E225C – Lecture 3
System on a Chip Design

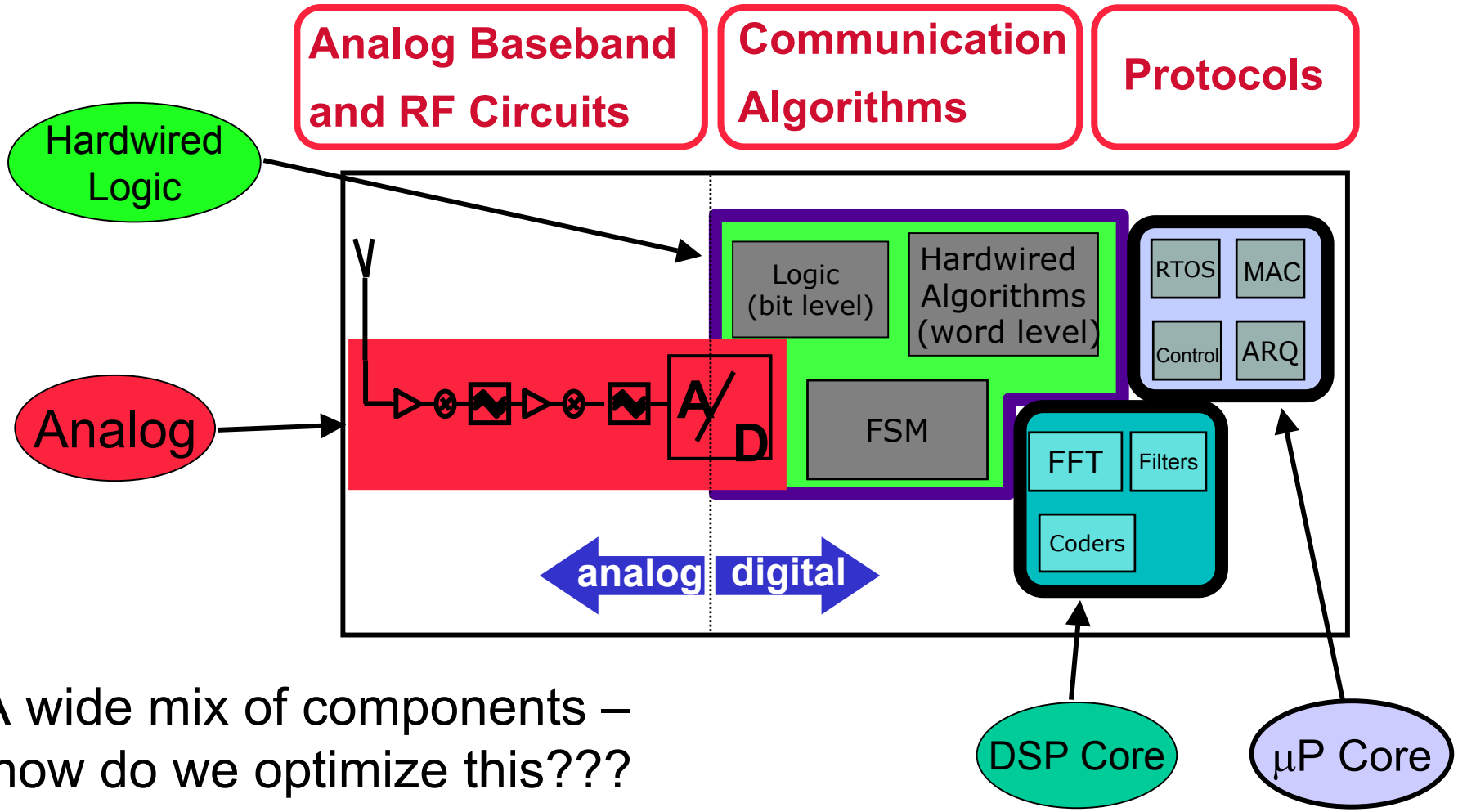
Bob Brodersen

What is an SoC?

