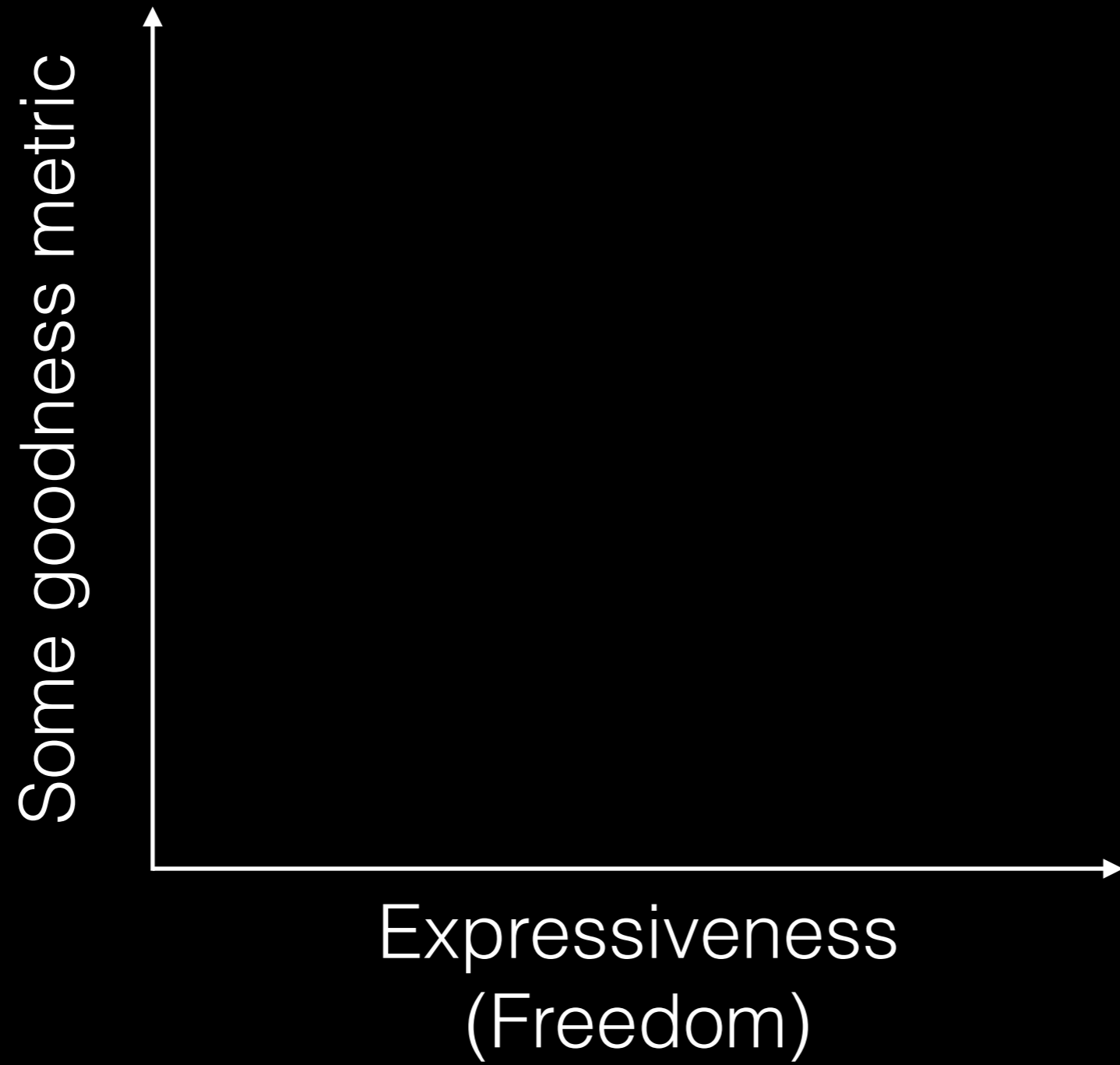
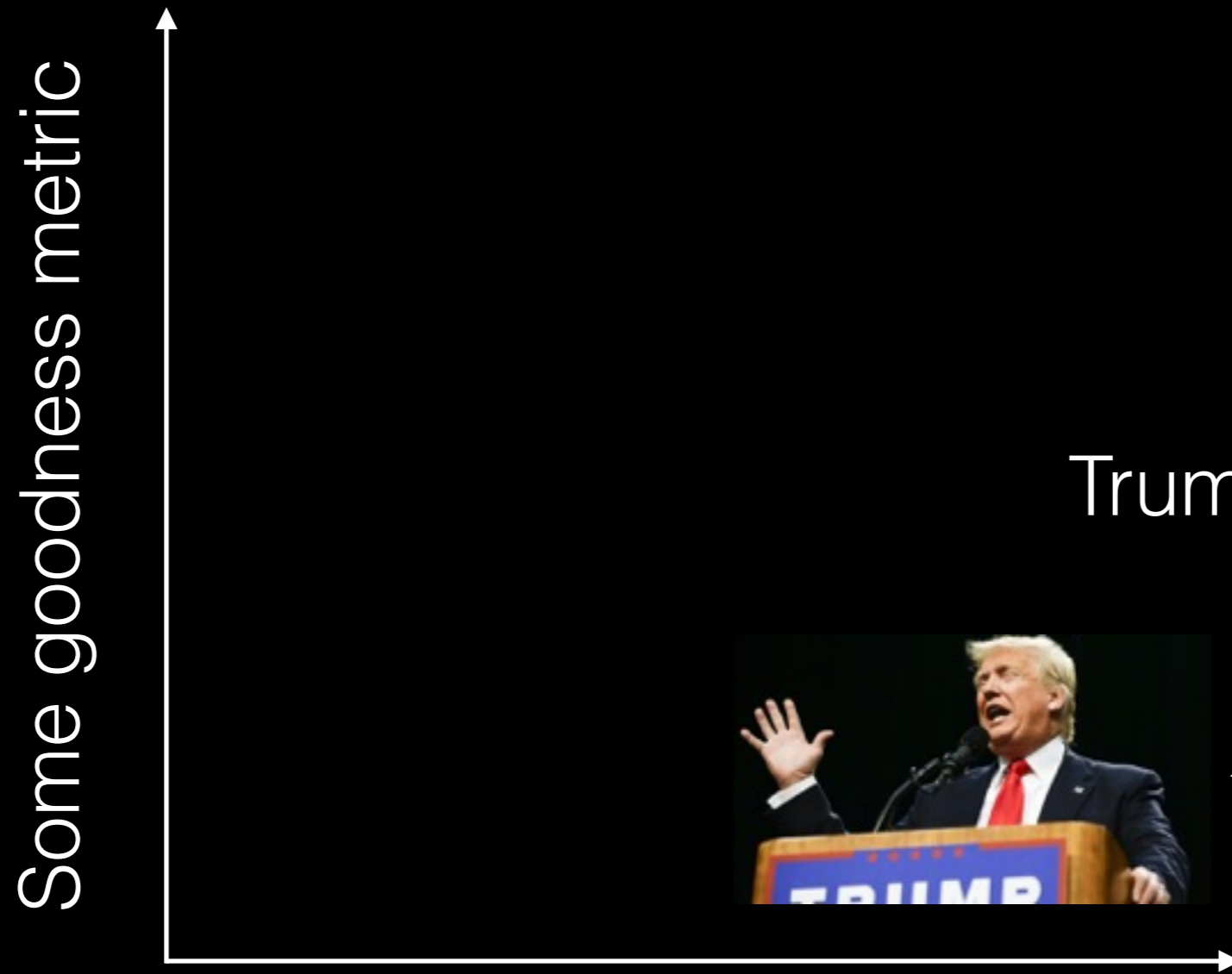


Survey of Domain-Specific Languages for FPGA Computing

Nachiket Kapre
nachiket@ieee.org





Trump's attack on judge

Expressiveness
(Freedom)

Singapore's
contempt of court bill



Trump's attack on judge



Some goodness metric

Expressiveness
(Freedom)

Singapore's contempt of court bill

Singapore: Contempt of court bill is a threat to freedom of expression
<https://www.amnesty.org/en/latest/news/2016/08/singapore-contempt-of-court-law/>



