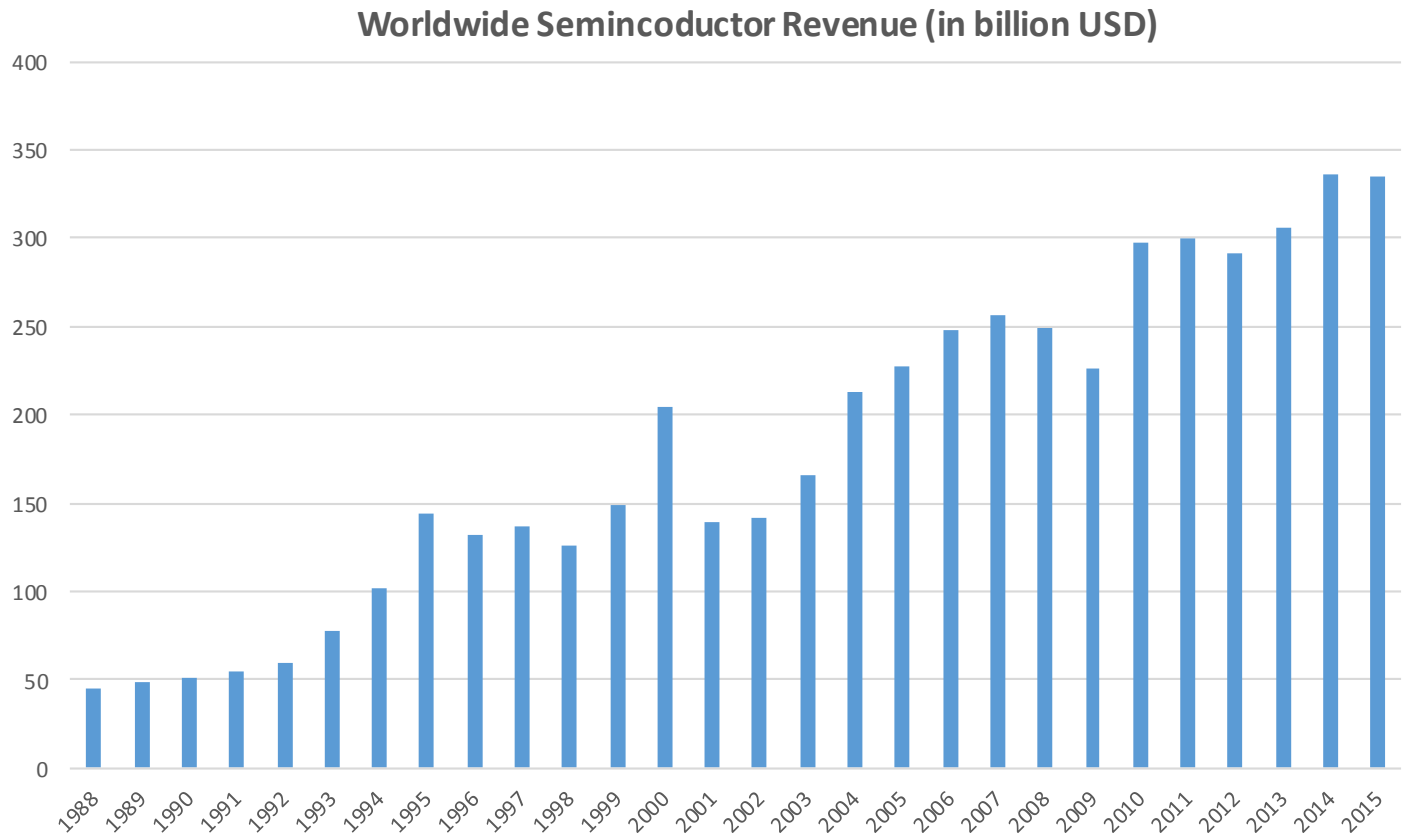
Design Challenges In The “More-than-Moore” Era