Let me define what I think it is….

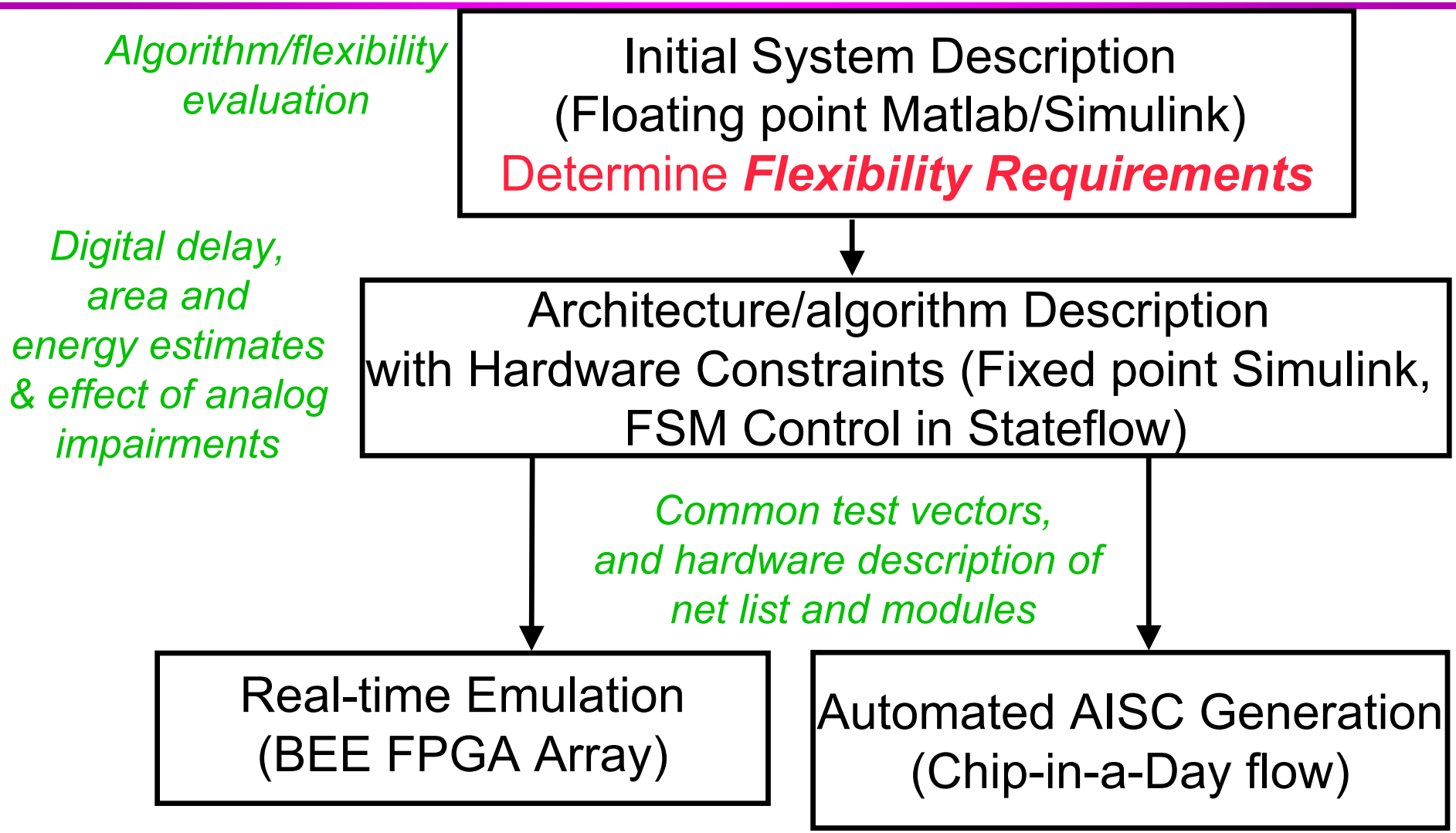
"A chip designed for "complete" system functionality that incorporates a heterogeneous mix of processing and computation architectures"

A Wireless System – Typical SOC Design



A wide mix of components – how do we optimize this???