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

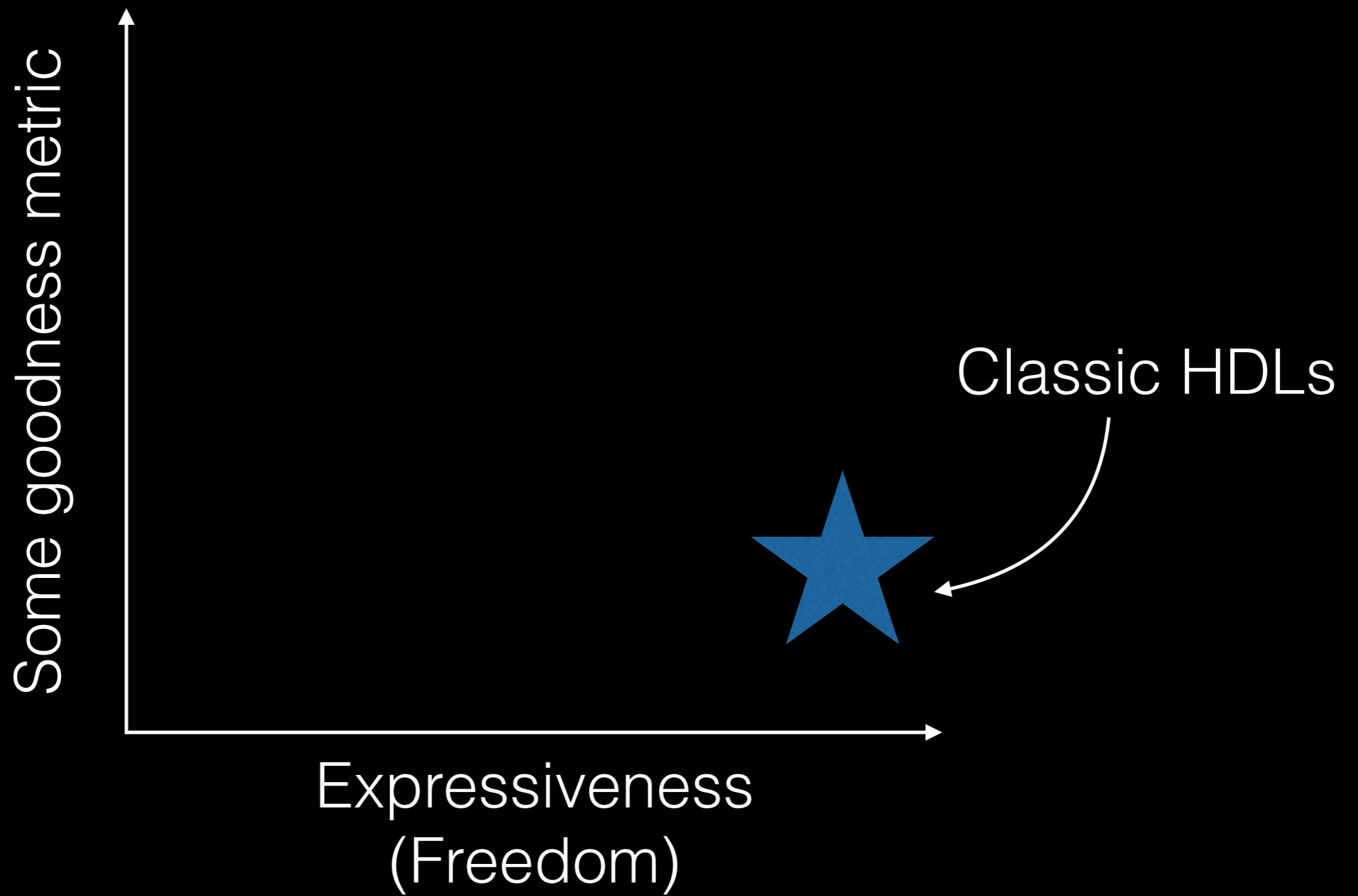
Donald Trump's hate-filled rhetoric & bigoted scapegoating flies in the face of equality & MUST be rejected.

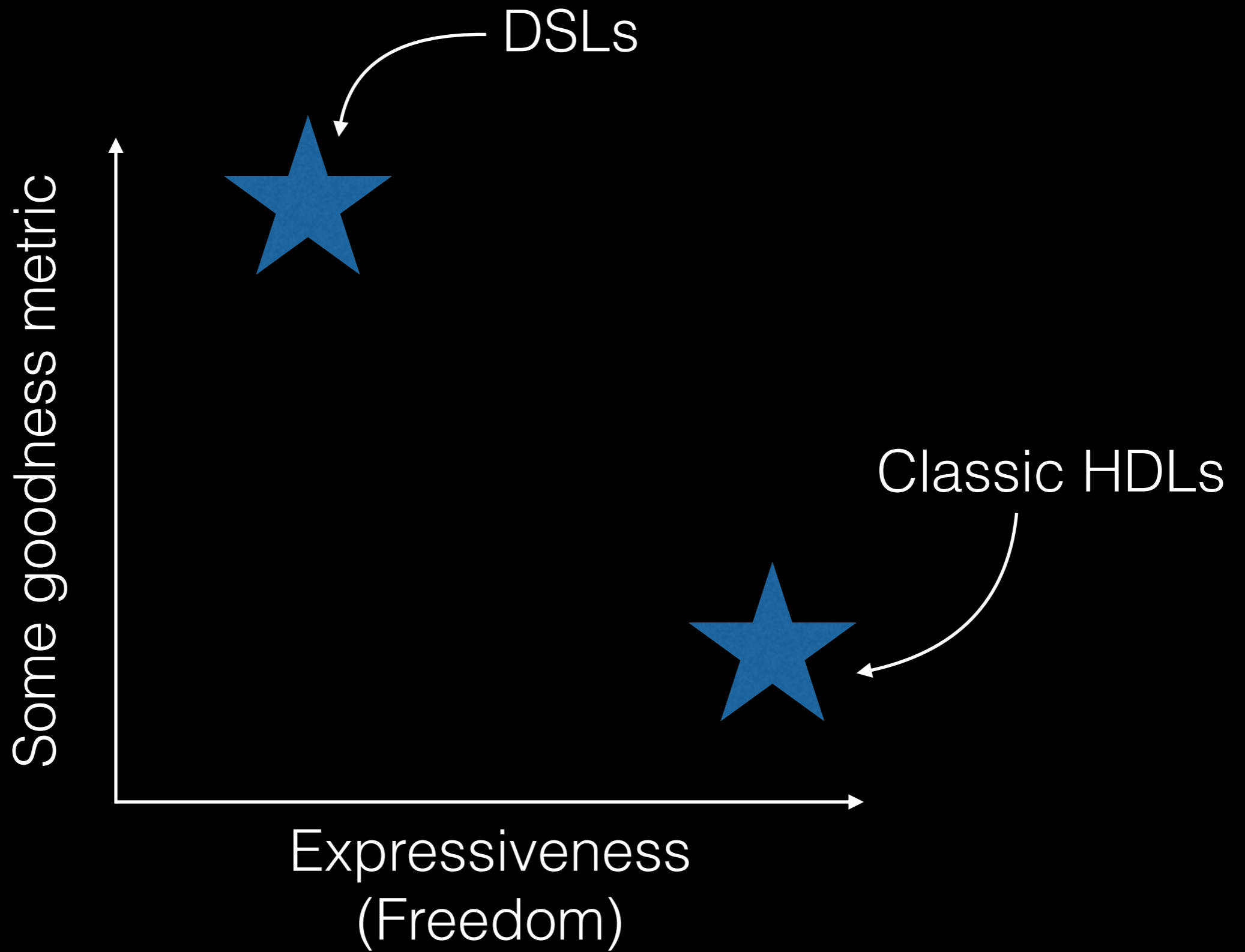
Trump's attack on judge



Some goodness metric

Expressiveness
(Freedom)