7/11/2016

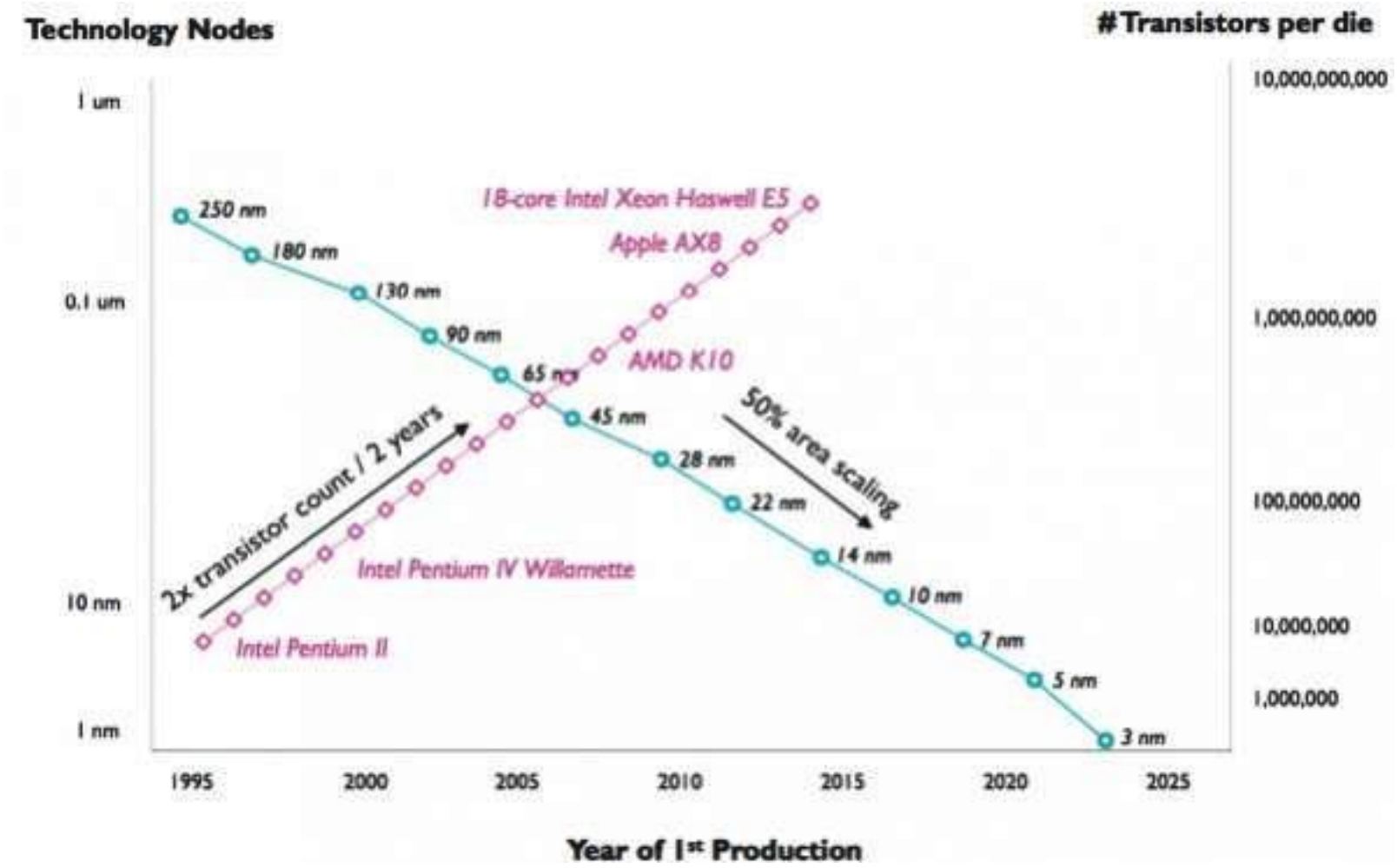
MARVELL

Decent Growth of Semiconductor Industry



Source: WSTS

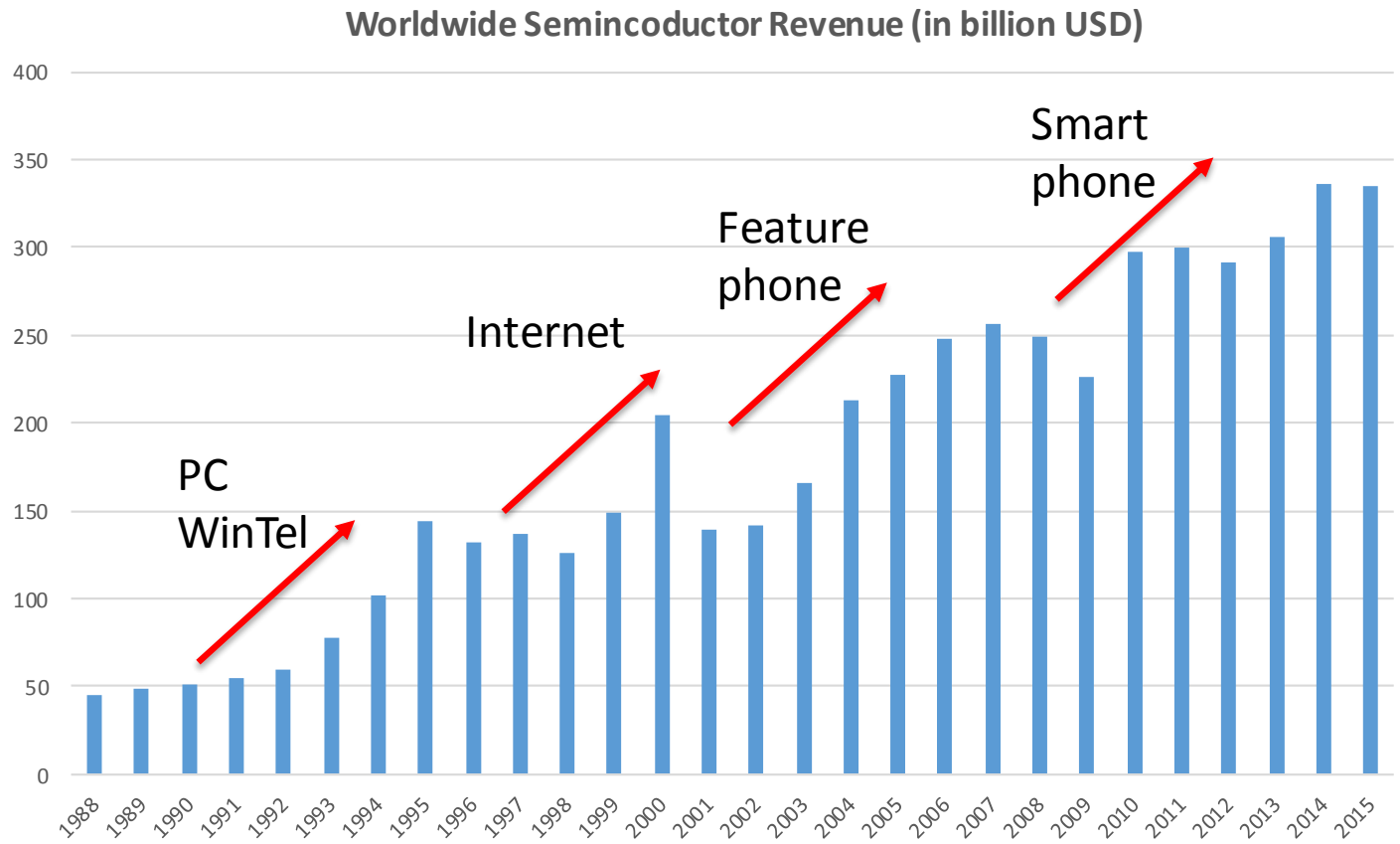
Moore's Law is the Biggest Contributor



Moore's Law is not Dead

Source: IMEC
An Steegen, 2015

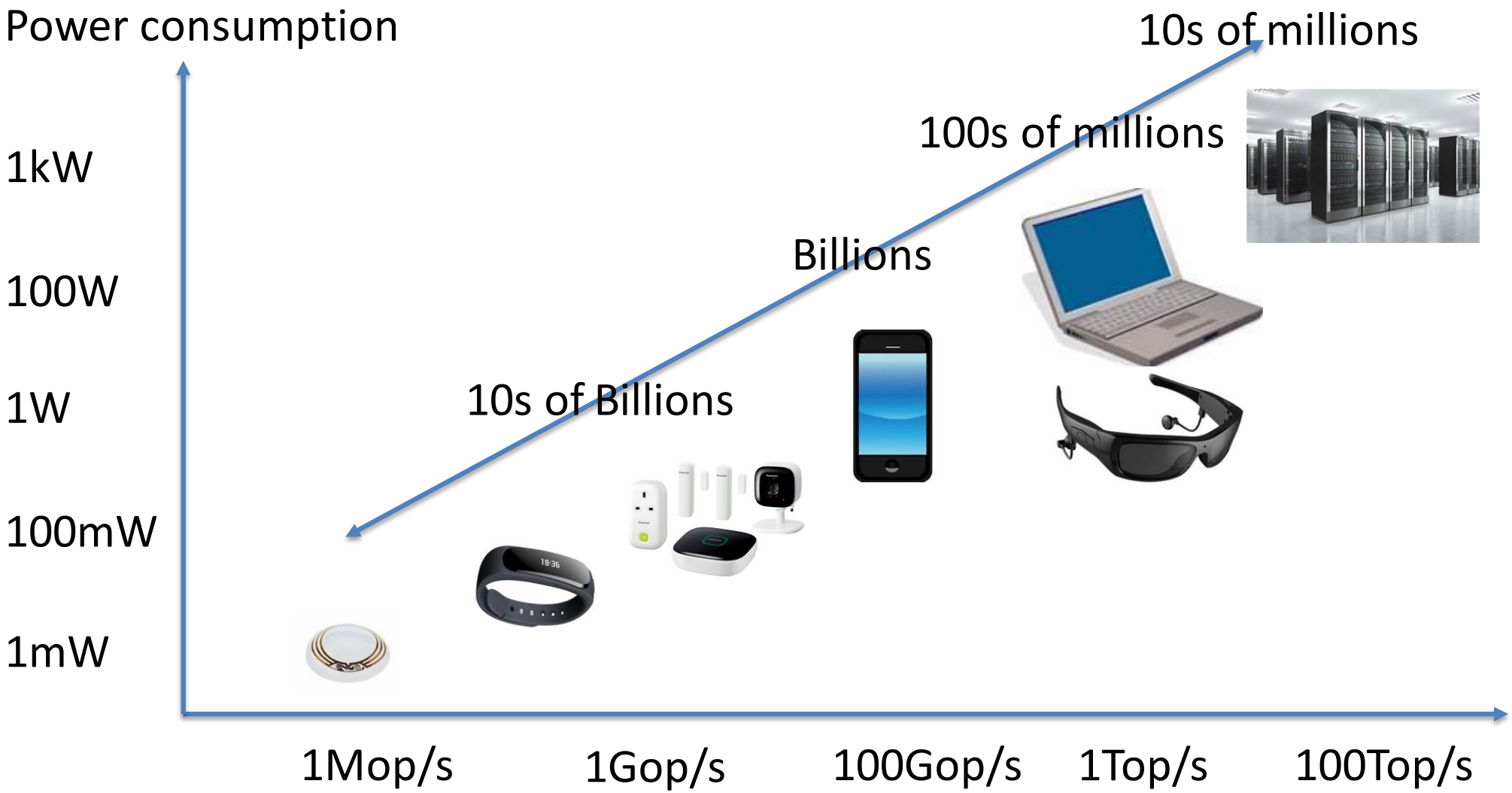
There Is Always New Market To Fuel The Growth



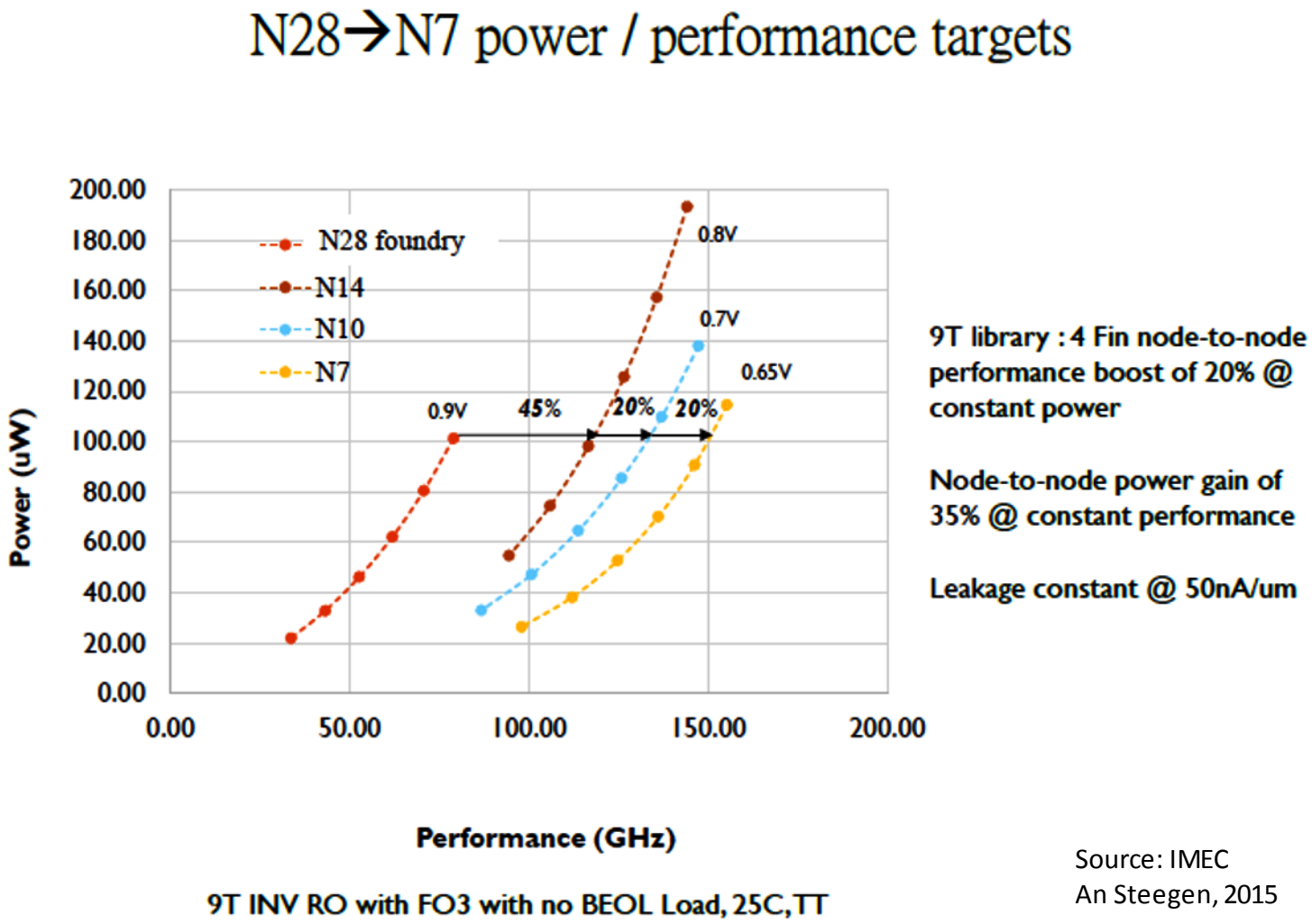
Source: WSTS

Device	Units (in millions)
PC	350
Smartphone	1433

The Era of Internet of Everything – Even Bigger Market



CMOS Will Provide What IoT Needs, …



Source: IMEC
An Steegen, 2015

- Good process needs to provide improvements in power, performance, and area (cost) – PPA
- No problem on power and performance, but what about cost?

Together With “More-Than-Moore” Technologies

- 3D packing, MEMS, near threshold or subthreshold operations, …

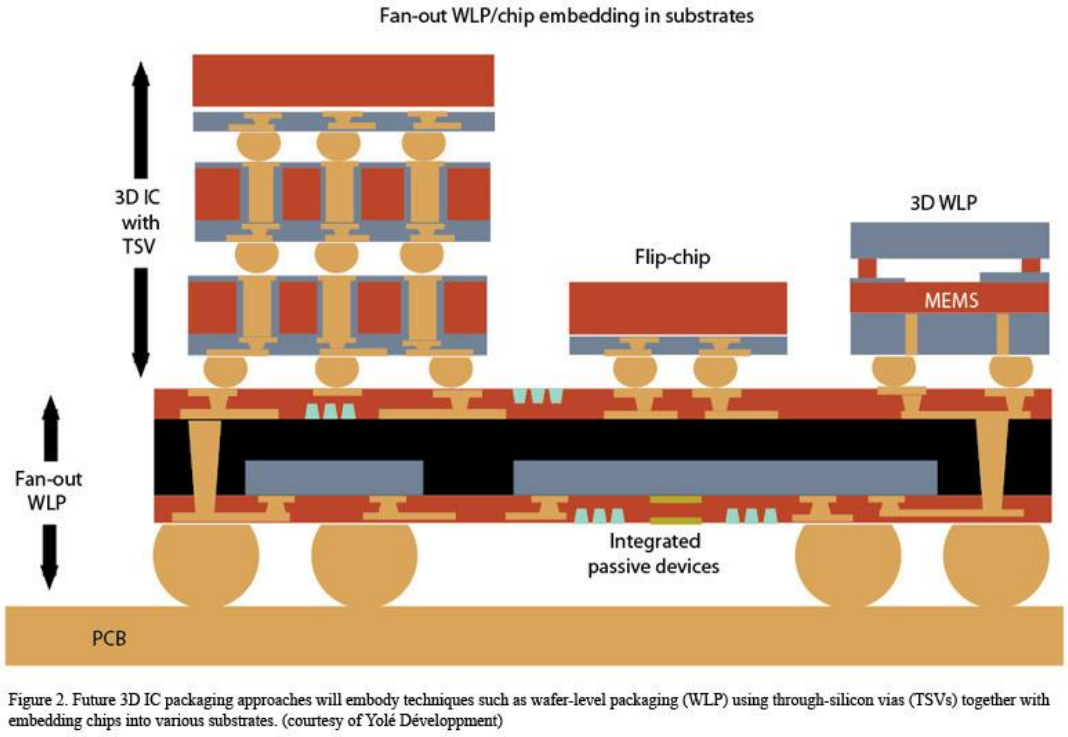
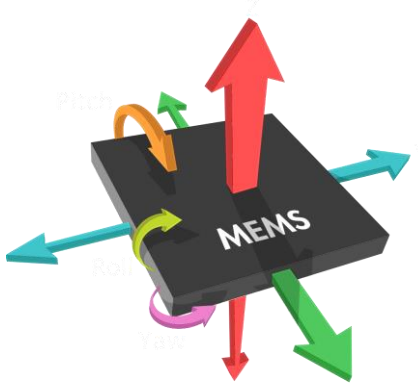
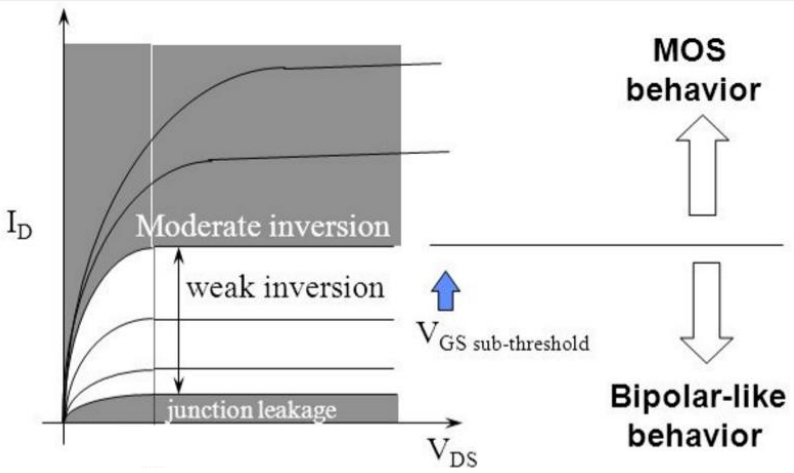


Figure 2. Future 3D IC packaging approaches will embody techniques such as wafer-level packaging (WLP) using through-silicon vias (TSVs) together with embedding chips into various substrates. (courtesy of Yolé Développment)

