Survey of Domain-Specific Languages for FPGA Computing

Nachiket Kapre
nachiket@ieee.org
Expressiveness (Freedom)

Some goodness metric
Expressiveness (Freedom)

Some goodness metric

Trump’s attack on judge
Singapore’s contempt of court bill

Singapore: Contempt of court bill is a threat to freedom of expression


Donald Trump’s hate-filled rhetoric & bigoted scapegoating flies in the face of equality & MUST be rejected.

https://twitter.com/amnesty/status/674053786520915969

Trump’s attack on judge
Expressiveness (Freedom)

Some goodness metric

Classic HDLs
Expressiveness (Freedom)

Some goodness metric

DSLs

Classic HDLs
Intel unveils new Xeon chip with integrated FPGA, touts 20x performance boost

By Sebastian Anthony on June 19, 2014 at 1:19 pm | Comment

MINISTRY OF INNOVATION / BUSINESS OF TECH

Intel will acquire FPGA maker Altera for $16.7 billion
Consolidation continues in the semi industry with Avago-Broadcom, now this.

by Sebastian Anthony (UK) - Jun 1, 2015 9:28pm CST
Outline

• Review of FPGA Design Flow
  — Where we stand?
  — Need for DSLs

• Classification of DSLs

• Code Vignettes

• Experimental Results
Outline

• Review of FPGA Design Flow
  — Where we stand?
  — Need for DSLs

• Classification of DSLs

• Code Vignettes

• Experimental Results
FPGA flow

- FPGA flow longer, more complex
- **Problem 1:** Write low-level Verilog code
- **Problem 2:** Wait hours to compile (adds insult to injury)
- **Problem 3:** Long verification feedback cycles.
Example code sketches

```
module poly(
    input clk, rst,
    input [31:0] x,
    output reg [31:0] y);

    reg [31:0] a=3, b=2, c=1;

    always @ (posedge clk)
    begin
        if(rst)
            y = 32'b0;
        else
            y = a*x*x + b*x + c;
    end

endmodule
```
Example code sketches

```c
void poly(int x, int* y) {
    int a=3, b=2, c=1;
    *y = a*x*x + b*x + c;
}
```
What's different?

- What makes the C code smaller?
- Clocking/Reset?
- Explicit pipelining
- Type information — registers, wires, number of bits
Simple forms of parallelism
Simple forms of parallelism
Limits of OpenCL/HLS

• One alternative to HDLs — OpenCL/HLS flow

• Restricted subset of C (no pointers, no complex data sharing) —> sacrifice freedom for speed

• Drawbacks:
  — Overheads due to implicit assumptions
    — more area, slower design, not fully optimised
  — Only really addresses time-to-compilation
    — still need to do synth + P&R
Outline

• Review of FPGA Design Flow
  — Where we stand?
  — Need for DSLs

• Classification of DSLs

• Code Vignettes

• Experimental Results
Domain-Specific Languages

• “Beauty lies in the eye of the beholder”

• Conventional “application-domain” view — finance, HPC, radio, multimedia, networking, databases, security.

• Suggest two alternate views in this paper...
Axes of classification

• (1) Conventional “application-domain” view
  — focus on end-user of FPGA technology

• (2) “compute-model” view
  — analogous to Berkeley’s Ptolemy classification

• (3) “design” view
  — behind-the-scenes tinkerers, library developers, system builders, academics
Axes of classification

- (1) Conventional "application-domain" view
  — focus on end-user of FPGA technology

- (2) "compute-model" view
  — analogous to Berkeley’s Ptolemy classification

- (3) "design" view
  — behind-the-scenes tinkerers, library developers, system builders, academics
Application Domain