ROBERT MCMILLAN BUSINESS 06.16.14 6:30 AM

MICROSOFT SUPERCHARGING BING SEARCH PROGRAM CHIPS



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

[Share](#) [Tweet](#) 34



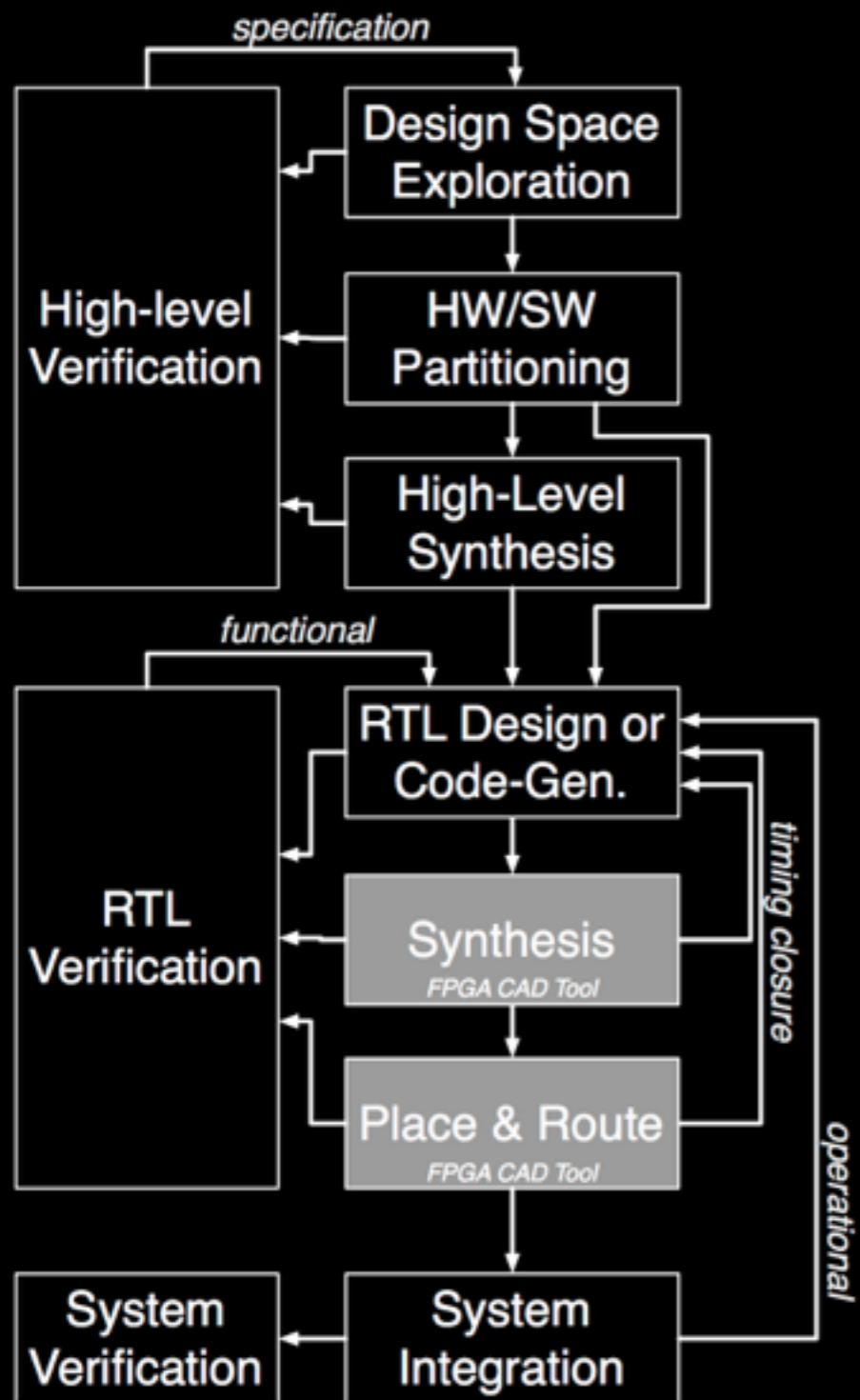
Outline

- 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

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

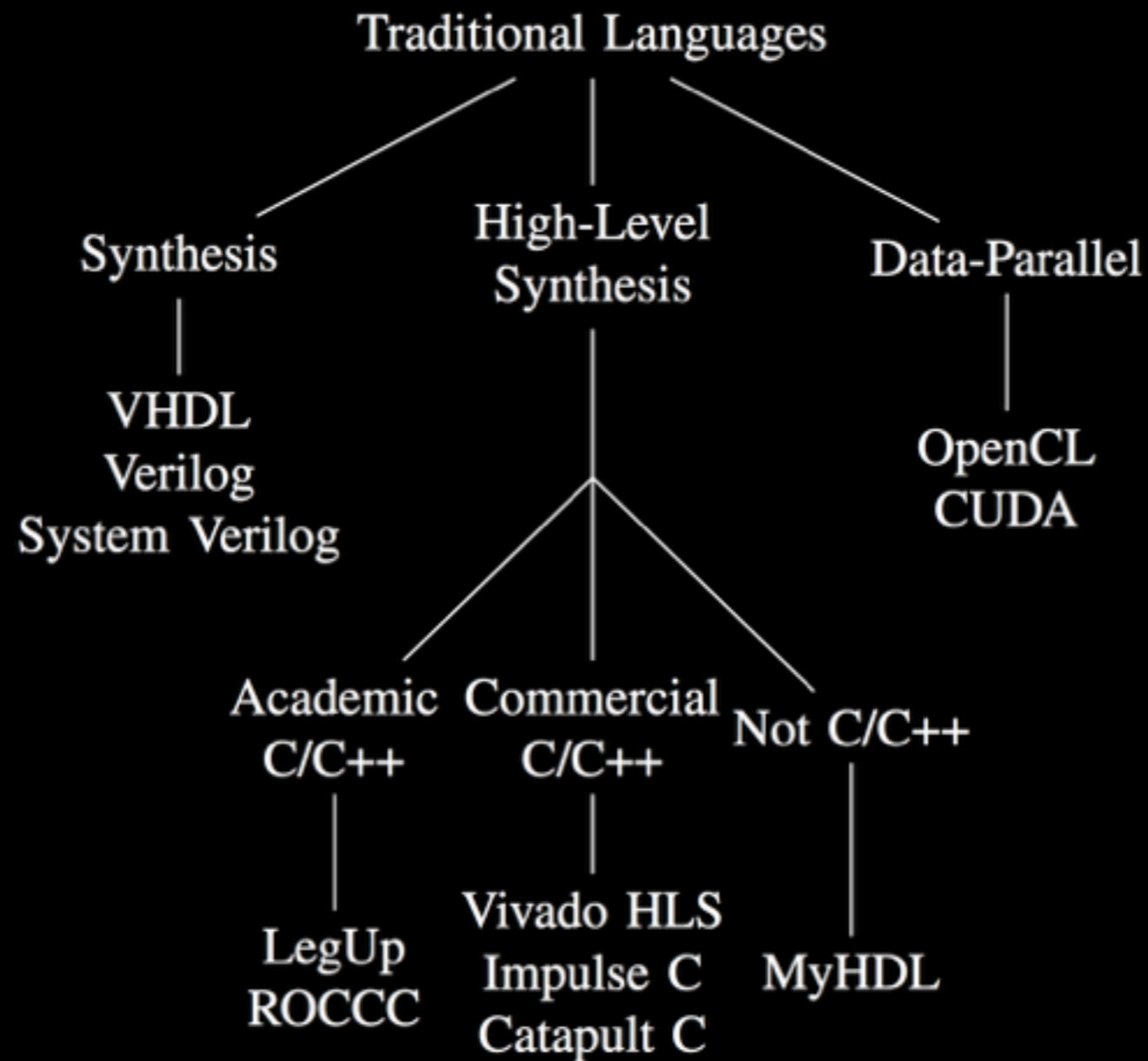
What's different?

```
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
```