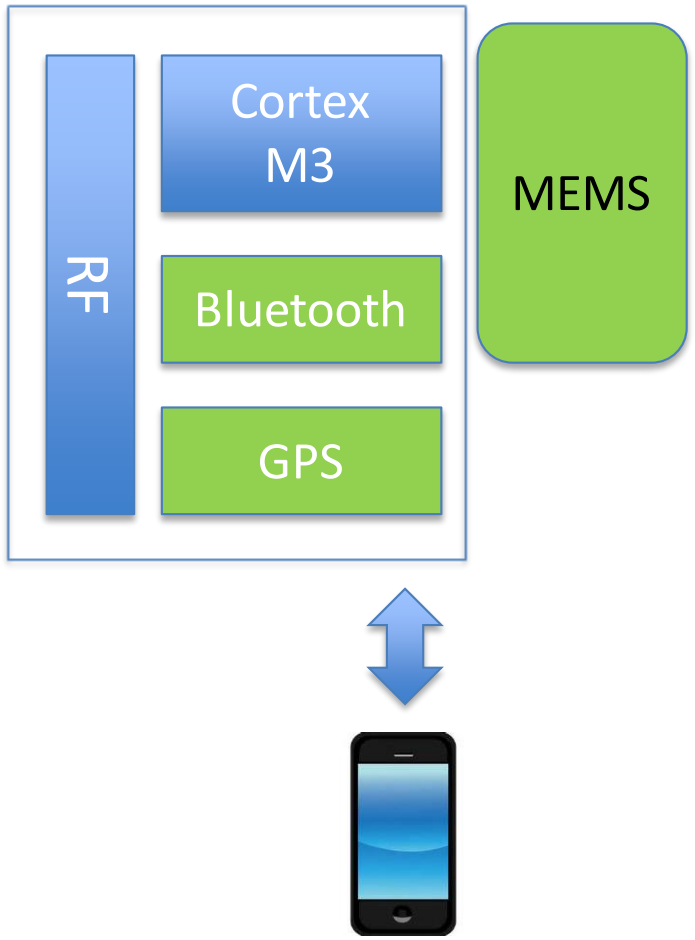
Source: http://electronicdesign.com/archive/3d-ic-technology-delivers-total-package



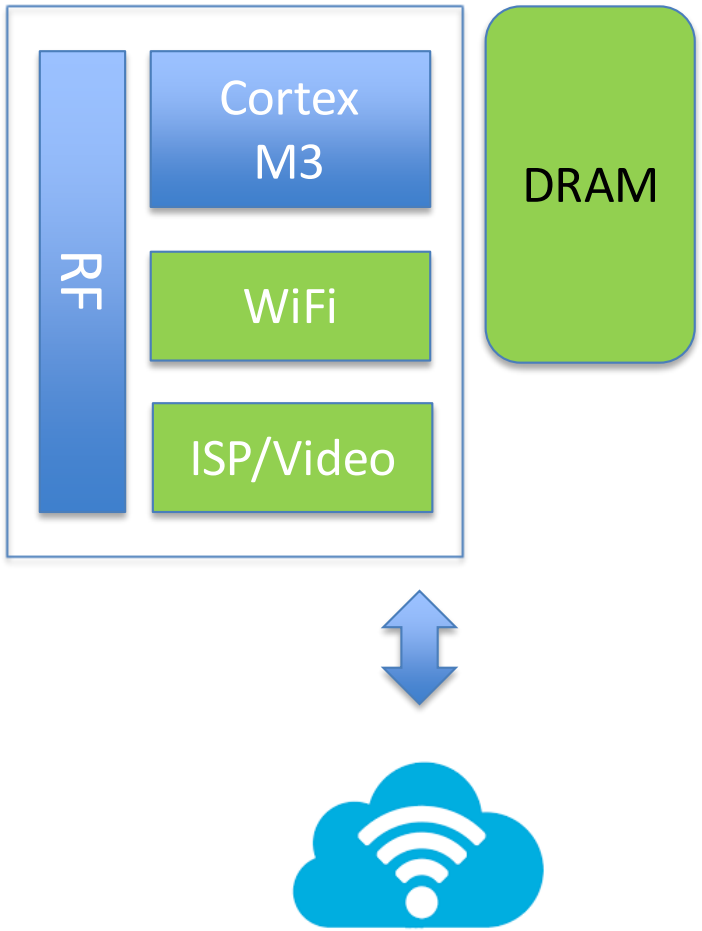


IoT Devices Are Simple, But …

Wristband/Smart Watch



WiFi Home Monitor

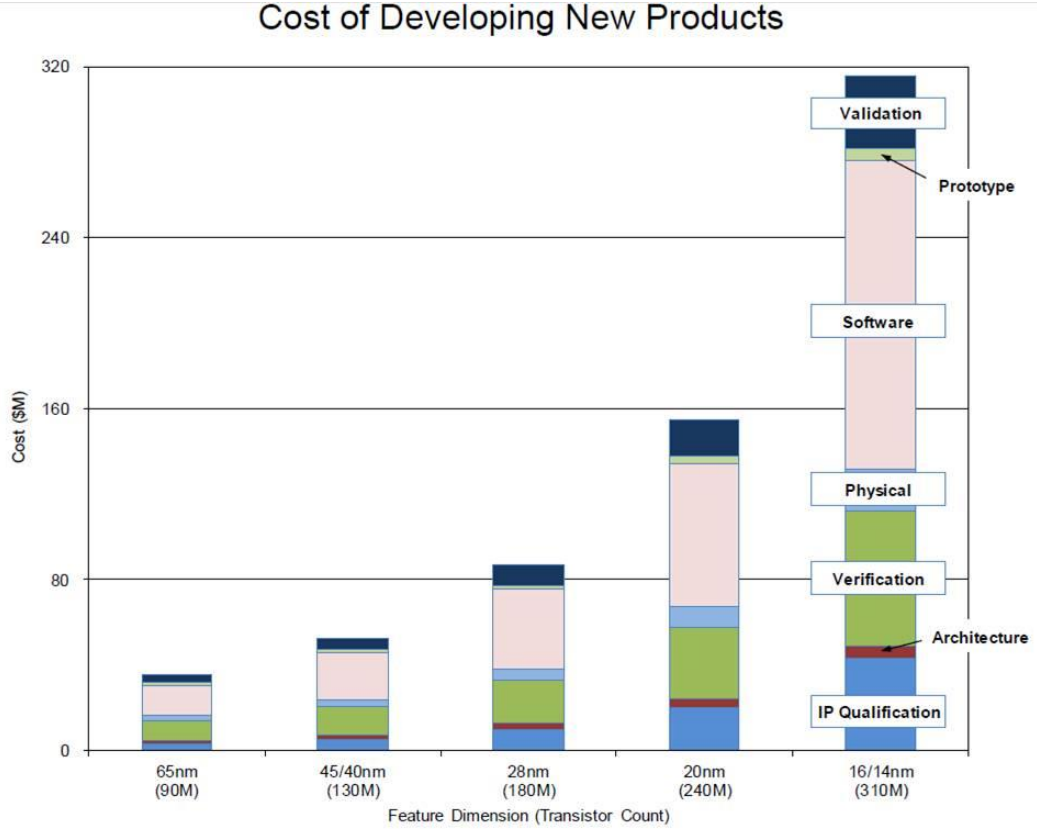


IoT is a very fragmented market

Similar architecture, but different building blocks

Economy Of Chip Design Doesn't Work Out

$$\text{unit price} = \frac{\text{R\&D cost}+\text{Mask cost}+\text{Package NRE}}{\#\text{ of chips}} + \text{Per die cost} + \text{Per package cost}$$



R&D cost	Increase
Mask cost	Increase
Package NRE	Increase
# of chip	Decrease
Per dies cost	Decrease
Per package cost	Flat or decrease

Requires 10s of Millions Units for The Same Design

Example - R&D cost $50M, Mask cost $7.5M, Per die cost $1, with100% ROI requirement

Challenges for SoC Companies

Moore's law is not dead, just not affordable

Moving down technology nodes can improve Performance and Power, not Price

Harder for smaller company to compete

Finding More Cost Effective Solutions

$$\text{chip price} = \frac{\text{R\&D cost+Mask cost+Package NRE}}{\text{\# of chips}} + \text{Per die cost+Per package cost}$$

