTEMS (SCCS)
Motivation

• SCCS provides a remote repository
  – Collaborate between users
  – Saves all design versions
    • Document every design change
    • Compare versions
  – Tag release file sets

• CVS and SVN are commonly used
  – Command line versions for Linux
  – GUI programs for Windows (ex: Tortoise)
The Old Binary File Way

- PowerSupp.sch
- PowerSupp_new.sch
- PowerSupp_newer.sch
- PowerSupp_june15.sch
- PowerSupp_v1.3

Each is a different version of the design.

No enforced naming or numbering system.

User must keep track of them.

- What changed between two versions?

```bash
% diff a.sch b.sch
Binary files a.sch and b.sch differ
%
```

Not very helpful... How do we know what really changed?
The SCCS Way (CVS)

% cvs diff
cvs diff: Difffing .
Index: a.phdl
=============================================
RCS file: /fpga2/cvsroot/users/nelson/test/a.phdl,v
retrieving revision 1.2
diff -r1.2 a.phdl
31c31
<   attr FOOTPRINT = "1V60R-5";
---
>   attr FOOTPRINT = "1V60R";

This shows that line 31 has changed.
Comparison between local copy and most recent archived version (v1.2).
Can compare any two arbitrarily chosen versions.
This shows that line 31 has changed.
THE ECLIPSE PLUG-IN FOR PHDL
Eclipse PHDL Plug-In

The files in my project

The file being edited

Syntax coloring helps understand structure and find simple errors (ex: no closing quote on a string)

An outline of the structure of the project
Real-Time Syntax Checking (1)

Error mark shows up instantly

Mouse hover gives popup error message
When possible, IDE proposes “Quick Fixes”.

```
inst(1:8) rArray of Resistor {
    VALUE = "100";
    combine(a) = r2sw;
    b = gnd;
}
```

Key click brings up context-specific content suggestions.
Template has been inserted.

Tab between fields to fill it in.

In body of subinstance, will suggest what you can do based on what has been defined thus far in the project and insert template if selected.
Hovering over a named element will give its definition

CTRL-clicking it will take you to that definition, even if in another file
Plug-In: Possible Features

• Click on a signal definition
  – Highlight every place in code it is wired

• Design hierarchy browsing

• Design visualization
  – Net extents
  – Localized connectivity
Integration with SCCS (SVN)

Project navigator screen shows which files are out of date with repository.

Clicking takes you to repository synchronization screen
SVN Synchronize Screen

Click icons to check files into repository

List of files that have been modified

Double click a file to bring up side-by-side comparison window
This shows that line 31 has changed.
Part 7

3rd PARTY TOOLS
Example: FPGA Pin Generation (csv2phdl)

FPGA VHDL Design

```vhdl
library ieee;
use ieee.numeric_std.all;
use ieee.std_logic_1164.all;
entity fpga is
  port(
    clk : in std_logic;
    rst : in std_logic;

    -- RS232 serial ports
    rxd : in std_logic;
    txd : out std_logic;
    rxd_a : in std_logic;
    txd_a : out std_logic;

    -- 12-bit DAC
    sclk : out std_logic;
    sync : out std_logic;
    sdata : out std_logic
  );
  data : out std_logic_vector(7 downto 0)
end entity fpga;
```

Synthesis, PAR, csv2phdl

```
library ieee;
use ieee.numeric_std.all;
use ieee.std_logic_1164.all;
entity fpga is
  port(
    clk : in std_logic;
    rst : in std_logic;

    -- RS232 serial ports
    rxd : in std_logic;
    txd : out std_logic;
    rxd_a : in std_logic;
    txd_a : out std_logic;

    -- 12-bit DAC
    sclk : out std_logic;
    sync : out std_logic;
    sdata : out std_logic
  );
  data : out std_logic_vector(7 downto 0)
end entity fpga;
```