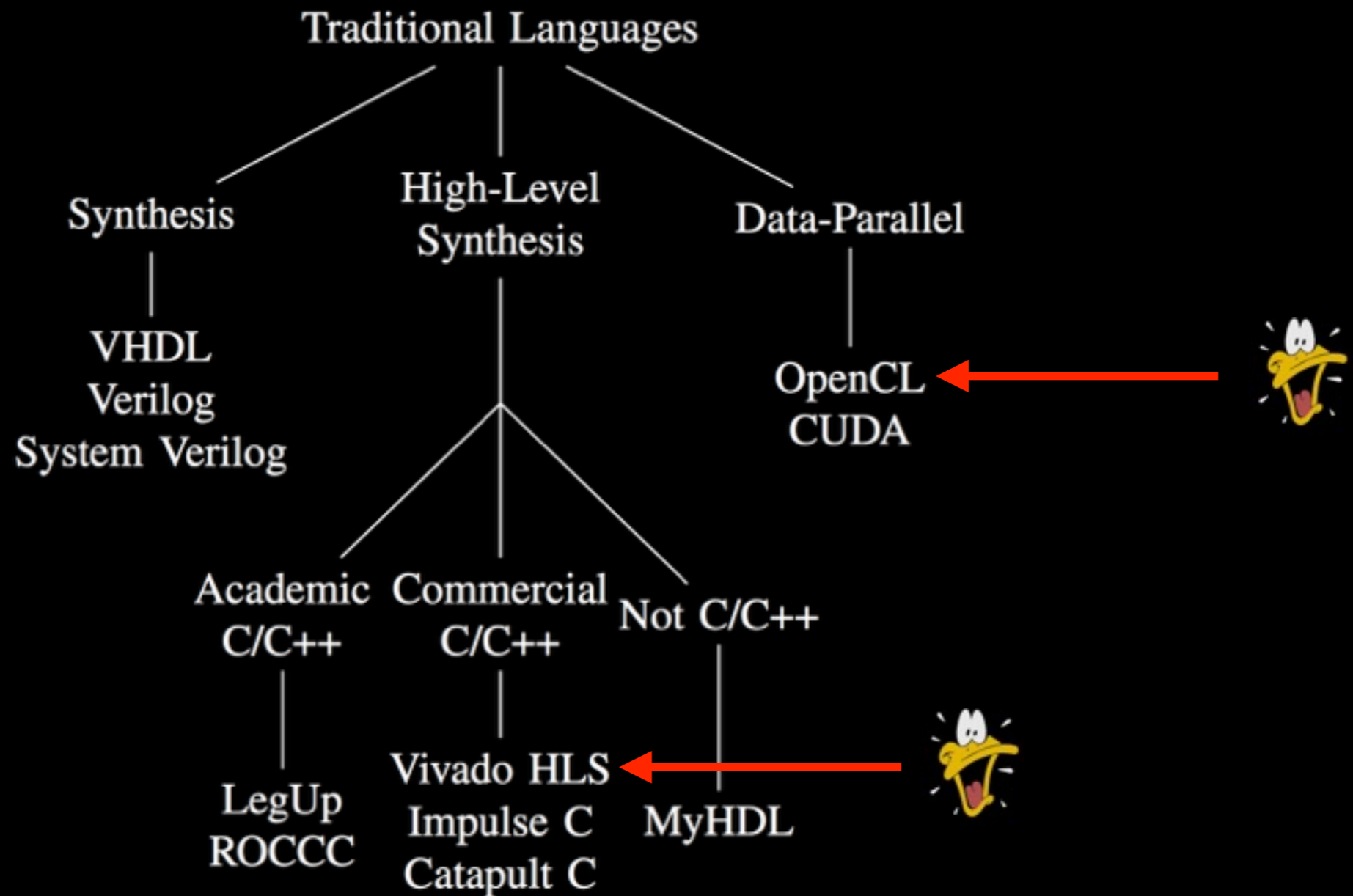
- What makes the C code smaller?
- Clocking/Reset?
- Explicit pipelining
- Type information — registers, wires, number of bits

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

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

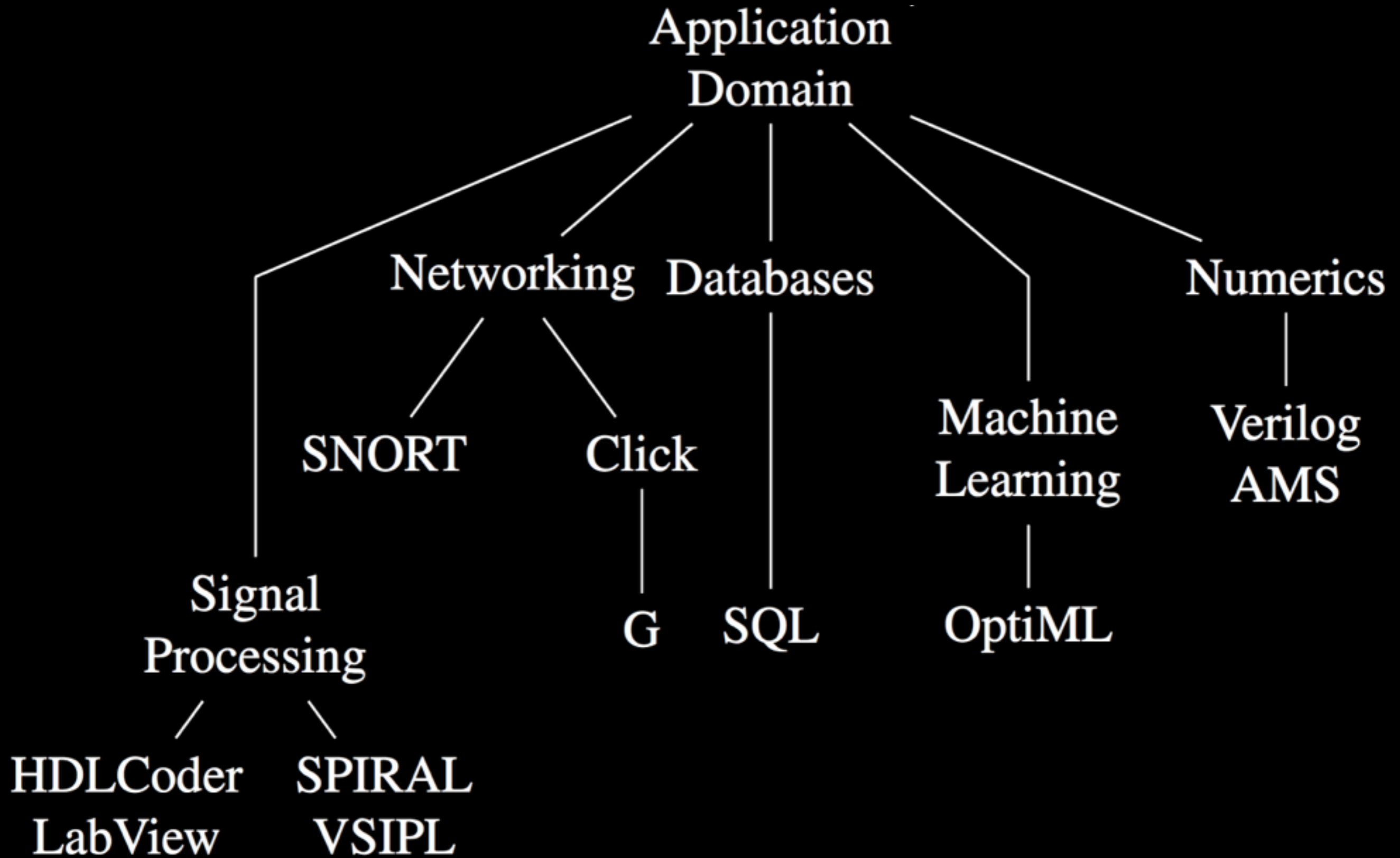
- “Beauty lies in the eye of the beholder”
- Conventional “application-domain” view
— finance, HPC, radio, multimedia, networking, databases, security.
- Suggest two alternate views in this paper...