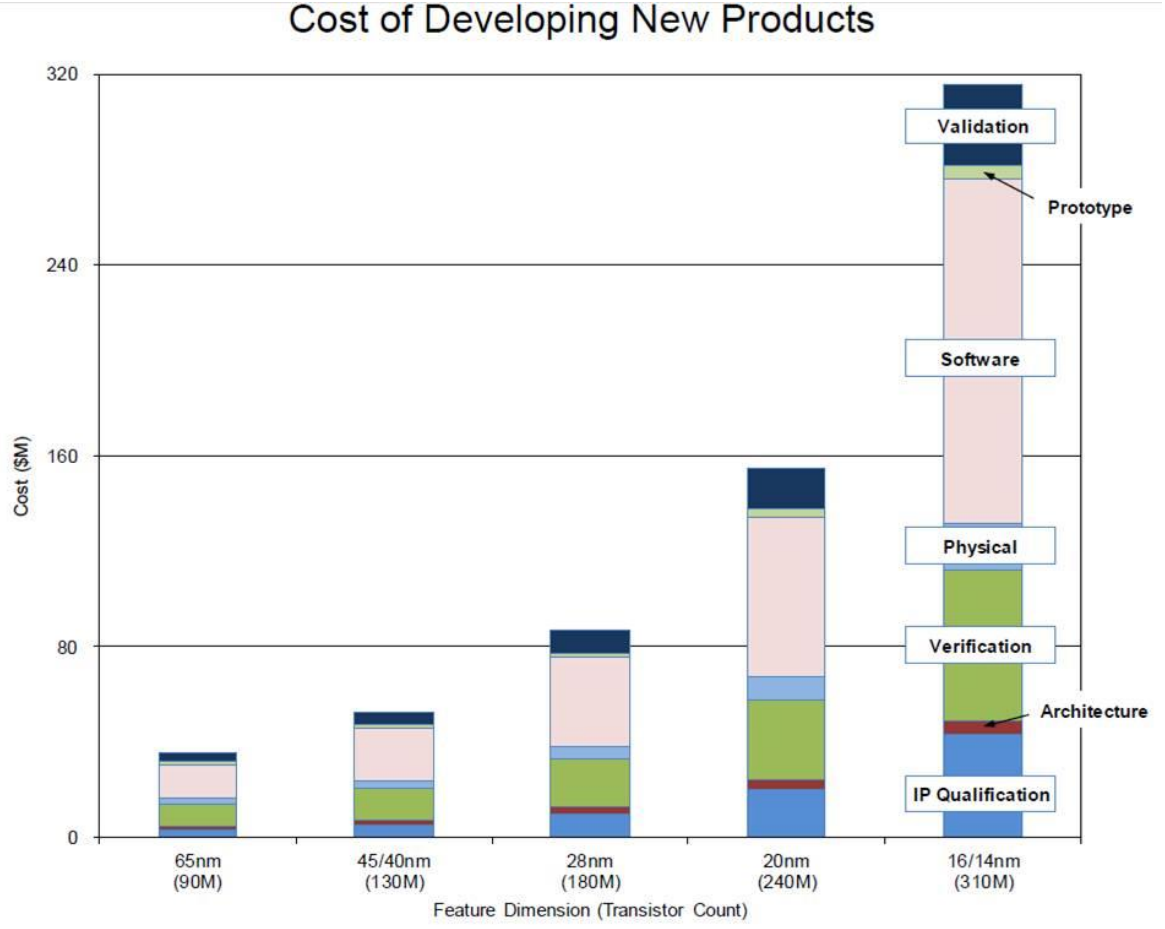
Reduce R&D Cost

Increase units of chips sold

Stay in old geometry longer by decomposition

Improve architecture

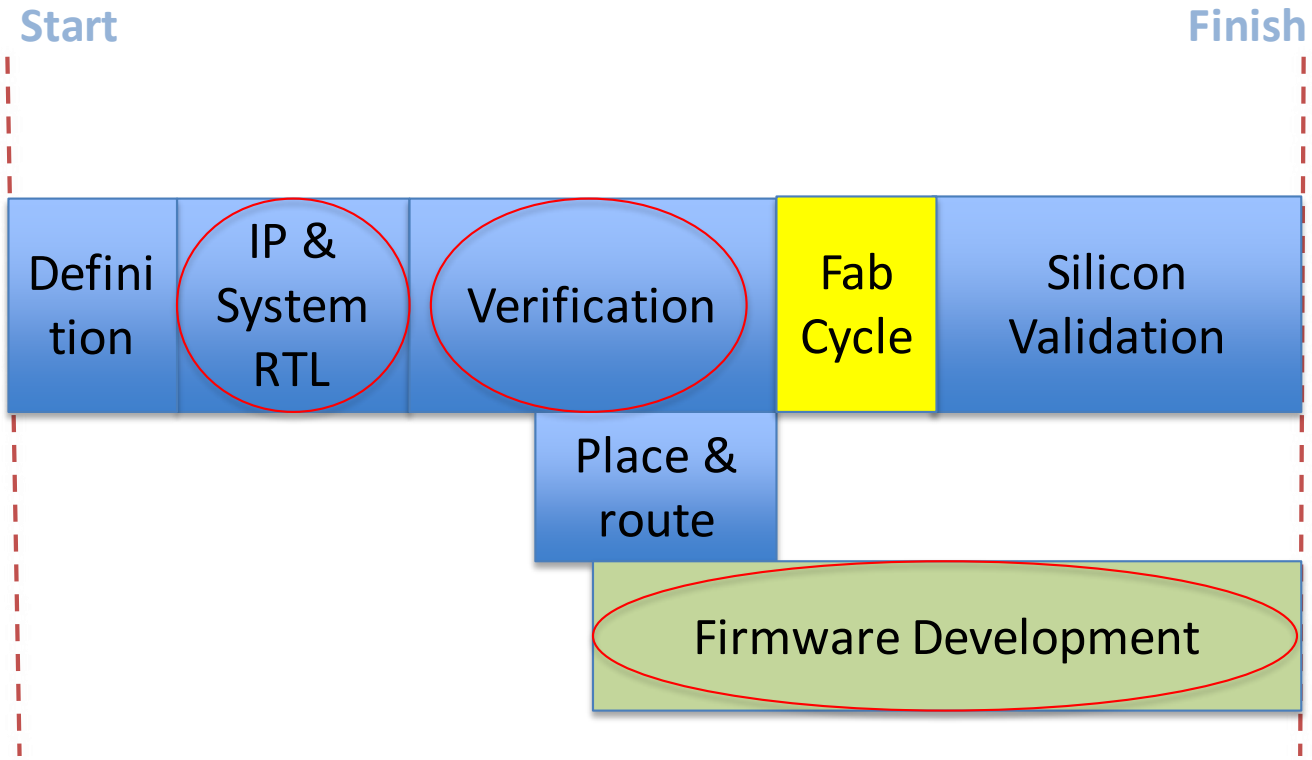
Reduce R&D Cost



R&D Cost ≈ Man-Years

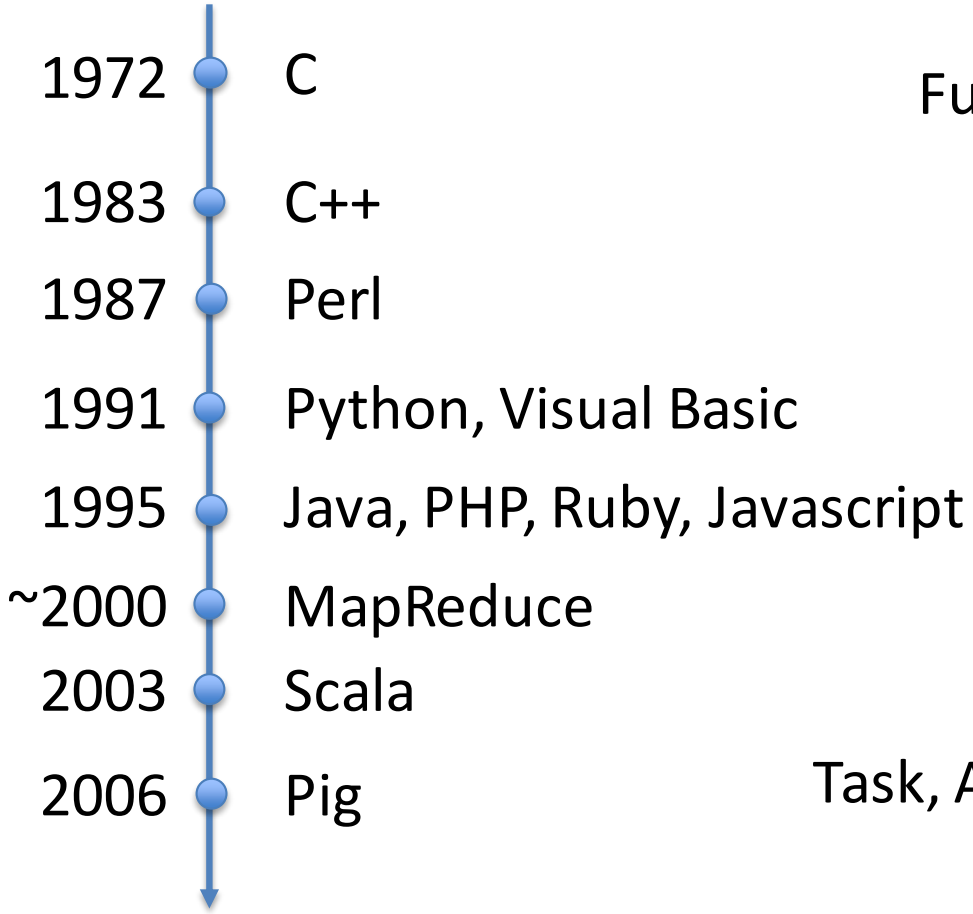
- Make design flow simpler
- Reduce silicon re-spin
- Shorten software development cycle

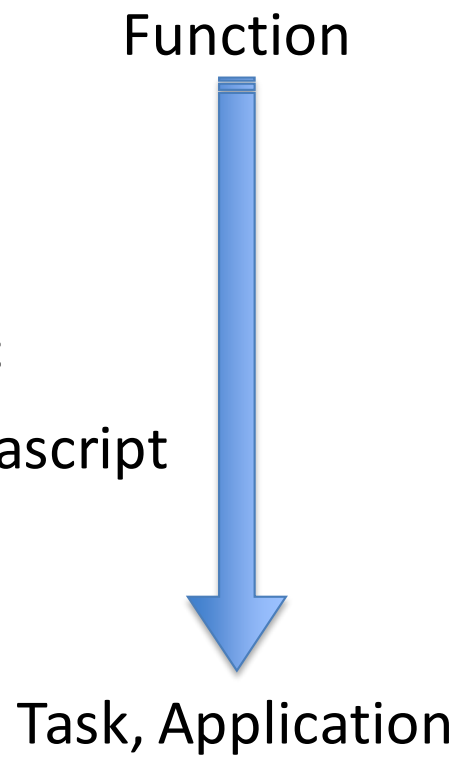
Typical SOC Design Cycle



Enablers Behinds Software Industry

Higher level languages





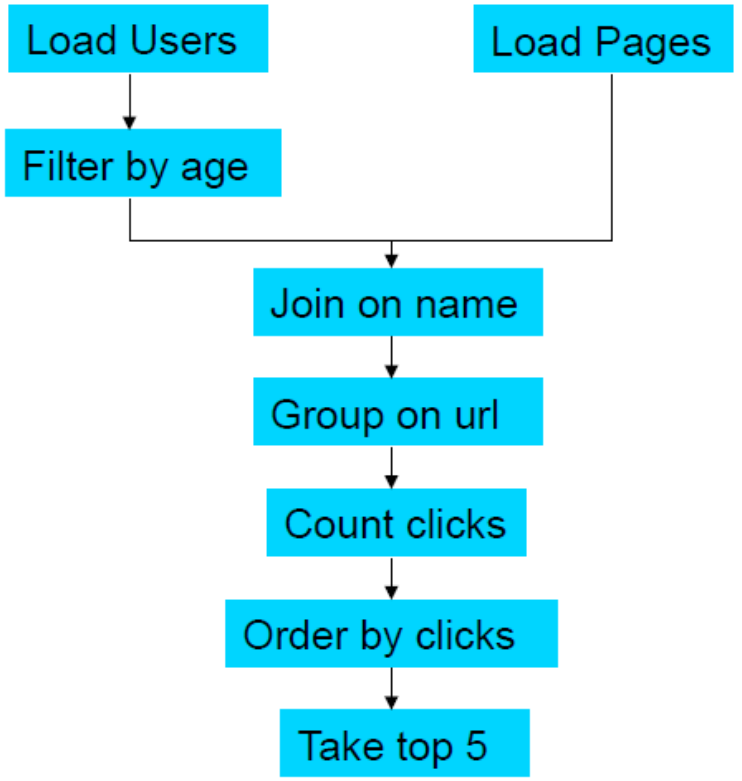
Open Source

- Linux
- Andriod
- Apache
- OpenStack
- Caffe
- …

Learn From Software Industry

Software programming has moved up to more abstract layers – e.g., from Hadoop to Pig

Suppose you have user data in one file, website data in another, and you need to find the top 5 most visited pages by users aged 18 - 25.



Example from http://wiki.apache.org/pig-data/attachments/PigTalksPapers/attachments/ApacheConEurope09.ppt