PHDL Device Declaration

```
device fpga is
  attr REFPREFIX = "U";
  attr FOOTPRINT = "tq144";
  attr LIBRARY = "XILINX";
  attr mfgr = "XILINX";
  attr partNumber = "xc3s400-4tq144";

  // User I/O pins.
  pin clk = {P52};
  pin rst = {P40};
  pin rxd = {P47};
  pin rxd_a = {P41};
  pin txd = {P46};
  pin txd_a = {P44};
  pin sclk = {P86};
  pin sdata = {P87};
  pin sync = {P85};
  pin[7:0] data = {P23,P21,P20,P18...};
end;
```

Location Constraints

Thanks to Pete Dudley...
Example: Automatic Device Generation (DeviceGen)

- Eagle device files are in XML format
  - 3rd party Java GUI program
  - Easily browse, select, convert to PHDL

Name of Eagle library

Browsable list of devices and packages in library

List of selected devices and packages for PHDL device generation

Click to generate PHDL device declarations

Thanks to Richard Black…
Part 8

PHDL IS OPEN SOURCE AND AVAILABLE
What is PHDL?

PHDL is an open-source research-based Hardware Description Language (HDL) created at Brigham Young University that models text-based schematics for Printed Circuit Boards (PCBs). Its primary objectives for increasing designer productivity are to overcome limitations of scale, collaboration, and design reuse inherent with graphical schematic capture CAD tools.

PHDL now has a plugin for Eclipse as well as a command-line only option for users who wish to use their own editor. However, it is strongly advised for new users to start with the plugin. The many features available from within an IDE such as eclipse (syntax highlighting, content assist, code completion) generally reduce the learning curve with any new language.

PHDL at PCB West 2012

Posted on June 30, 2012

PHDL will be discussed and demonstrated at PCB West this year. If you are going to PCB West, be sure to come visit us!

Eclipse plugin coming July 2012

Posted on June 30, 2012

PHDL is currently being integrated into the Eclipse IDE framework with the help of Xtext, showcasing many of the popular benefits inherent with Integrated Development Environments. Support for packages and namespaces has been added to the language specification.

Getting Started With PHDL

The best place to start is to visit our installation instructions which will help you get PHDL up and running on your machine. Then, be sure to visit the tutorial page.

PHDL Screenshots
What is PHDL?
PHDL is an open-source research-based Hardware Description Language (HDL) created at Brigham Young University that models text-based schematics for Printed Circuit Boards (PCBs). Its primary objectives for increasing designer productivity are to overcome limitations of scale, collaboration, and design reuse inherent with graphical schematic capture CAD tools.

PHDL now has a plugin for Eclipse as well as a command-line only option for users who wish to use their own editor. However, it is strongly advised for new users to start with the plugin. The many features available from within an IDE such as eclipse (syntax highlighting, content assist, code completion) generally reduce the learning curve with any new language.

PHDL at PCB West 2012
Found on June 30, 2012

PHDL will be discussed and demonstrated at PCB West this year. If you are going to PCB West, be sure to come visit us!

Eclipse plugin coming July 2012
Found on June 30, 2012

PHDL is currently being integrated into the Eclipse IDE framework with the help of Xiext, showcasing many of the popular benefits inherent with Integrated Development Environments. Support for packages and namespaces has been added to the language specification.

Getting Started With PHDL
The best place to start is to visit our installation instructions which will help you get PHDL up and running on your machine. Then, be sure to visit the tutorial page.

PHDL Screenshots
Acknowledgements

• Sandia National Laboratories
  – Supported the work
  – Provided technical direction and management
  – Chuck Graham and Wes Landaker

• Pete Dudley – proposed PHDL
  – Formerly of Sandia, now of hdlguy.com, an FPGA & PCB board design consultancy
  – Authored the csv2phdl tool
Part 8

FUTURE WORK
Future Tasks

• Tool & library integration
• Hierarchical refdes generation
  – Provide natural grouping mechanism
• Connectivity ERC
• Design visualization tools
  – Hierarchy browsing
  – Cross-probing
  – Graphical viewing