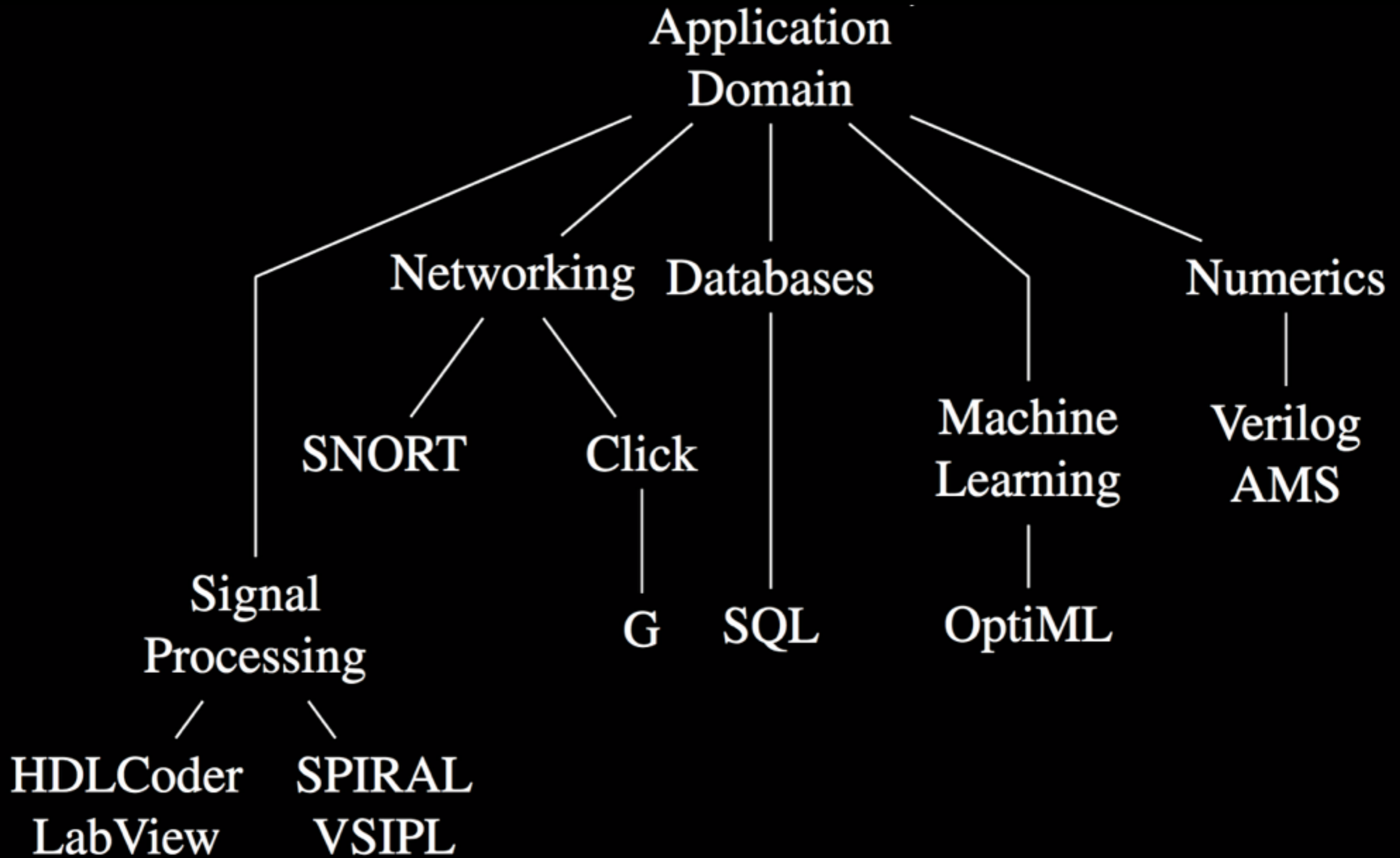
Axes of classification

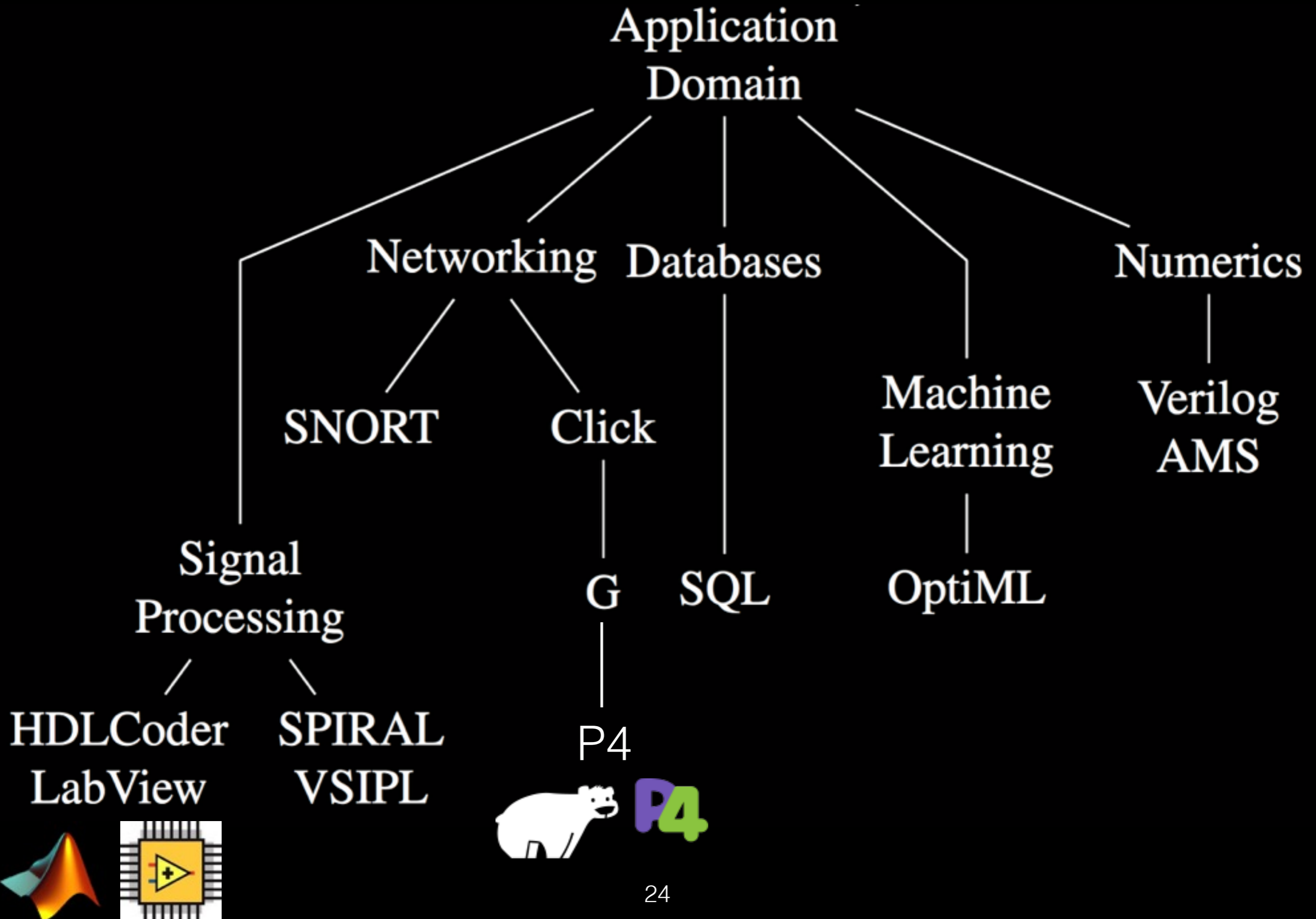
- (1) Conventional “**application-domain**” view
— focus on end-user of FPGA technology
- (2) “**compute-model**” view
— analogous to Berkeley’s Ptolemy classification
- (3) “**design**” view
— behind-the-scenes tinkerers, library developers, system builders, academics