MapReduce is Power, But Still Hard to Program

```
import java.io.IOException;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;

import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.io.Writable;
import org.apache.hadoop.io.WritableComparable;
import org.apache.hadoop.mapred.FileInputFormat;
import org.apache.hadoop.mapred.FileOutputFormat;
import org.apache.hadoop.mapred.JobConf;
import org.apache.hadoop.mapred.KeyValueTextInputFormat;
import org.apache.hadoop.mapred.Mapper;
import org.apache.hadoop.mapred.MapReduceBase;
import org.apache.hadoop.mapred.OutputCollector;
import org.apache.hadoop.mapred.RecordReader;
import org.apache.hadoop.mapred.Reducer;
import org.apache.hadoop.mapred.Reporter;
import org.apache.hadoop.mapred.SequenceFileInputFormat;
import org.apache.hadoop.mapred.SequenceFileOutputFormat;
import org.apache.hadoop.mapred.TextInputFormat;
import org.apache.hadoop.mapred.jobcontrol.Job;
import org.apache.hadoop.mapred.jobcontrol.JobControl;
import org.apache.hadoop.mapred.lib.IdentityMapper;

public class MRExample {
    public static class LoadPages extends MapReduceBase
        implements Mapper<LongWritable, Text, Text, Text> {

        public void map(LongWritable k, Text val,
                OutputCollector<Text, Text> oc,
                Reporter reporter) throws IOException {
            // Pull the key out
            String line = val.toString();
            int firstComma = line.indexOf(',');
            String key = line.substring(0, firstComma);
            String value = line.substring(firstComma + 1);
            Text outKey = new Text(key);
            // Prepend an index to the value so we know which file
            // it came from.
            Text outVal = new Text("1" + value);
            oc.collect(outKey, outVal);
        }
    }
    public static class LoadAndFilterUsers extends MapReduceBase
        implements Mapper<LongWritable, Text, Text, Text> {

        public void map(LongWritable k, Text val,
                OutputCollector<Text, Text> oc,
                Reporter reporter) throws IOException {
            // Pull the key out
            String line = val.toString();
            int firstComma = line.indexOf(',');
            String value = line.substring(firstComma + 1);
            int age = Integer.parseInt(value);
            if (age < 18 || age > 25) return;
            String key = line.substring(0, firstComma);
            Text outKey = new Text(key);
            // Prepend an index to the value so we know which file
            // it came from.
            Text outVal = new Text("2" + value);
            oc.collect(outKey, outVal);
        }
    }
    public static class Join extends MapReduceBase
        implements Reducer<Text, Text, Text, Text> {

        public void reduce(Text key,
                Iterator<Text> iter,
                OutputCollector<Text, Text> oc,
                Reporter reporter) throws IOException {
            // For each value, figure out which file it's from and
store it
            // accordingly.
            List<String> first = new ArrayList<String>();
            List<String> second = new ArrayList<String>();

            while (iter.hasNext()) {
                Text t = iter.next();
                String value = t.toString();
                if (value.charAt(0) == '1')
first.add(value.substring(1));
                else second.add(value.substring(1));
                reporter.setStatus("OK");
            }

            // Do the cross product and collect the values
            for (String s1 : first) {
                for (String s2 : second) {
                    String outval = key + "," + s1 + "," + s2;
                    oc.collect(null, new Text(outval));
                    reporter.setStatus("OK");
                }
            }
        }
    }
    public static class LoadJoined extends MapReduceBase
        implements Mapper<Text, Text, Text, LongWritable> {

        public void map(
                Text k,
                Text val,
                OutputCollector<Text, LongWritable> oc,
                Reporter reporter) throws IOException {
            // Find the url
            String line = val.toString();
            int firstComma = line.indexOf(',');
            int secondComma = line.indexOf(',', firstComma);
            String key = line.substring(firstComma, secondComma);
            // drop the rest of the record, I don't need it anymore,
            // just pass a 1 for the combiner/reducer to sum instead.
            Text outKey = new Text(key);
            oc.collect(outKey, new LongWritable(1L));
        }
    }
    public static class ReduceUrls extends MapReduceBase
        implements Reducer<Text, LongWritable, WritableComparable,
Writable> {

        public void reduce(
                Text key,
                Iterator<LongWritable> iter,
                OutputCollector<WritableComparable, Writable> oc,
                Reporter reporter) throws IOException {
            // Add up all the values we see

            long sum = 0;
            while (iter.hasNext()) {
                sum += iter.next().get();
                reporter.setStatus("OK");
            }

            oc.collect(key, new LongWritable(sum));
        }
    }
    public static class LoadClicks extends MapReduceBase
        implements Mapper<WritableComparable, Writable, LongWritable,
Text> {

        public void map(
                WritableComparable key,
                Writable val,
                OutputCollector<LongWritable, Text> oc,
                Reporter reporter) throws IOException {
            oc.collect((LongWritable)val, (Text)key);
        }
    }
    public static class LimitClicks extends MapReduceBase
        implements Reducer<LongWritable, Text, LongWritable, Text> {

        int count = 0;
        public void reduce(
            LongWritable key,
            Iterator<Text> iter,
            OutputCollector<LongWritable, Text> oc,
            Reporter reporter) throws IOException {

            // Only output the first 100 records
            while (count < 100 && iter.hasNext()) {
                oc.collect(key, iter.next());
                count++;
            }
        }
    }
    public static void main(String[] args) throws IOException {
        JobConf lp = new JobConf(MRExample.class);
        lp.setJobName("Load Pages");
        lp.setInputFormat(TextInputFormat.class);
        lp.setOutputKeyClass(Text.class);
        lp.setOutputValueClass(Text.class);
        lp.setMapperClass(LoadPages.class);
        FileInputFormat.addInputPath(lp, new
Path("/user/gates/pages"));
        FileOutputFormat.setOutputPath(lp,
            new Path("/user/gates/tmp/indexed_pages"));
        lp.setNumReduceTasks(0);
        Job loadPages = new Job(lp);

        JobConf lfu = new JobConf(MRExample.class);
        lfu.setJobName("Load and Filter Users");
        lfu.setInputFormat(TextInputFormat.class);
        lfu.setOutputKeyClass(Text.class);
        lfu.setOutputValueClass(Text.class);
        lfu.setMapperClass(LoadAndFilterUsers.class);
        FileInputFormat.addInputPath(lfu, new
Path("/user/gates/users"));
        FileOutputFormat.setOutputPath(lfu,
            new Path("/user/gates/tmp/filtered_users"));
        lfu.setNumReduceTasks(0);
        Job loadUsers = new Job(lfu);

        JobConf join = new JobConf(MRExample.class);
        join.setJobName("Join Users and Pages");
        join.setInputFormat(KeyValueTextInputFormat.class);
        join.setOutputKeyClass(Text.class);
        join.setOutputValueClass(Text.class);
        join.setMapperClass(IdentityMapper.class);
        join.setReducerClass(Join.class);
        FileInputFormat.addInputPath(join, new
Path("/user/gates/tmp/indexed_pages"));
        FileInputFormat.addInputPath(join, new
Path("/user/gates/tmp/filtered_users"));
        FileOutputFormat.setOutputPath(join, new
Path("/user/gates/tmp/joined"));
        join.setNumReduceTasks(50);
        Job joinJob = new Job(join);
        joinJob.addDependingJob(loadPages);
        joinJob.addDependingJob(loadUsers);

        JobConf group = new JobConf(MRExample.class);
        group.setJobName("Group URLs");
        group.setInputFormat(KeyValueTextInputFormat.class);
        group.setOutputKeyClass(Text.class);
        group.setOutputValueClass(LongWritable.class);
        group.setOutputFormat(SequenceFileOutputFormat.class);
        group.setMapperClass(LoadJoined.class);
        group.setCombinerClass(ReduceUrls.class);
        group.setReducerClass(ReduceUrls.class);
        FileInputFormat.addInputPath(group, new
Path("/user/gates/tmp/joined"));
        FileOutputFormat.setOutputPath(group, new
Path("/user/gates/tmp/grouped"));
        group.setNumReduceTasks(50);
        Job groupJob = new Job(group);
        groupJob.addDependingJob(joinJob);

        JobConf top100 = new JobConf(MRExample.class);
        top100.setJobName("Top 100 sites");
        top100.setInputFormat(SequenceFileInputFormat.class);
        top100.setOutputKeyClass(LongWritable.class);
        top100.setOutputValueClass(Text.class);
        top100.setOutputFormat(SequenceFileOutputFormat.class);
        top100.setMapperClass(LoadClicks.class);
        top100.setCombinerClass(LimitClicks.class);
        top100.setReducerClass(LimitClicks.class);
        FileInputFormat.addInputPath(top100, new
Path("/user/gates/tmp/grouped"));
        FileOutputFormat.setOutputPath(top100, new
Path("/user/gates/top100sitesforusers18to25"));
        top100.setNumReduceTasks(1);
        Job limit = new Job(top100);
        limit.addDependingJob(groupJob);

        JobControl jc = new JobControl("Find top 100 sites for users
18 to 25");
        jc.addJob(loadPages);
        jc.addJob(loadUsers);
        jc.addJob(joinJob);
        jc.addJob(groupJob);
        jc.addJob(limit);
        jc.run();
    }
}
```

Example from http://wiki.apache.org/pig-data/attachments/PigTalksPapers/attachments/ApacheConEurope09.ppt

Much Simpler In PigLatin

```
Users    = load ‘users’ as (name, age);
Filtered = filter Users by
                  age >= 18 and age <= 25;
Pages    = load ‘pages’ as (user, url);
Joined   = join Filtered by name, Pages by user;
Grouped  = group Joined by url;
Summed   = foreach Grouped generate group,
                   count(Joined) as clicks;
Sorted   = order Summed by clicks desc;
Top5     = limit Sorted 5;

store Top5 into ‘top5sites’;
```

Example from http://wiki.apache.org/pig-data/attachments/PigTalksPapers/attachments/ApacheConEurope09.ppt

High Level Synthesis

Verilog RTL

```
module Max2(
    input [7:0] io_in0,
    input [7:0] io_in1,
    output[7:0] io_out
);

  wire [7:0] T0;
  wire T1;

  assign io_out = T0;
  assign T0 = T1 ? io_in0 : io_in1;
  assign T1 = io_in1 < io_in0;
endmodule
```

Chisel

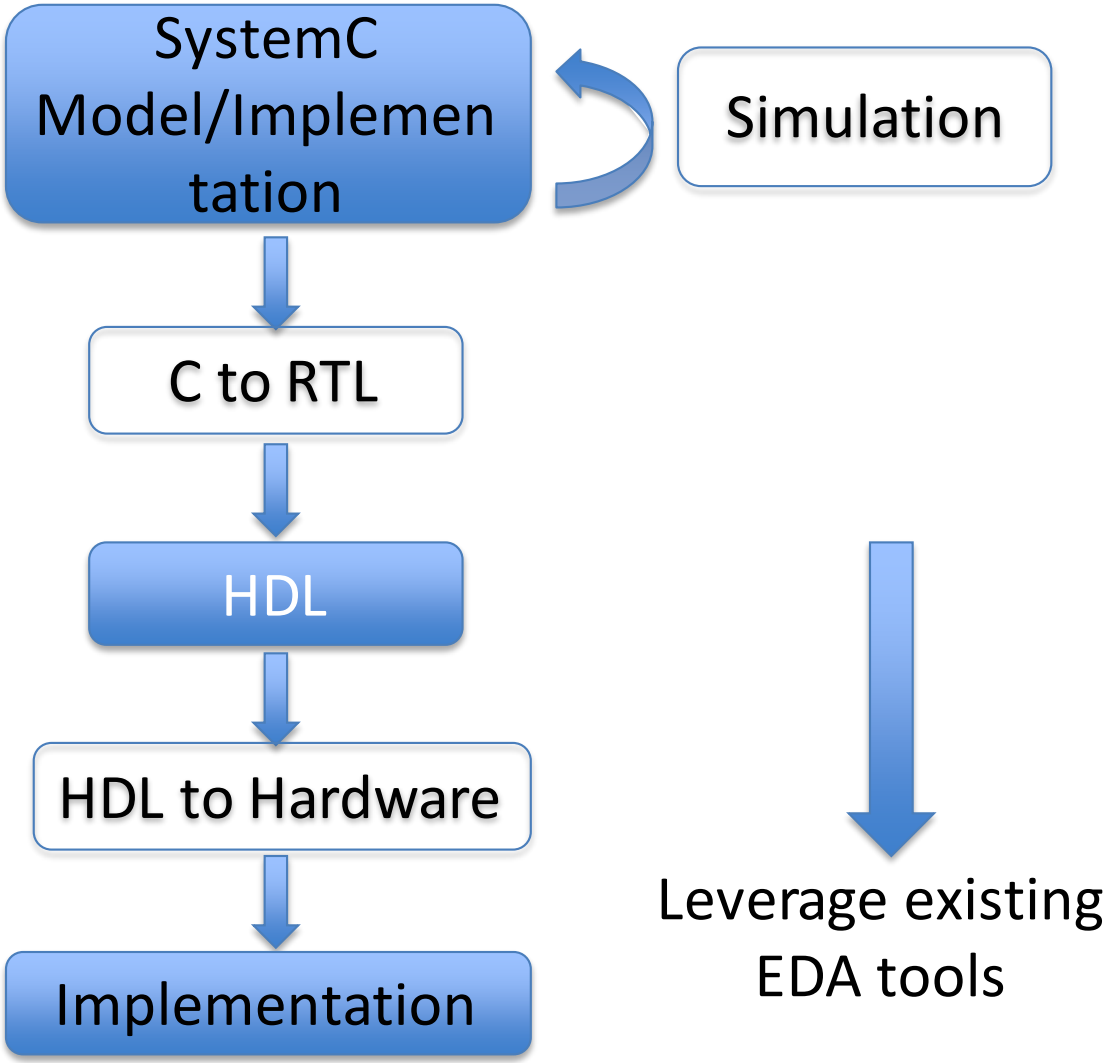
```
import Chisel._

class Max2 extends Module {
  val io = new Bundle {
    val in0 = UInt(INPUT,  8)
    val in1 = UInt(INPUT,  8)
    val out = UInt(OUTPUT, 8)
  }

  io.out := Mux(io.in0 > io.in1, io.in0, io.in1)
}
```