An SOC Design Flow with Prototyping



The Issues I am Going to Address

- How much flexibility is needed and how best to include it…
- A single system description including interaction between the analog and digital domains
- “Realtime” SOC prototyping
- Automated ASIC design flow

Flexibility

- Determining how much to include and how to do it in the most efficient way possible
- Claims (to be shown)
  - There are good reasons for flexibility
  - The “cost” of flexibility is orders of magnitude of inefficiency over an optimized solution
  - There are many different ways to provide flexibility

Good reasons for flexibility

- One design for a number of SoC customers – more sales volume
- Customers able to provide added value and uniqueness
- Unsure of specification or can't make a decision
- Backwards compatibility with debugged software
- Risk, cost and time of implementing hardwired solutions

Important to note: these are business, not technical reasons

So, what is the cost of flexibility?

We need technical metrics that we can look to compare flexible and non-flexible implementations

- A power metric because of thermal limitations
- An energy metric for portable operation
- A cost metric related to the area of the chip
- Performance (computational throughput)

Lets use metrics normalized to the amount of computation being performed – so now lets define computation

Definitions…

Computation

- Operation = OP=algorithmically interesting computation (i.e. multiply, add, delay)
- MOPS = Millions of OP's per Second
- N_{op}=Number of parallel OP's in ***each*** clock cycle

Power

- P_{chip}= Total power of chip = $A_{chip}*C_{sw}*(V_{dd})^2 * f_{clk}$
- C_{sw} = Switched Capacitance/mm²
  $= P_{chip} /(A_{chip} *V_{dd}^2 * f_{clk})$

Area

- A_{chip} = Total area of chip
- A_{op} = Average area of each operation = A_{chip}/N_{op}

Energy Efficiency Metric: MOPS/mW

How much computing (number of operations) can we can do with a finite energy source (e.g. battery)?

$$\text{Energy Efficiency} = \frac{\text{Number of useful operations}}{\text{Energy required}}$$

$$= \frac{\text{\# of Operations}}{\text{NanoJoule}} = \text{OP/nJ}$$

$$= \frac{\text{OP/Sec}}{\text{NanoJoule/Sec}} = \frac{\text{MOPS}}{\text{mW}}$$

$$= \text{Power Efficiency}$$

Energy and Power Efficiency

OP/nJ = MOPS/mW

Interestingly the energy efficiency metric for energy constrained applications (OP/nJ) for a fixed number of operations is the same as that for thermal (power) considerations when maximizing throughput (MOPS/mW).

So lets look at a number of chips to see how these efficiency numbers compare

ISSCC Chips (.18µ-.25µ)

Chip #	Year	Paper	Description
1	1997	10.3	µP - S/390
2	2000	5.2	µP – PPC (SOI)
3	1999	5.2	µP - G5
4	2000	5.6	µP - G6
5	2000	5.1	µP - Alpha
6	1998	15.4	µP - P6
7	1998	18.4	µP - Alpha
8	1999	5.6	µP – PPC
9	1998	18.6	DSP - StrongArm
10	2000	4.2	DSP – Comm