Networking
- SNORT
- Click
  - G
  - SQL

Databases
- Machine Learning
  - OptiML

Numerics
- Verilog
  - AMS

Signal Processing
- HDLCoder
- SPIRAL
- LabView
- VSIPL
Outline

• Review of FPGA Design Flow
  — Where we stand?
  — Need for DSLs

• Classification of DSLs

• Code Vignettes

• Experimental Results
Matlab HDL Coder

function [y] = poly (x)
    a=3; b=2; c=1;
    y=a*x*x+b*x+c;

class Poly extends Kernel {
    Poly(KernelParameters parameters) {
        super(parameters);
        DFEVar x = io.input("x", dfeUInt(32));
        int a = 3, b = 2, c = 1;
        DFEVar y = a*x*x + b*c + c;
        io.output("y", y, dfeUInt(32));
    }
}
poly(input unsigned[32] x,
       output unsigned[32] y)
{
    unsigned[32] a=3, b=2, c=1;

    state always (x):
        y = a*x*x*x + b*x + c;
}

using PA=Microsoft.ParallelArrays.ParallelArrays;
	namespace Poly
{

class Program
{

static void Main(string[] args)
{
int N = 1024;
int a = 3, b = 2, c = 1;

int[] xArr = new int[N];
int[] yArr = new int[N];

FPGATarget t = new FPGATarget();

PA x = new PA(xArr);

PA t1 = PA.Multiply(a, x);
PA t2 = PA.Multiply(t1, x);
PA t3 = PA.Multiply(b, x);
PA t4 = PA.Add(t3, t2);
PA t5 = PA.Add(t4, c);

yArr = t.ToArray1D(t5);
}
}
}
public class Poly extends Logic {

    // Interface
    public static CellInterface[] cif = {
        in("x", 18), out("y", 36),
    };

    // Constructor
    public Poly(Node parent, Wire y, Wire x) {

        // Connect wires
        connect("y", y);
        connect("x", x);

        // Build our logic
        new mult18x18(this, x, x, t1);
        new mult18x18(this, t1, a, t2);
        new mult18x18(this, b, x, t3);
        new adder(this, t2, t3, cin, t4, cout);
        new adder(this, t4, c, cin, y, cout);
    }
}
class Poly extends Component {
    val io = new Bundle {
        val a = Bits(32, INPUT)
        val b = Bits(32, INPUT)
        val c = Bits(32, INPUT)
        val x = Bits(32, INPUT)
        val y = Bits(32, OUTPUT)
    }

    io.y := io.a * io.x * io.x +
             io.b * io.x + io.c
}

CHISEL
Outline

• Review of FPGA Design Flow
  — Where we stand?
  — Need for DSLs

• Classification of DSLs

• Code Vignettes

• Experimental Results
Experimental Evaluation

• NTU MSc Embedded Systems cohort
  — Class of 2014-15
  — ~25-30 students

• 3-4 students per DSL

• One 4hr lab session devoted to working on the $ax^2+bx+c$ mapping example
<table>
<thead>
<tr>
<th>DSL</th>
<th>Dev. Time</th>
<th>Lines of Code</th>
<th>Resources</th>
<th>Freq. MHz</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>DSL</td>
<td>RTL</td>
<td>LUTs</td>
</tr>
<tr>
<td>Flopoco(^1)</td>
<td>30m</td>
<td>2</td>
<td>1702</td>
<td>1679</td>
</tr>
<tr>
<td>Maxeler (baseline)</td>
<td>30m 30m</td>
<td>15</td>
<td>NA(^2)</td>
<td>6036</td>
</tr>
<tr>
<td>Vivado HLS</td>
<td>1h</td>
<td>4</td>
<td>92</td>
<td>53</td>
</tr>
<tr>
<td>Lime (baseline)</td>
<td>2h30m 2h30m</td>
<td>22</td>
<td>111</td>
<td>245</td>
</tr>
<tr>
<td>OpenCL(^3) (baseline)</td>
<td>2h30m 2h30m</td>
<td>4</td>
<td>1262</td>
<td>3281</td>
</tr>
<tr>
<td>Chisel</td>
<td>3h</td>
<td>25</td>
<td>39</td>
<td>129</td>
</tr>
<tr>
<td>OpenDF</td>
<td>3h30m</td>
<td>26</td>
<td>689</td>
<td>171</td>
</tr>
<tr>
<td>JHDL</td>
<td>4h</td>
<td>40</td>
<td>2529(^4)</td>
<td>41</td>
</tr>
<tr>
<td>SCORE</td>
<td>4h</td>
<td>7</td>
<td>111</td>
<td>139</td>
</tr>
</tbody>
</table>