https://chisel.eecs.berkeley.edu/

Enable High Level Synthesis

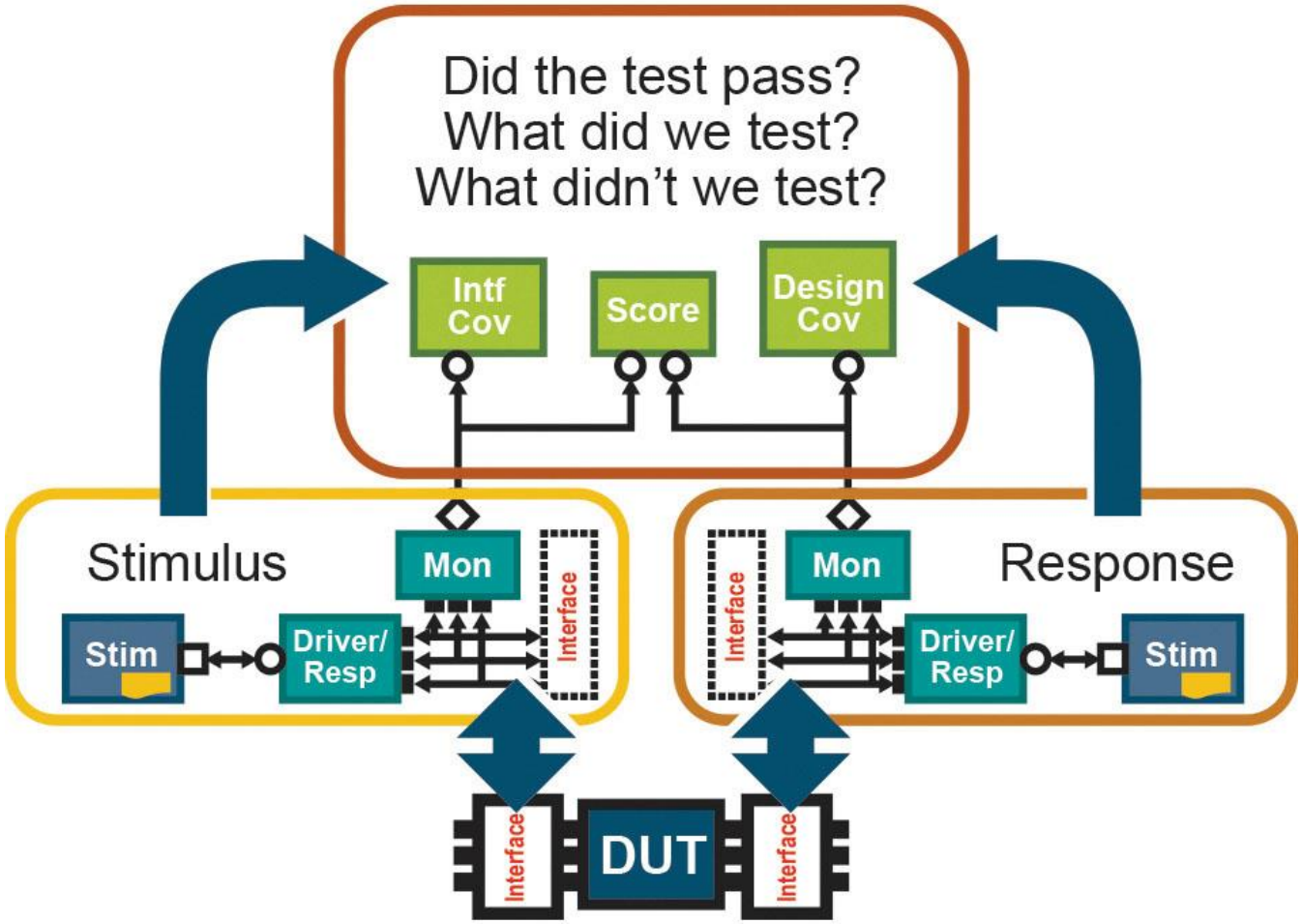


- ✓ Simulation tools
- ✓ C/Matlab/Chisel to RTL tools
- × Abundant libraries
- × Easy ECO tools
- × Education

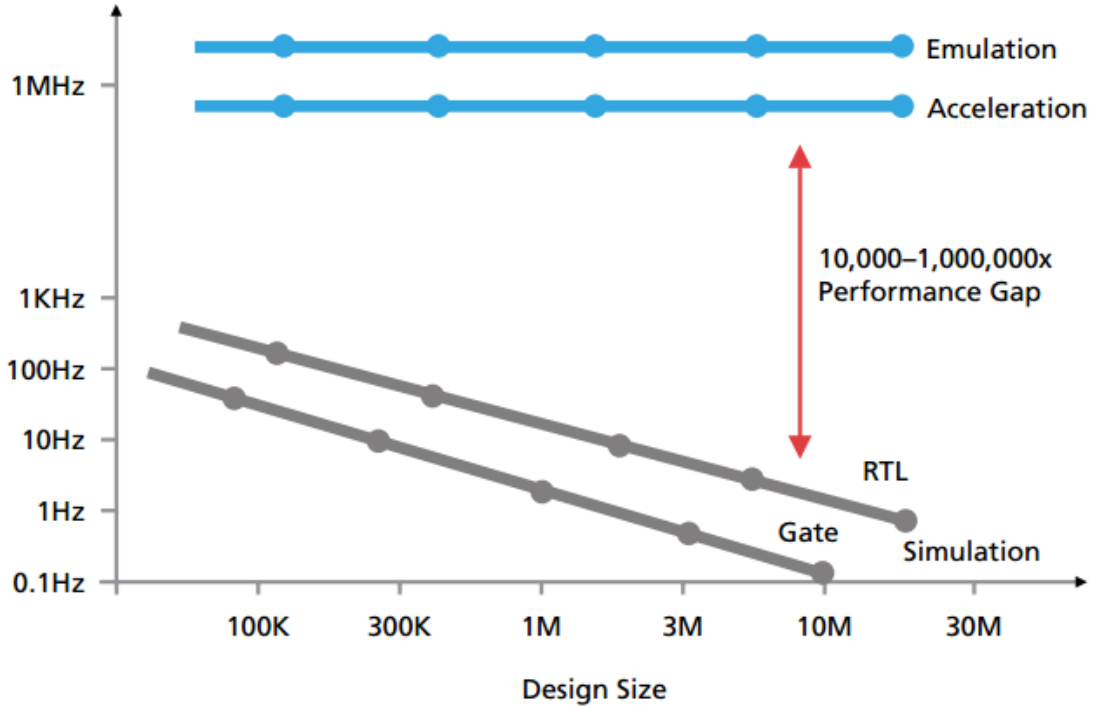
Reduce Silicon Re-spin – UVM Verification

- From direct simulation to constrained random verification
- Transaction level modeling (TLM) and built-in randomization capability reduces engineers' effort.
- Leave heavy computation work to machines
- Maximize component reuse