Microprocessors

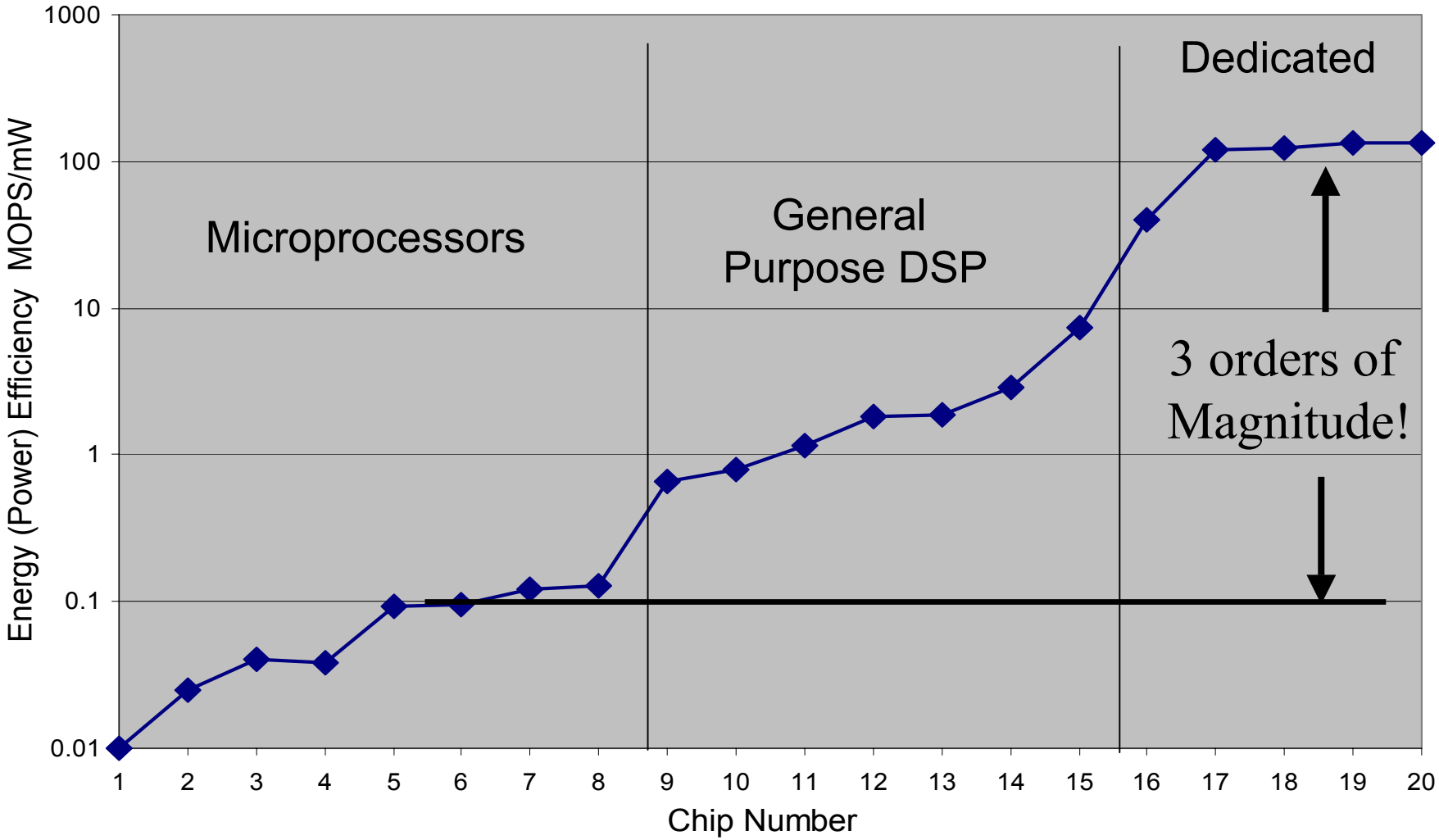
DSP's

Chip #	Year	Paper	Description
11	1998	18.1	DSP -Graphics
12	1998	18.2	DSP - Multimedia
13	2000	14.6	DSP – Multimedia
14	2002	22.1	DSP – Mpeg Decoder
15	1998	18.3	DSP - Multimedia
16	2001	21.2	Encryption Processor
17	2000	14.5	Hearing Aid Processor
18	2000	4.7	FIR for Disk Read Head
19	1998	2.1	MPEG Encoder
20	2002	7.2	802.11a Baseband