\(^1\) Flopoco only provides floating-point support for these expressions
\(^2\) MaxCompiler does not produce any intermediate RTL, directly generates executable bitstreams
\(^3\) Altera resources measured in LEs instead of LUTs, Altera 18×18 DSPs are also different from Xilinx 25×18 DSPs
\(^4\) JHDL directly generates a circuit netlist in EDF format instead of generating RTL
TABLE I: Comparing DSLs with $ax^2 + bx + c$ mapping

<table>
<thead>
<tr>
<th>DSL</th>
<th>Dev. Time</th>
<th>Lines of Code</th>
<th>LUTs</th>
<th>FFs</th>
<th>DSPs</th>
<th>Freq. MHz</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flopoco$^1$</td>
<td>30m</td>
<td>2</td>
<td>1702</td>
<td>1679</td>
<td>1288</td>
<td>91</td>
</tr>
<tr>
<td>Maxeler (baseline)</td>
<td>30m</td>
<td>15</td>
<td>NA$^2$</td>
<td>6036</td>
<td>5391</td>
<td>3</td>
</tr>
<tr>
<td>Vivado HLS</td>
<td>1h</td>
<td>4</td>
<td>92</td>
<td>53</td>
<td>71</td>
<td>3</td>
</tr>
<tr>
<td>Lime (baseline)</td>
<td>2h30m</td>
<td>22</td>
<td>111</td>
<td>245</td>
<td>284</td>
<td>2</td>
</tr>
<tr>
<td>OpenCL$^3$ (baseline)</td>
<td>2h30m</td>
<td>4</td>
<td>1262</td>
<td>3281</td>
<td>4443</td>
<td>2</td>
</tr>
<tr>
<td>Chisel</td>
<td>3h</td>
<td>25</td>
<td>39</td>
<td>129</td>
<td>64</td>
<td>10</td>
</tr>
<tr>
<td>OpenDF</td>
<td>3h30m</td>
<td>26</td>
<td>689</td>
<td>171</td>
<td>305</td>
<td>9</td>
</tr>
<tr>
<td>JHDL</td>
<td>4h</td>
<td>40</td>
<td>2529$^4$</td>
<td>41</td>
<td>90</td>
<td>3</td>
</tr>
<tr>
<td>SCORE</td>
<td>4h</td>
<td>7</td>
<td>111</td>
<td>139</td>
<td>245</td>
<td>2</td>
</tr>
</tbody>
</table>

$^1$Flopoco only provides floating-point support for these expressions.
$^2$MaxCompiler does not produce any intermediate RTL, directly generates executable bitstreams.
$^3$Altera resources measured in LEs instead of LUTs, Altera 18×18 DSPs are also different from Xilinx 25×18 DSPs.
$^4$JHDL directly generates a circuit netlist in EDIF format instead of generating RTL
<table>
<thead>
<tr>
<th>DSL</th>
<th>Dev. Time</th>
<th>Lines of Code</th>
<th>Resources</th>
<th>Freq. MHz</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flopoco¹</td>
<td>30m</td>
<td>2</td>
<td>1702</td>
<td>1679</td>
</tr>
<tr>
<td>Maxeler (baseline)</td>
<td>30m</td>
<td>15</td>
<td>NA²</td>
<td>6036</td>
</tr>
<tr>
<td>Vivado HLS</td>
<td>1h</td>
<td>4</td>
<td>92</td>
<td>53</td>
</tr>
<tr>
<td>Lime (baseline)</td>
<td>2h30m</td>
<td>22</td>
<td>111</td>
<td>245</td>
</tr>
<tr>
<td>OpenCL³ (baseline)</td>
<td>2h30m</td>
<td>4</td>
<td>1262</td>
<td>3281</td>
</tr>
<tr>
<td>Chisel</td>
<td>3h</td>
<td>25</td>
<td>39</td>
<td>129</td>
</tr>
<tr>
<td>OpenDF</td>
<td>3h30m</td>
<td>26</td>
<td>689</td>
<td>171</td>
</tr>
<tr>
<td>JHDL</td>
<td>4h</td>
<td>40</td>
<td>2529⁴</td>
<td>41</td>
</tr>
<tr>
<td>SCORE</td>
<td>4h</td>
<td>7</td>
<td>111</td>
<td>139</td>
</tr>
</tbody>
</table>