- Standardization spurs tool development and adoption



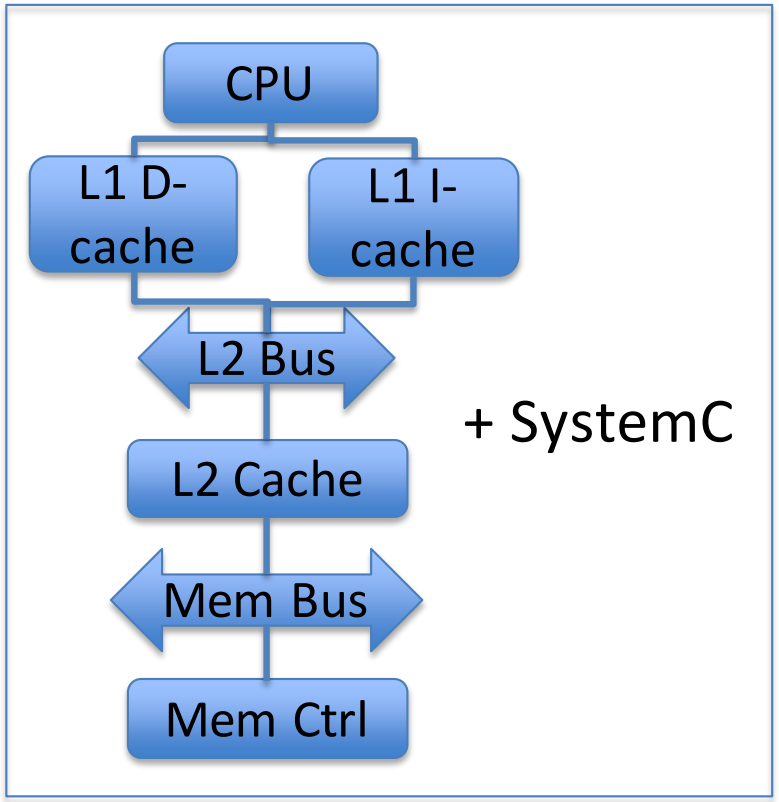
Reduce Silicon Re-spin – Emulation

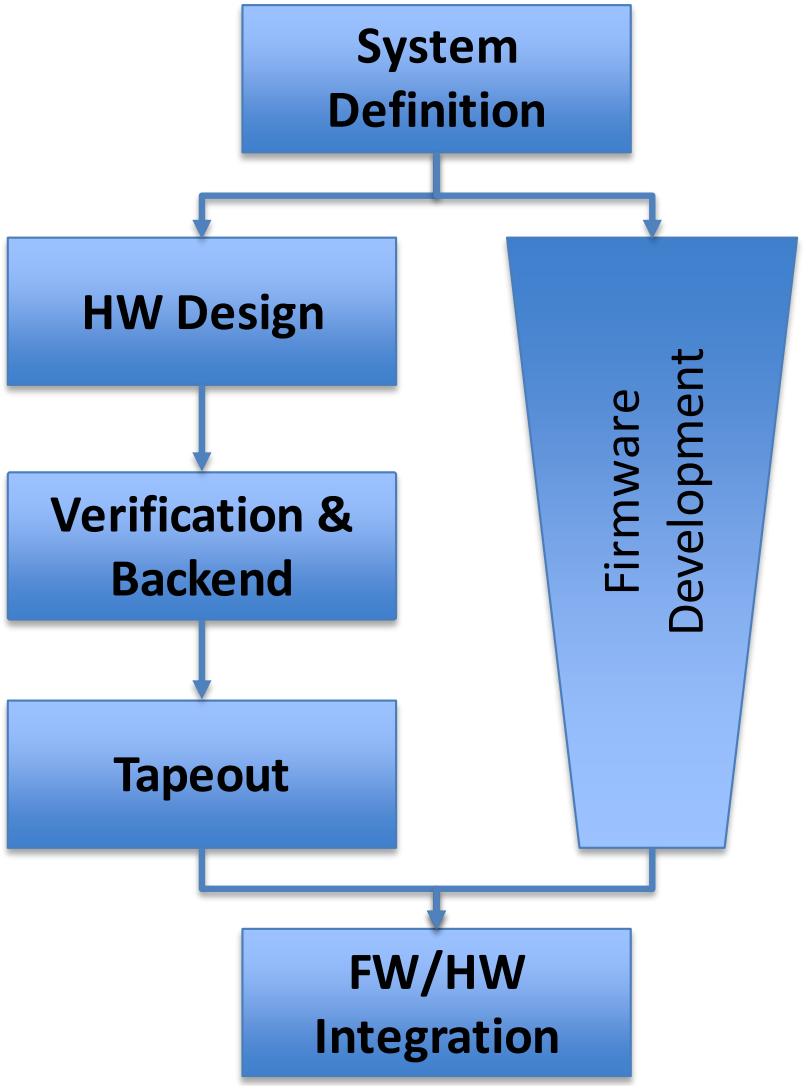


1,000,000x speed up → >1,000,000x test cases → Less chance for silicon re-spin

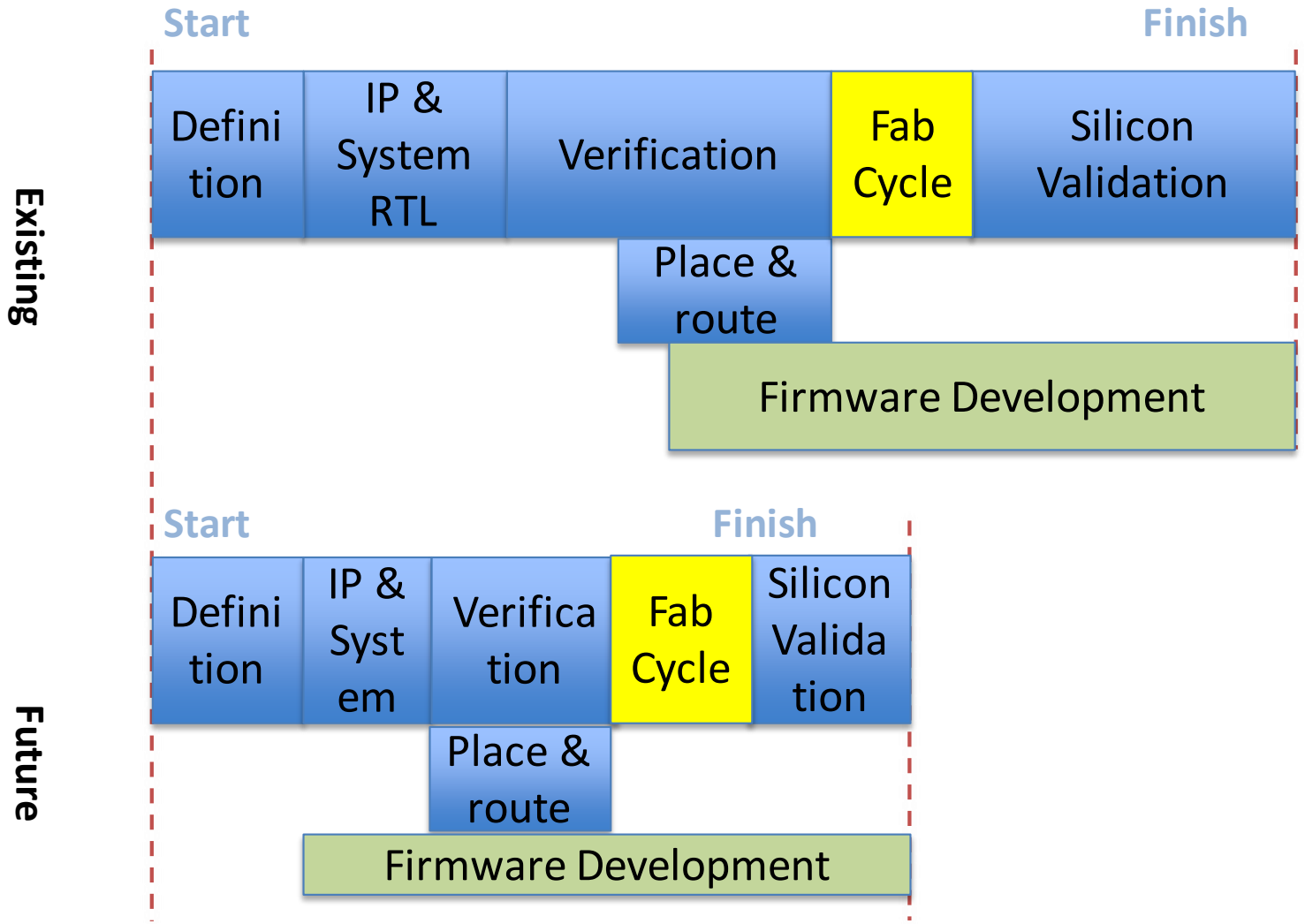
Shorten Firmware Development with Virtual Models

Example





Design Cycle Can Be Shortened to Save Cost



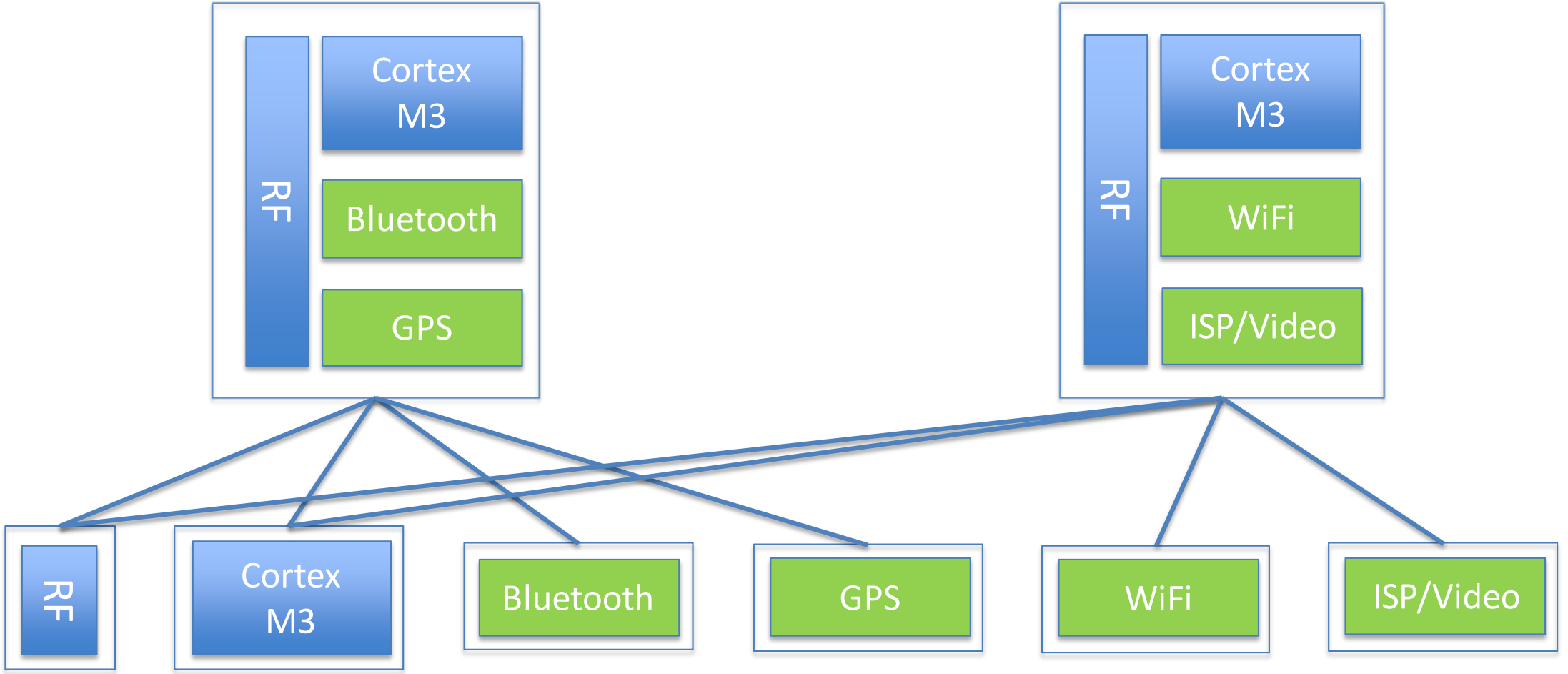
Reduce R&D Cost

Increase units of chips sold

Stay in old geometry longer by decomposition

Improve architecture

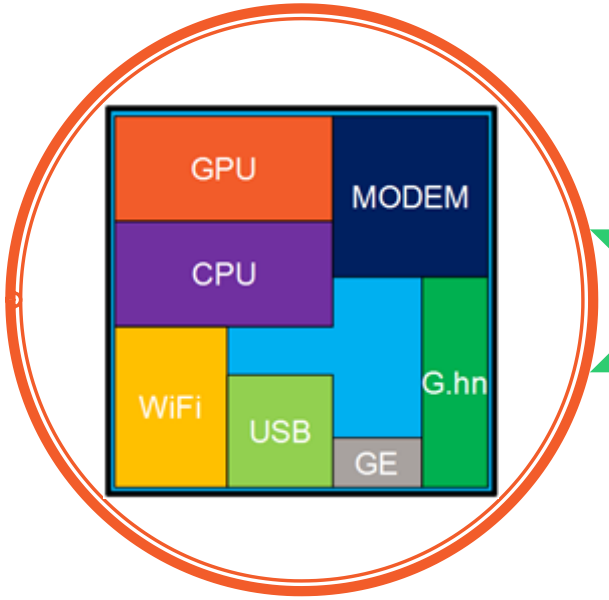
Increase # of Units by Finding "Common Denominators"



What about a Lego-like design?

Modular Chip (MoChi™)

Single-Die SoC

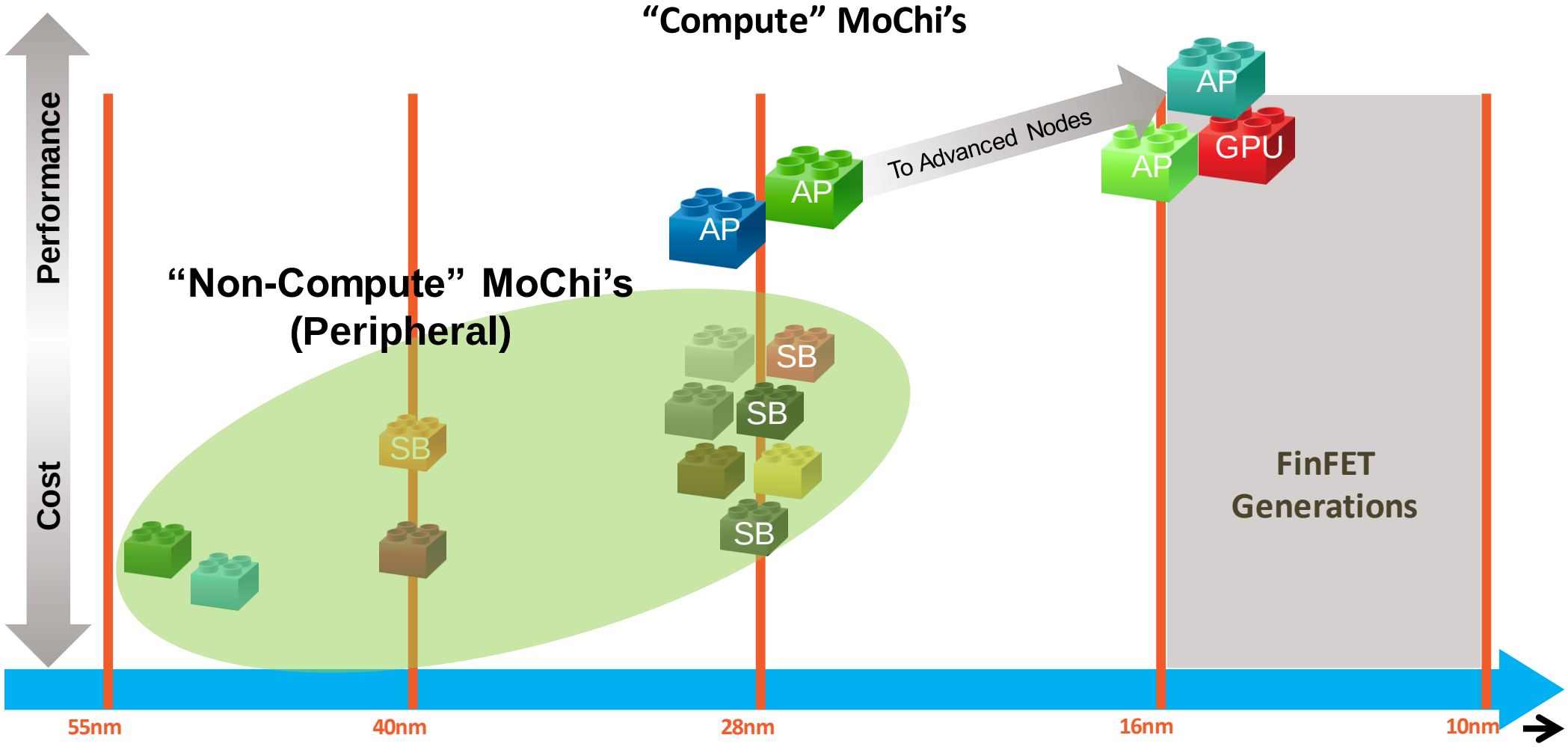


(Conventional)