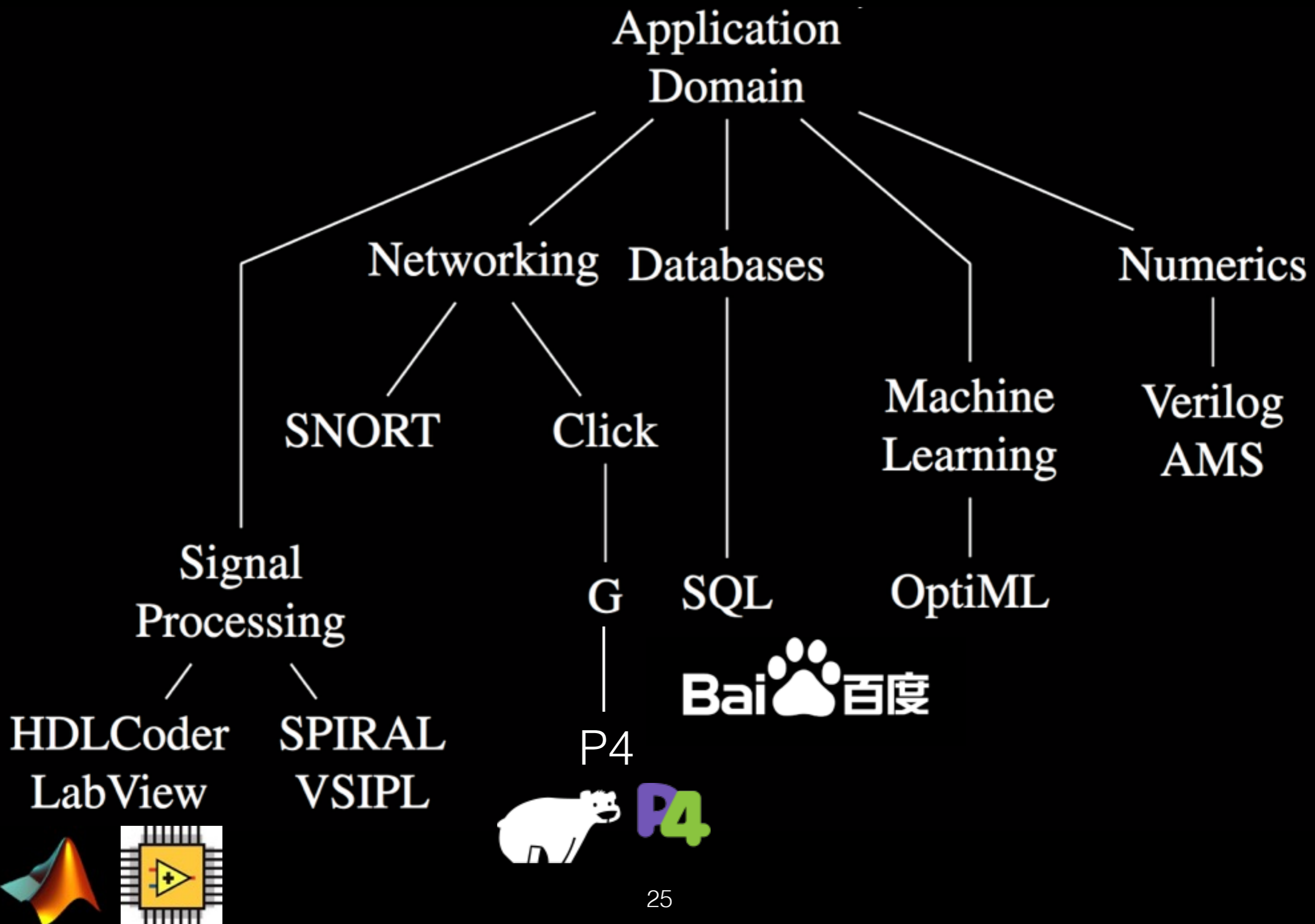
Axes of classification

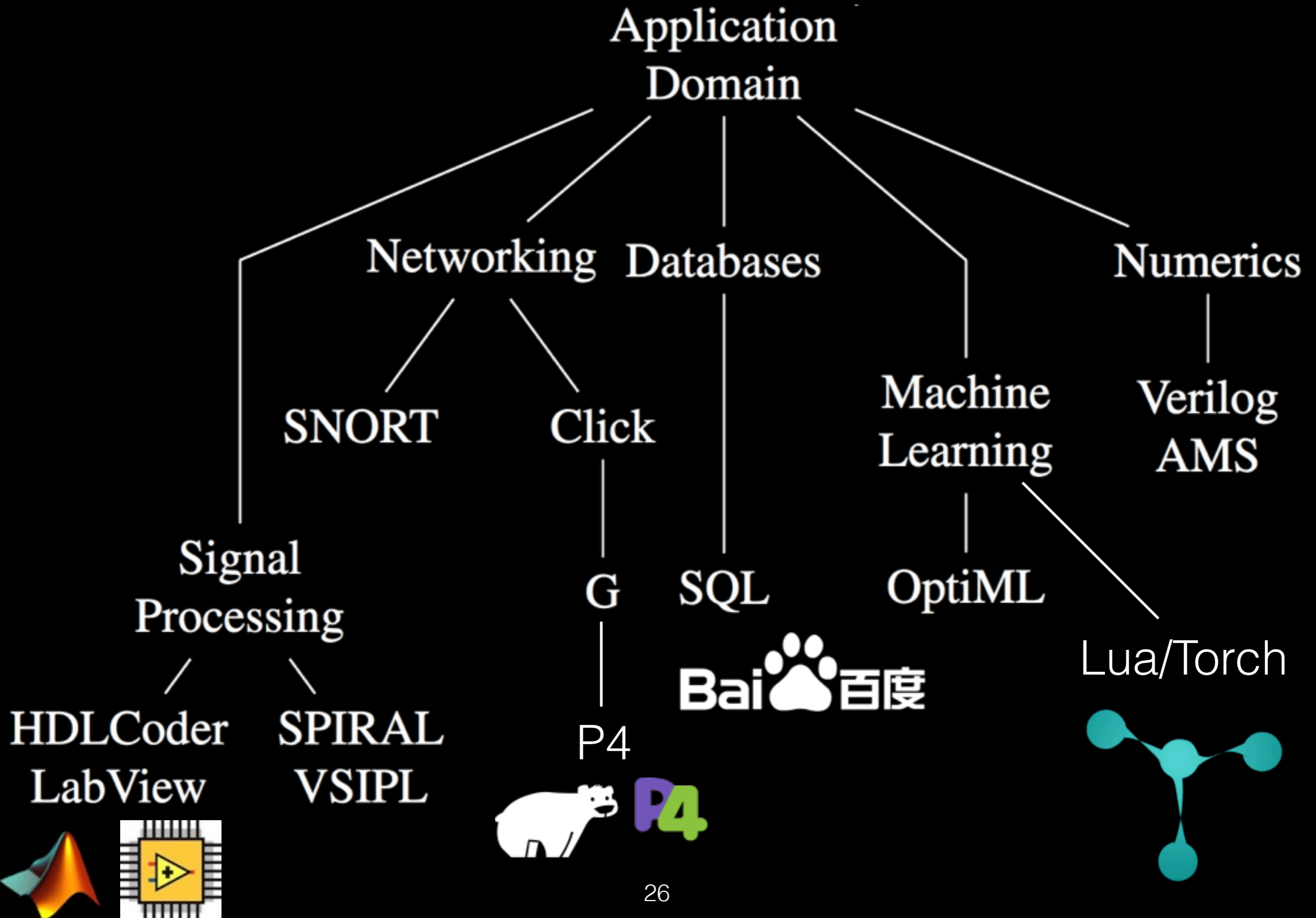
- (1) Conventional “**application-domain**” view
— focus on end-user of FPGA technology
- (2) “**compute-model**” view
— analogous to Berkeley’s Ptolemy classification
- (3) “**design**” view 
— behind-the-scenes tinkerers, library developers, system builders, academics