¹Flopoco only provides floating-point support for these expressions.
²MaxCompiler does not produce any intermediate RTL, directly generates executable bitstreams.
³Altera resources measured in LEs instead of LUTs, Altera 18×18 DSPs are also different from Xilinx 25×18 DSPs.
⁴JHDL directly generates a circuit netlist in EDF format instead of generating RTL.
<table>
<thead>
<tr>
<th>DSL</th>
<th>Dev. Time</th>
<th>Lines of Code</th>
<th>Resources</th>
<th>Freq. MHz</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flopoco¹</td>
<td>30m</td>
<td>2</td>
<td>1702</td>
<td>1679</td>
</tr>
<tr>
<td>Maxeler</td>
<td>30m</td>
<td>15</td>
<td>NA²</td>
<td>6036</td>
</tr>
<tr>
<td></td>
<td>30m</td>
<td></td>
<td></td>
<td>5837</td>
</tr>
<tr>
<td>Vivado HLS</td>
<td>1h</td>
<td>4</td>
<td>92</td>
<td>53</td>
</tr>
<tr>
<td>Lime (baseline)</td>
<td>2h30m</td>
<td>22</td>
<td>111</td>
<td>245</td>
</tr>
<tr>
<td></td>
<td>2h30m</td>
<td></td>
<td></td>
<td>189</td>
</tr>
<tr>
<td>OpenCL ³</td>
<td>2h30m</td>
<td>4</td>
<td>1262</td>
<td>3281</td>
</tr>
<tr>
<td></td>
<td>2h30m</td>
<td></td>
<td></td>
<td>3230</td>
</tr>
<tr>
<td>Chisel</td>
<td>3h</td>
<td>25</td>
<td>39</td>
<td>129</td>
</tr>
<tr>
<td>OpenDF</td>
<td>3h30m</td>
<td>26</td>
<td>689</td>
<td>171</td>
</tr>
<tr>
<td>JHDL</td>
<td>4h</td>
<td>40</td>
<td>2529⁴</td>
<td>41</td>
</tr>
<tr>
<td>SCORE</td>
<td>4h</td>
<td>7</td>
<td>111</td>
<td>139</td>
</tr>
</tbody>
</table>

¹ Flopoco only provides floating-point support for these expressions
² MaxCompiler does not produce any intermediate RTL, directly generates executable bitstreams
³ Altera resources measured in LEs instead of LUTs, Altera 18×18 DSPs are also different from Xilinx 25×18 DSPs
⁴ JHDL directly generates a circuit netlist in EDF format instead of generating RTL
### TABLE I: Comparing DSLs with $ax^2 + bx + c$ mapping