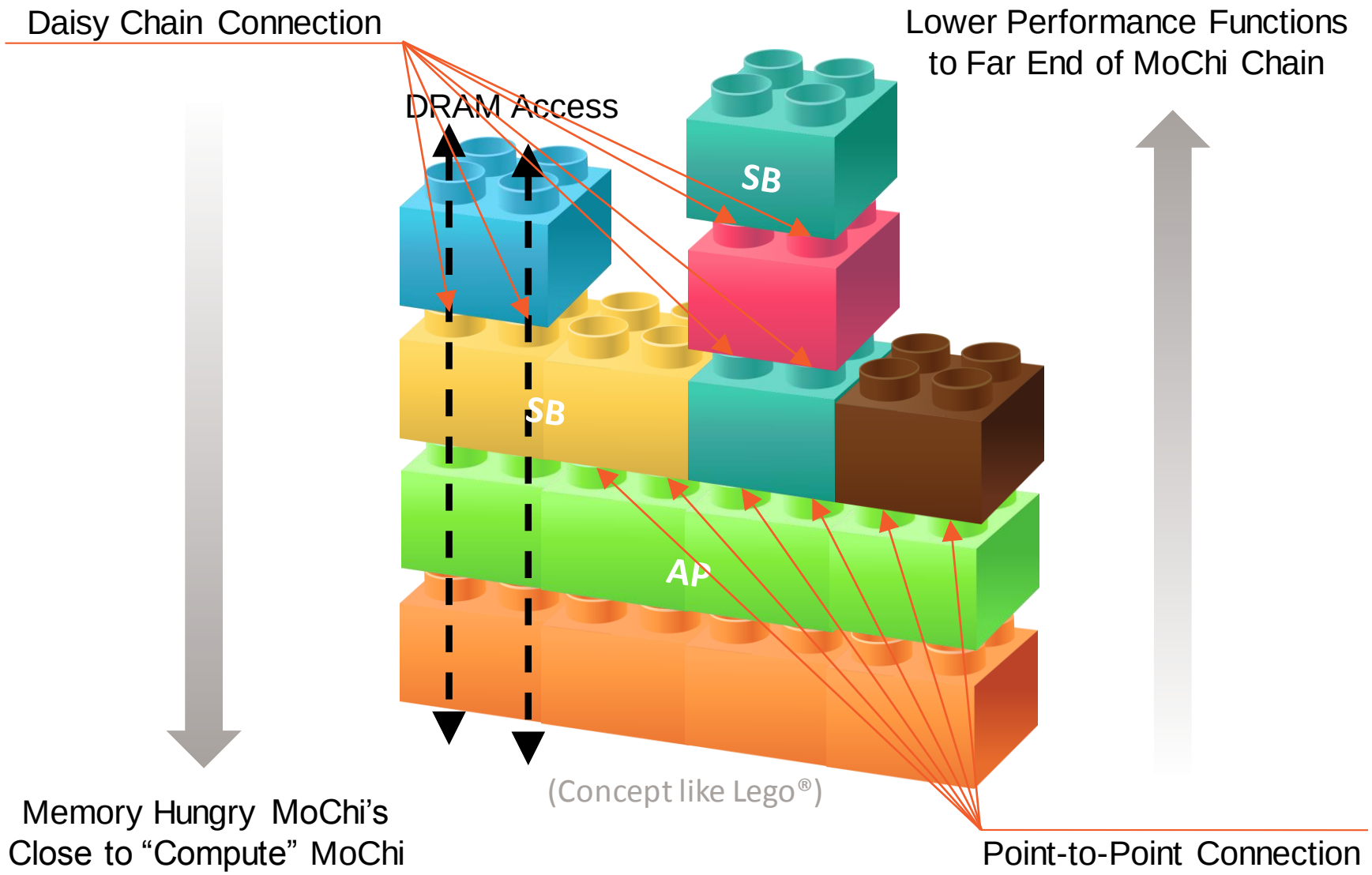
MoChi Blocks

(Concept like Lego®)

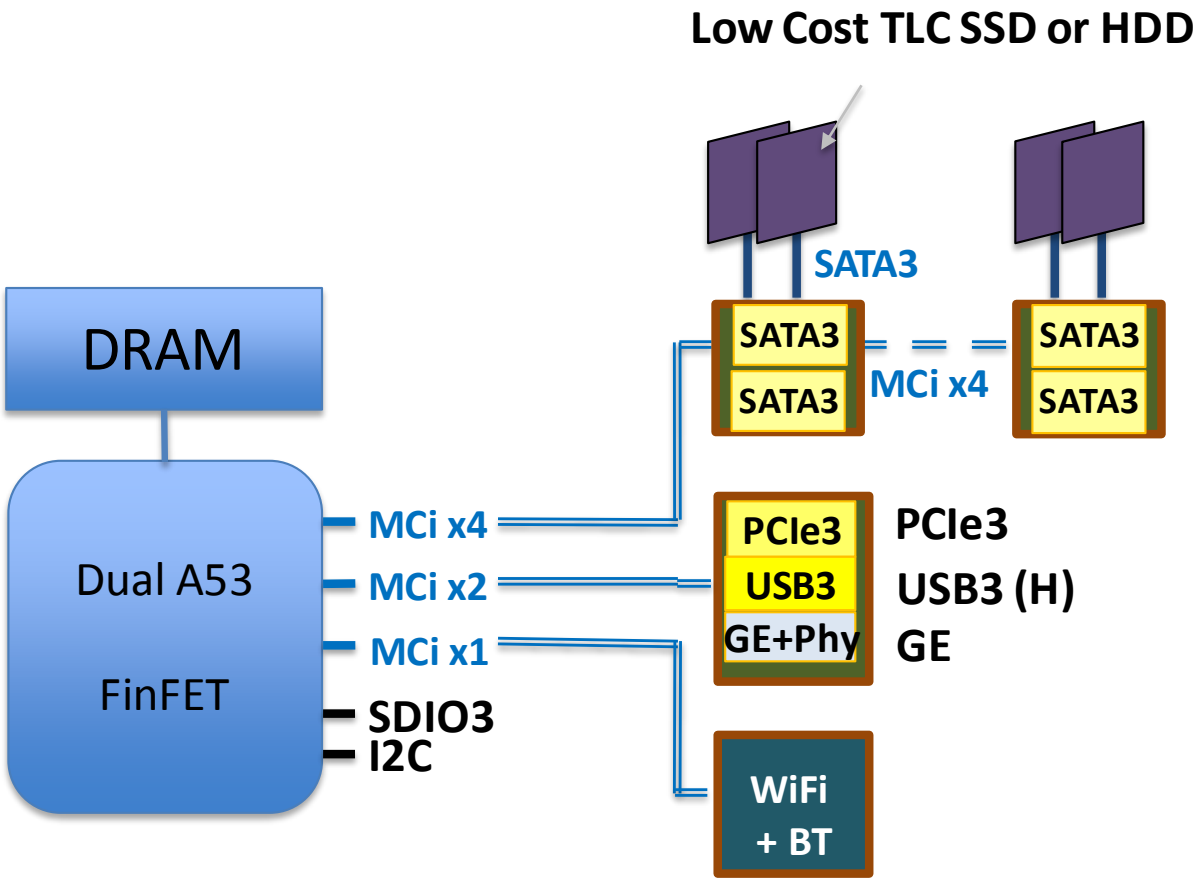
Optimize Different Blocks At Different Technology Nodes



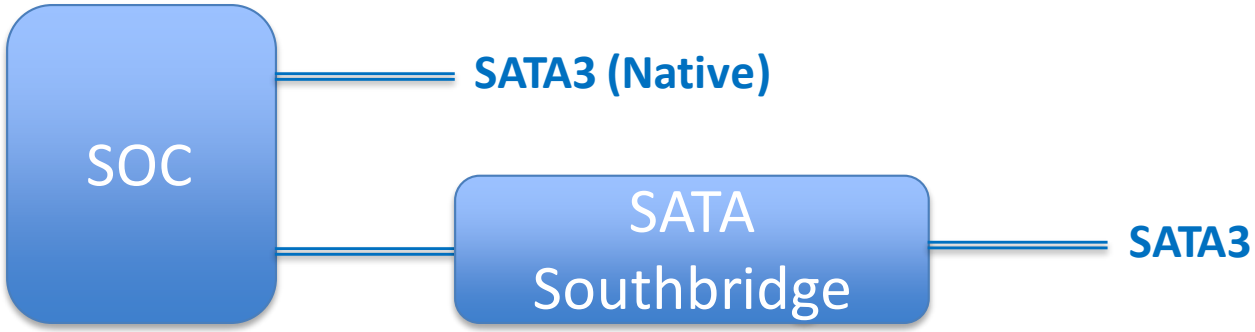
Key Challenge Is How To Connect MoChi Blocks



MoChi Example – Network Attached Storage



MoChi Interconnect (MCi) Is Transparent



	READ bandwidth (MB/s)	WRITE Bandwidth (MB/s)	R/W Mixed Bandwidth (MB/s)
Native SATA	515	490	500
SATA through MCi	492	510	501

Architecture Improvement Becomes More Important

- Example – CNN looks feasible in mobile phone with SqueezeNet

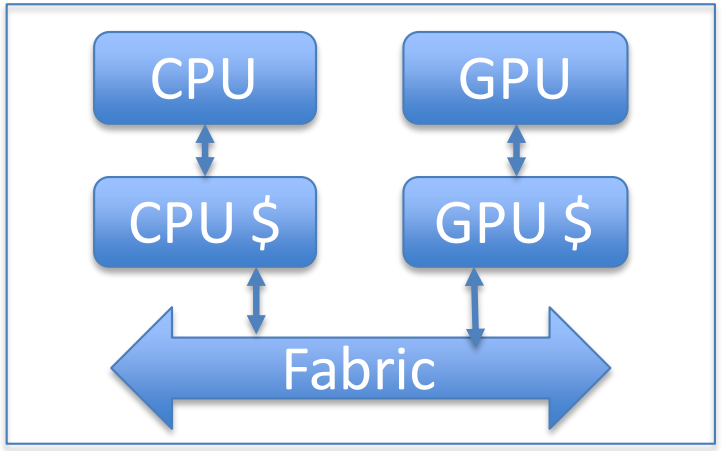
CNN architecture	Compression Approach	Data Type	Original → Compressed Model Size	Reduction in Model Size vs. AlexNet	Top-1 ImageNet Accuracy	Top-5 ImageNet Accuracy
AlexNet	None (baseline)	32 bit	240MB	1x	57.2%	80.3%
AlexNet	SVD [5]	32 bit	240MB → 48MB	5x	56.0%	79.4%
AlexNet	Network Pruning [11]	32 bit	240MB → 27MB	9x	57.2%	80.3%
AlexNet	Deep Compression [10]	5-8 bit	240MB → 6.9MB	35x	57.2%	80.3%
SqueezeNet (ours)	None	32 bit	4.8MB	**50x**	57.5%	80.3%
SqueezeNet (ours)	Deep Compression	8 bit	4.8MB → 0.66MB	**363x**	57.5%	80.3%
SqueezeNet (ours)	Deep Compression	6 bit	4.8MB → 0.47MB	**510x**	57.5%	80.3%

Source: SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size
F. Iandola, etc., arXiv:1602.07360v3 [cs.CV] 6 Apr 2016

Heterogeneous Computing

	Big Data	Deep Learning	IoT
CPU GPU	✓	✓	
CPU GPU FPGA	✓	✓	
CPU GPU Sensor DSP			✓
CPU GPU TPU		✓	

Challenges In Heterogeneous Computing



- Efficient movement of data
  - Coherency
  - Precision
  - Bandwidth
- Software tools and programming paradigm
  - Who does job dispatch? Compiler, OS, HW?
  - Unified programming language like OpenCL and CUDA?

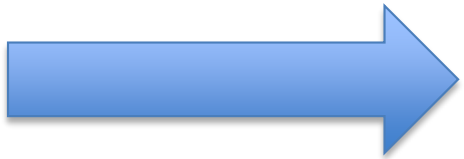
Summary

Moore's law is not dead, just not affordable

Moving down the technology node can improve Performance and Power, not Price

Harder for smaller company to compete

SoC designers need to address the cost aspect of Moore's Law



Shorten design cycles by relying more on high level designs

Learn from software industry, is Open Source possible?

Focus on architectural innovations

Modular design of SOC to increase design reuse