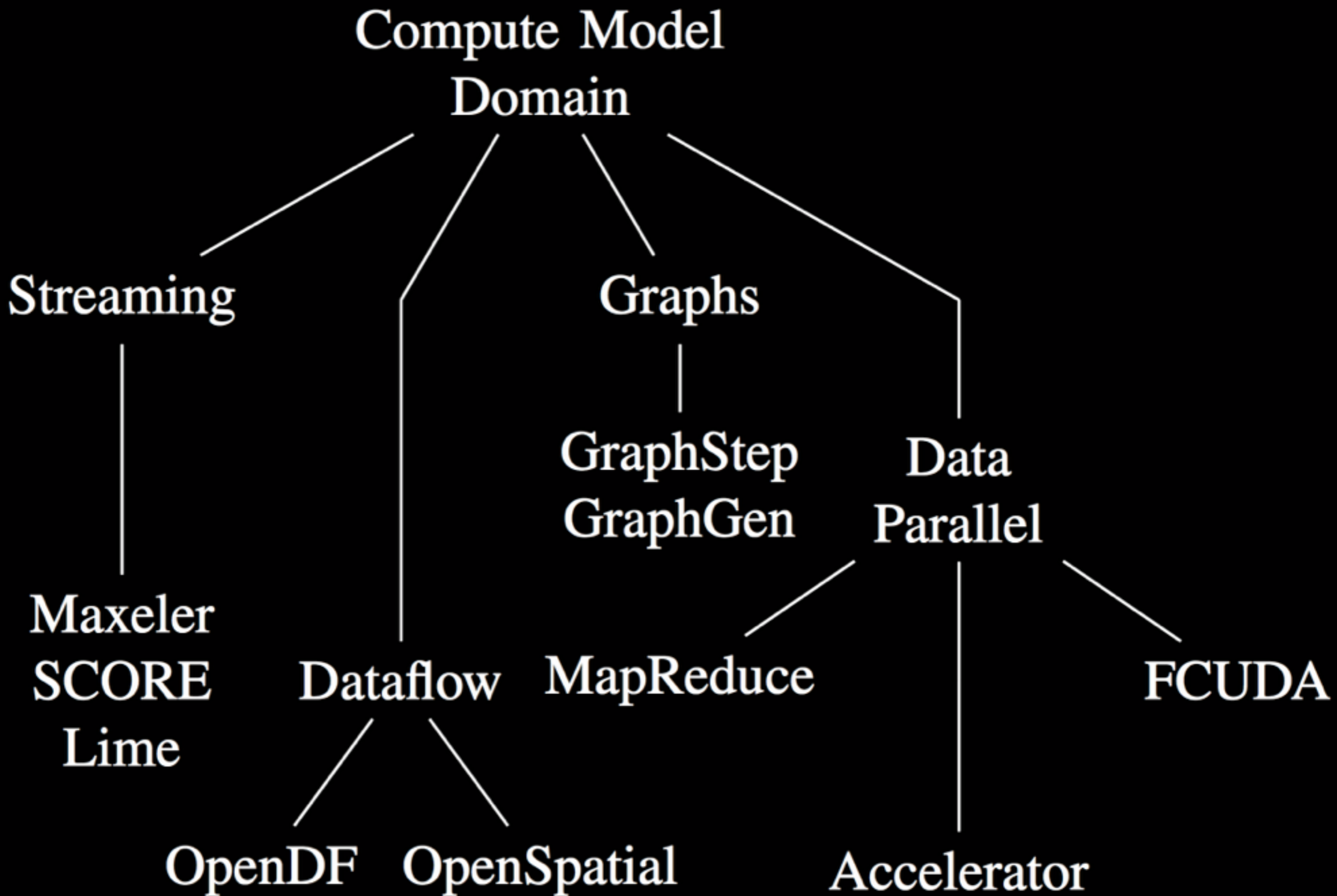












Compute Model Domain

Streaming

Graphs

Data
Parallel

GraphStep
GraphGen

MapReduce

FCUDA

Dataflow

Accelerator

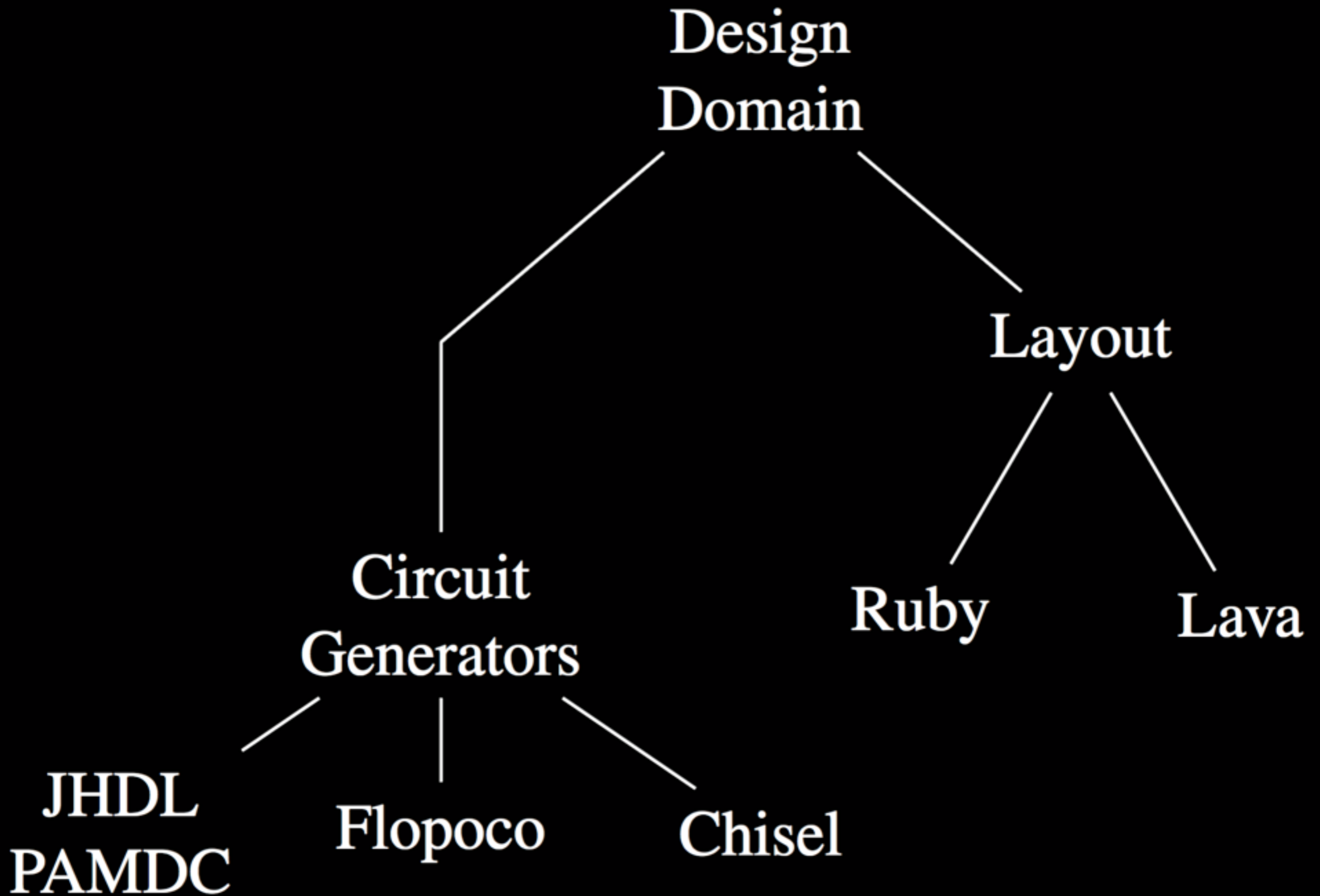
OpenSpatial

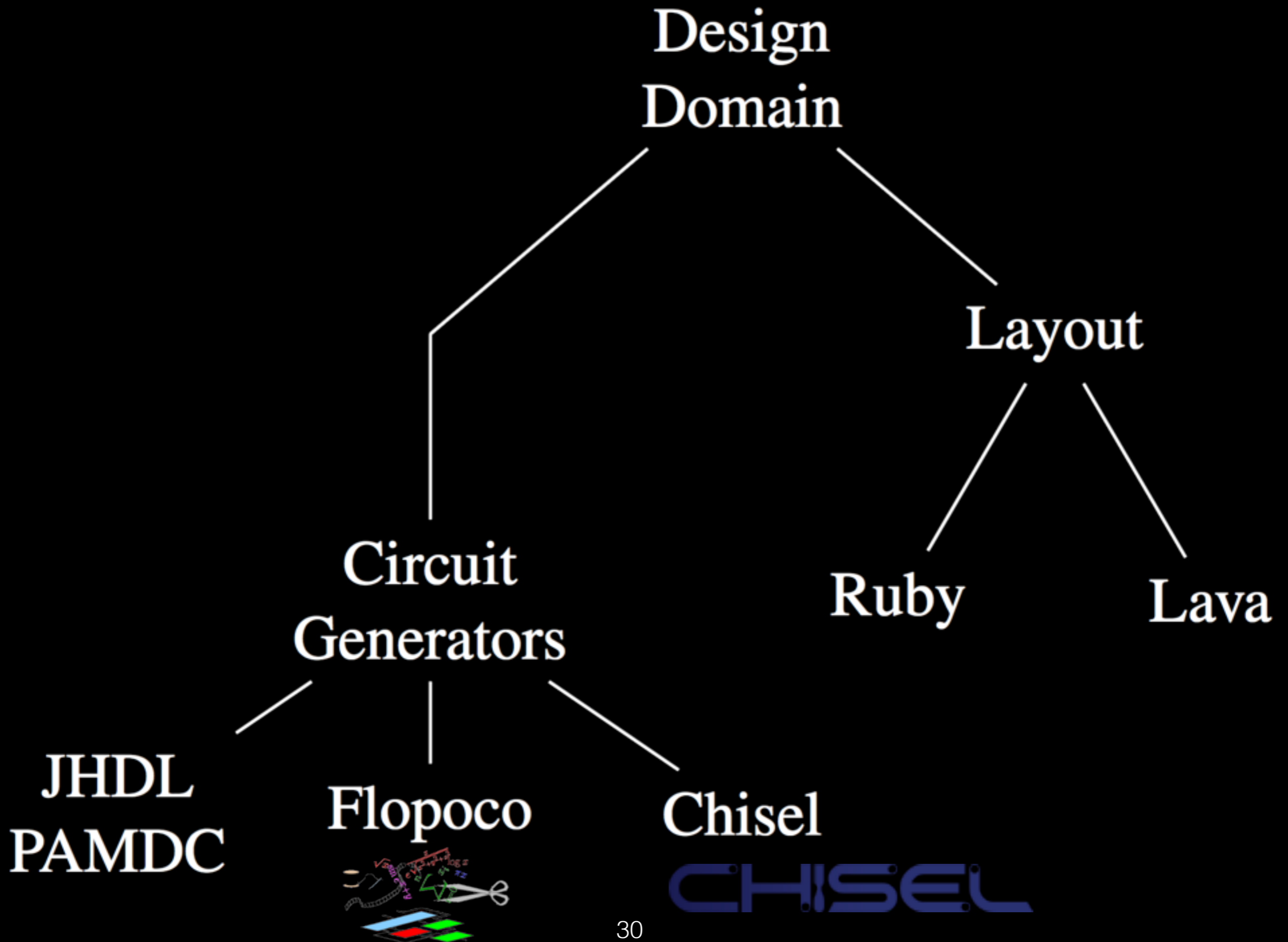
OpenDF

Maxeler
SCORE
Lime

MAXELER
Technologies







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;
```

Maxeler

```
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));  
  