<table>
<thead>
<tr>
<th>DSL</th>
<th>Dev. Time</th>
<th>Lines of Code</th>
<th>Resources</th>
<th>Freq. MHz</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>DSL</td>
<td>RTL</td>
<td>LUTs</td>
<td>FFs</td>
</tr>
<tr>
<td>Flopoco$^1$</td>
<td>30m</td>
<td>2</td>
<td>1702</td>
<td>1679</td>
</tr>
<tr>
<td>Maxeler (baseline)</td>
<td>30m</td>
<td>15</td>
<td>NA$^2$</td>
<td>6036</td>
</tr>
<tr>
<td></td>
<td>30m</td>
<td></td>
<td>5837</td>
<td>5364</td>
</tr>
<tr>
<td>Vivado HLS</td>
<td>1h</td>
<td>4</td>
<td>92</td>
<td>53</td>
</tr>
<tr>
<td>Lime (baseline)</td>
<td>2h30m</td>
<td>22</td>
<td>111</td>
<td>245</td>
</tr>
<tr>
<td></td>
<td>2h30m</td>
<td></td>
<td></td>
<td>189</td>
</tr>
<tr>
<td>OpenCL$^3$</td>
<td>2h30m</td>
<td>4</td>
<td>1262</td>
<td>3281</td>
</tr>
<tr>
<td>(baseline)</td>
<td>2h30m</td>
<td></td>
<td>3230</td>
<td>4192</td>
</tr>
<tr>
<td>Chisel</td>
<td>3h</td>
<td>25</td>
<td>39</td>
<td>129</td>
</tr>
<tr>
<td>OpenDF</td>
<td>3h30m</td>
<td>26</td>
<td>689</td>
<td>171</td>
</tr>
<tr>
<td>JHDL</td>
<td>4h</td>
<td>40</td>
<td>2529$^4$</td>
<td>41</td>
</tr>
<tr>
<td>SCORE</td>
<td>4h</td>
<td>7</td>
<td>111</td>
<td>139</td>
</tr>
</tbody>
</table>

$^1$Flopoco only provides floating-point support for these expressions
$^2$MaxCompiler does not produce any intermediate RTL, directly generates executable bitstreams
$^3$Altera resources measured in LEs instead of LUTs, Altera 18×18 DSPs are also different from Xilinx 25×18 DSPs
$^4$JHDL directly generates a circuit netlist in EDIF format instead of generating RTL
### TABLE I: Comparing DSLs with $ax^2 + bx + c$ mapping

<table>
<thead>
<tr>
<th>DSL</th>
<th>Dev. Time</th>
<th>Lines of Code</th>
<th>Resources</th>
<th>Freq. MHz</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flopoco</td>
<td>30m</td>
<td>2</td>
<td>1702</td>
<td>1679</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1288</td>
</tr>
<tr>
<td>Maxeler</td>
<td>30m/30m</td>
<td>15/15 NA²</td>
<td>6036/5837</td>
<td>5391/5364</td>
</tr>
<tr>
<td>Vivado HLS</td>
<td>1h</td>
<td>4</td>
<td>92</td>
<td>53</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>71</td>
</tr>
<tr>
<td>Lime</td>
<td>2h30m/2h30m</td>
<td>22/22</td>
<td>245/22</td>
<td>284/204</td>
</tr>
<tr>
<td>OpenCL</td>
<td>2h30m/2h30m</td>
<td>4/4</td>
<td>1262/3281</td>
<td>4443/4192</td>
</tr>
<tr>
<td>Chisel</td>
<td>3h</td>
<td>25</td>
<td>39</td>
<td>129</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>64</td>
</tr>
<tr>
<td>OpenDF</td>
<td>3h30m</td>
<td>26</td>
<td>689</td>
<td>171</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>305</td>
</tr>
<tr>
<td>JHDL</td>
<td>4h</td>
<td>40</td>
<td>2529⁴</td>
<td>41</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>90</td>
</tr>
<tr>
<td>SCORE</td>
<td>4h</td>
<td>7</td>
<td>111</td>
<td>139</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>245</td>
</tr>
</tbody>
</table>

¹ Flopoco only provides floating-point support for these expressions
² MaxCompiler does not produce any intermediate RTL, directly generates executable bitstreams
³ Altera resources measured in LEs instead of LUTs, Altera 18×18 DSPs are also different from Xilinx 25×18 DSPs
⁴ JHDL directly generates a circuit netlist in EDIF format instead of generating RTL
<table>
<thead>
<tr>
<th>DSL</th>
<th>Dev. Time</th>
<th>Lines of Code</th>
<th>Resources</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>DSL</td>
<td>RTL</td>
</tr>
<tr>
<td>Flopoco¹</td>
<td>30m</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>Maxeler</td>
<td>30m</td>
<td>15</td>
<td>NA²</td>
</tr>
<tr>
<td></td>
<td>30m</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Vivado HLS</td>
<td>1h</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>Lime</td>
<td>2h30m</td>
<td>22</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2h30m</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OpenCL³</td>
<td>2h30m</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2h30m</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Chisel</td>
<td>3h</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>OpenDF</td>
<td>3h30m</td>
<td>26</td>
<td></td>
</tr>
<tr>
<td>JHDL</td>
<td>4h</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>SCORE</td>
<td>4h</td>
<td>7</td>
<td></td>
</tr>
</tbody>
</table>

¹ Flopoco only provides floating-point support for these expressions
² MaxCompiler does not produce any intermediate RTL, directly generates executable bitstreams
³ Altera resources measured in LEs instead of LUTs, Altera 18×18 DSPs are also different from Xilinx 25×18 DSPs
⁴ JHDL directly generates a circuit netlist in EDF format instead of generating RTL
TABLE I: Comparing DSLs with \( ax^2 + bx + c \) mapping

<table>
<thead>
<tr>
<th>DSL</th>
<th>Dev. Time</th>
<th>Lines of Code</th>
<th>Resources</th>
<th>Freq. MHz</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flopoco(^1)</td>
<td>30m</td>
<td>2</td>
<td>1702</td>
<td>1679</td>
</tr>
<tr>
<td>Maxeler</td>
<td>30m</td>
<td>15</td>
<td>NA(^2)</td>
<td>6036</td>
</tr>
<tr>
<td>(baseline)</td>
<td>30m</td>
<td>15</td>
<td>NA(^2)</td>
<td>5837</td>
</tr>
<tr>
<td>Vivado HLS</td>
<td>1h</td>
<td>4</td>
<td>92</td>
<td>53</td>
</tr>
<tr>
<td>Lime</td>
<td>2h30m</td>
<td>22</td>
<td>111</td>
<td>245</td>
</tr>
<tr>
<td>(baseline)</td>
<td>2h30m</td>
<td>22</td>
<td>111</td>
<td>189</td>
</tr>
<tr>
<td>OpenCL(^3)</td>
<td>2h30m</td>
<td>4</td>
<td>1262</td>
<td>3281</td>
</tr>
<tr>
<td>(baseline)</td>
<td>2h30m</td>
<td>4</td>
<td>1262</td>
<td>3230</td>
</tr>
<tr>
<td>Chisel</td>
<td>3h</td>
<td>25</td>
<td>39</td>
<td>129</td>
</tr>
<tr>
<td>OpenDF</td>
<td>3h30m</td>
<td>26</td>
<td>689</td>
<td>171</td>
</tr>
<tr>
<td>JHDL</td>
<td>4h</td>
<td>40</td>
<td>2529(^4)</td>
<td>41</td>
</tr>
<tr>
<td>SCORE</td>
<td>4h</td>
<td>7</td>
<td>111</td>
<td>139</td>
</tr>
</tbody>
</table>

\(^1\) Flopoco only provides floating-point support for these expressions.
\(^2\) MaxCompiler does not produce any intermediate RTL, directly generates executable bitstreams.
\(^3\) Altera resources measured in LEs instead of LUTs.
\(^4\) Altera 18\(\times\)18 DSPs are also different from Xilinx 25\(\times\)18 DSPs. JHDL directly generates a circuit netlist in EDIF format instead of generating RTL.
Conclusions

• Summary
  — Vast space of DSLs
  — Various states of rot — unmaintained projects

• How to navigate?
  — First attempt: Does HLS/OpenCL work for you
  — Next try: Well-supported tools such as Matlab HDLCoder, Tabview FPGA, Maxeler Dataflow
  — Finally: Check amongst the DSLs, or write your own