  }  
}
```

SCORE

```
poly(input unsigned[32] x,  
      output unsigned[32] y)  
{  
    unsigned[32] a=3,b=2,c=1;  
  
    state always (x):  
        y = a*x*x + b*x + c;  
  
}
```

MSR Accelerator 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);
        }
    }
}
```

JHDL

```
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);  
    }  
}
```

CHISEL

```
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  
}
```

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 I: Comparing DSLs with $ax^2 + bx + c$ mapping

DSL	Dev. Time	Lines of Code		Resources			Freq. MHz
		DSL	RTL	LUTs	FFs	DSPs	
Flopoco ¹	30m	2	1702	1679	1288	0	91
Maxeler (baseline)	30m	15	NA ²	6036	5391	3	120
	30m			5837	5364	0	
Vivado HLS	1h	4	92	53	71	3	117
Lime (baseline)	2h30m	22	111	245	284	2	160
	2h30m			189	209	1	
OpenCL ³ (baseline)	2h30m	4	1262	3281	4443	2	267
	2h30m			3230	4192	0	
Chisel	3h	25	39	129	64	10	66
OpenDF	3h30m	26	689	171	305	9	120
JHDL	4h	40	2529 ⁴	41	90	3	84
SCORE	4h	7	111	139	245	2	74

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 EDIF format instead of generating RTL

TABLE I: Comparing DSLs with $ax^2 + bx + c$ mapping

DSL	Dev. Time	Lines of Code		Resources			Freq. MHz
		DSL	RTL	LUTs	FFs	DSPs	
Flopoco ¹	30m	2	1702	1679	1288	0	91
Maxeler (baseline)	30m	15	NA ²	6036	5391	3	120
	30m			5837	5364	0	
Vivado HLS	1h	4	92	53	71	3	117
Lime (baseline)	2h30m	22	111	245	284	2	160
	2h30m			189	209	1	
OpenCL ³ (baseline)	2h30m	4	1262	3281	4443	2	267
	2h30m			3230	4192	0	
Chisel	3h	25	39	129	64	10	66
OpenDF	3h30m	26	689	171	305	9	120
JHDL	4h	40	2529 ⁴	41	90	3	84
SCORE	4h	7	111	139	245	2	74

Compiler modified



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 EDIF format instead of generating RTL

TABLE I: Comparing DSLs with $ax^2 + bx + c$ mapping

DSL	Dev. Time	Lines of Code		Resources			Freq. MHz
		DSL	RTL	LUTs	FFs	DSPs	
Flopoco ¹	30m	2	1702	1679	1288	0	91
Maxeler (baseline)	30m	15	NA ²	6036	5391	3	120
	30m			5837	5364	0	
Vivado HLS	1h	4	92	53	71	3	117
Lime (baseline)	2h30m	22	111	245	284	2	160
	2h30m			189	209	1	
OpenCL ³ (baseline)	2h30m	4	1262	3281	4443	2	267
	2h30m			3230	4192	0	
Chisel	3h	25	39	129	64	10	66
OpenDF	3h30m	26	689	171	305	9	120
JHDL	4h	40	2529 ⁴	41	90	3	84
SCORE	4h	7	111	139	245	2	74

Vendor HLS

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 EDIF format instead of generating RTL

TABLE I: Comparing DSLs with $ax^2 + bx + c$ mapping

DSL	Dev. Time	Lines of Code		Resources			Freq. MHz
		DSL	RTL	LUTs	FFs	DSPs	
Flopoco ¹	30m	2	1702	1679	1288	0	91
Maxeler (baseline)	30m	15	NA ²	6036	5391	3	120
	30m			5837	5364	0	
Vivado HLS	1h	4	92	53	71	3	117
Lime (baseline)	2h30m	22	111	245	284	2	160
	2h30m			189	209	1	
OpenCL ³ (baseline)	2h30m	4	1262	3281	4443	2	267
	2h30m			3230	4192	0	
Chisel	3h	25	39	129	64	10	66
OpenDF	3h30m	26	689	171	305	9	120
JHDL	4h	40	2529 ⁴	41	90	3	84
SCORE	4h	7	111	139	245	2	74

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 EDIF format instead of generating RTL

Limited
to arith
expr

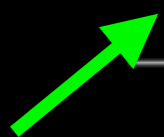


TABLE I: Comparing DSLs with $ax^2 + bx + c$ mapping

DSL	Dev. Time	Lines of Code		Resources			Freq. MHz
		DSL	RTL	LUTs	FFs	DSPs	
Flopoco ¹	30m	2	1702	1679	1288	0	91
Maxeler (baseline)	30m	15	NA ²	6036	5391	3	120
	30m			5837	5364	0	
Vivado HLS	1h	4	92	53	71	3	117
Lime (baseline)	2h30m	22	111	245	284	2	160
	2h30m			189	209	1	
OpenCL ³ (baseline)	2h30m	4	1262	3281	4443	2	267
	2h30m			3230	4192	0	
Chisel	3h	25	39	129	64	10	66
OpenDF	3h30m	26	689	171	305	9	120
JHDL	4h	40	2529 ⁴	41	90	3	84
SCORE	4h	7	111	139	245	2	74

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 EDIF format instead of generating RTL

Tool config tough

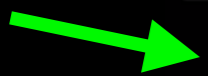
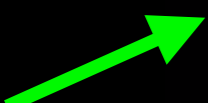


TABLE I: Comparing DSLs with $ax^2 + bx + c$ mapping

DSL	Dev. Time	Lines of Code		Resources			Freq. MHz
		DSL	RTL	LUTs	FFs	DSPs	
Flopoco ¹	30m	2	1702	1679	1288	0	91
Maxeler (baseline)	30m	15	NA ²	6036	5391	3	120
	30m			5837	5364	0	
Vivado HLS	1h	4	92	53	71	3	117
Lime (baseline)	2h30m	22	111	245	284	2	160
	2h30m			189	209	1	
OpenCL ³ (baseline)	2h30m	4	1262	3281	4443	2	267
	2h30m			3230	4192	0	
Chisel	3h	25	39	129	64	10	66
OpenDF	3h30m	26	689	171	305	9	120
JHDL	4h	40	2529 ⁴	41	90	3	84
SCORE	4h	7	111	139	245	2	74

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 EDIF format instead of generating RTL

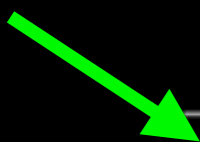
Dated EDIFs



TABLE I: Comparing DSLs with $ax^2 + bx + c$ mapping

DSL	Dev. Time	Lines of Code		Resources			Freq. MHz
		DSL	RTL	LUTs	FFs	DSPs	
Flopoco ¹	30m	2	1702	1679	1288	0	91
Maxeler (baseline)	30m	15	NA ²	6036	5391	3	120
	30m			5837	5364	0	
Vivado HLS	1h	4	92	53	71	3	117
Lime (baseline)	2h30m	22	111	245	284	2	160
	2h30m			189	209	1	
OpenCL ³ (baseline)	2h30m	4	1262	3281	4443	2	267
	2h30m			3230	4192	0	
Chisel	3h	25	39	129	64	10	66
OpenDF	3h30m	26	689	171	305	9	120
JHDL	4h	40	2529 ⁴	41	90	3	84
SCORE	4h	7	111	139	245	2	74

Hardware students disliked







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 I: Comparing DSLs with $ax^2 + bx + c$ mapping

	DSL	Dev. Time	Lines of Code		Resources			Freq. MHz
			DSL	RTL	LUTs	FFs	DSPs	
	Flopoco ¹	30m	2	1702	1679	1288	0	91
	Maxeler (baseline)	30m 30m	15	NA ²	6036 5837	5391 5364	3 0	120
	Vivado HLS	1h	4	92	53	71	3	117
	Lime (baseline)	2h30m 2h30m	22	111	245 189	284 209	2 1	160
	OpenCL ³ (baseline)	2h30m 2h30m	4	1262	3281 3230	4443 4192	2 0	267
	Chisel	3h	25	39	129	64	10	66
	OpenDF	3h30m	26	689	171	305	9	120
	JHDL	4h	40	2529 ⁴	41	90	3	84
	SCORE	4h	7	111	139	245	2	74

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 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