DSP's

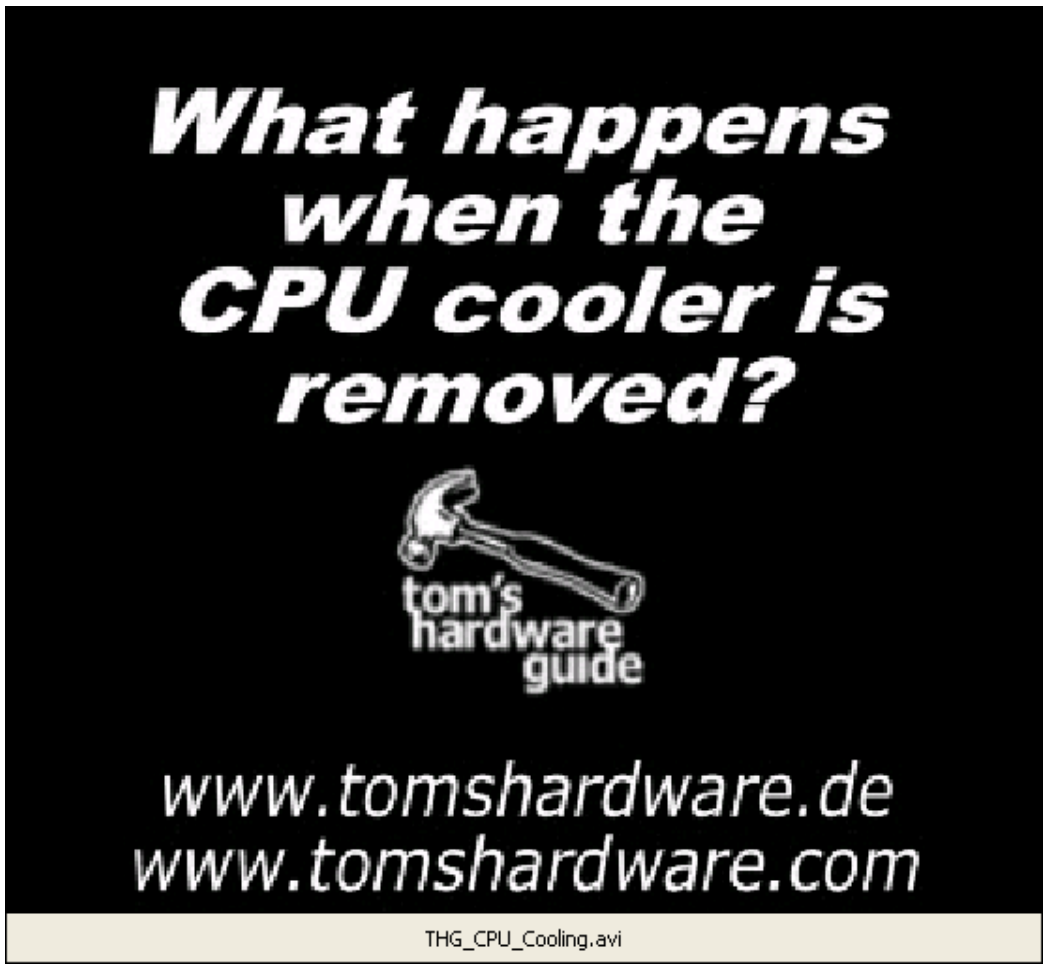
Dedicated

Energy Efficiency (MOPS/mW or OP/nJ)



What does the low efficiency really mean?

The basic processor architecture puts our circuits at the very limit of failure…



Why such a big difference?

Lets look at the components of MOPS/mW.

The operations per second:

$$MOPS = f_{clk} * N_{op}$$

The power:

$$P_{chip} = A_{chip} * C_{sw} * (V_{dd})^2 * f_{clk}$$

The ratio ($MOPS/P_{chip}$) gives the MOPS/mW

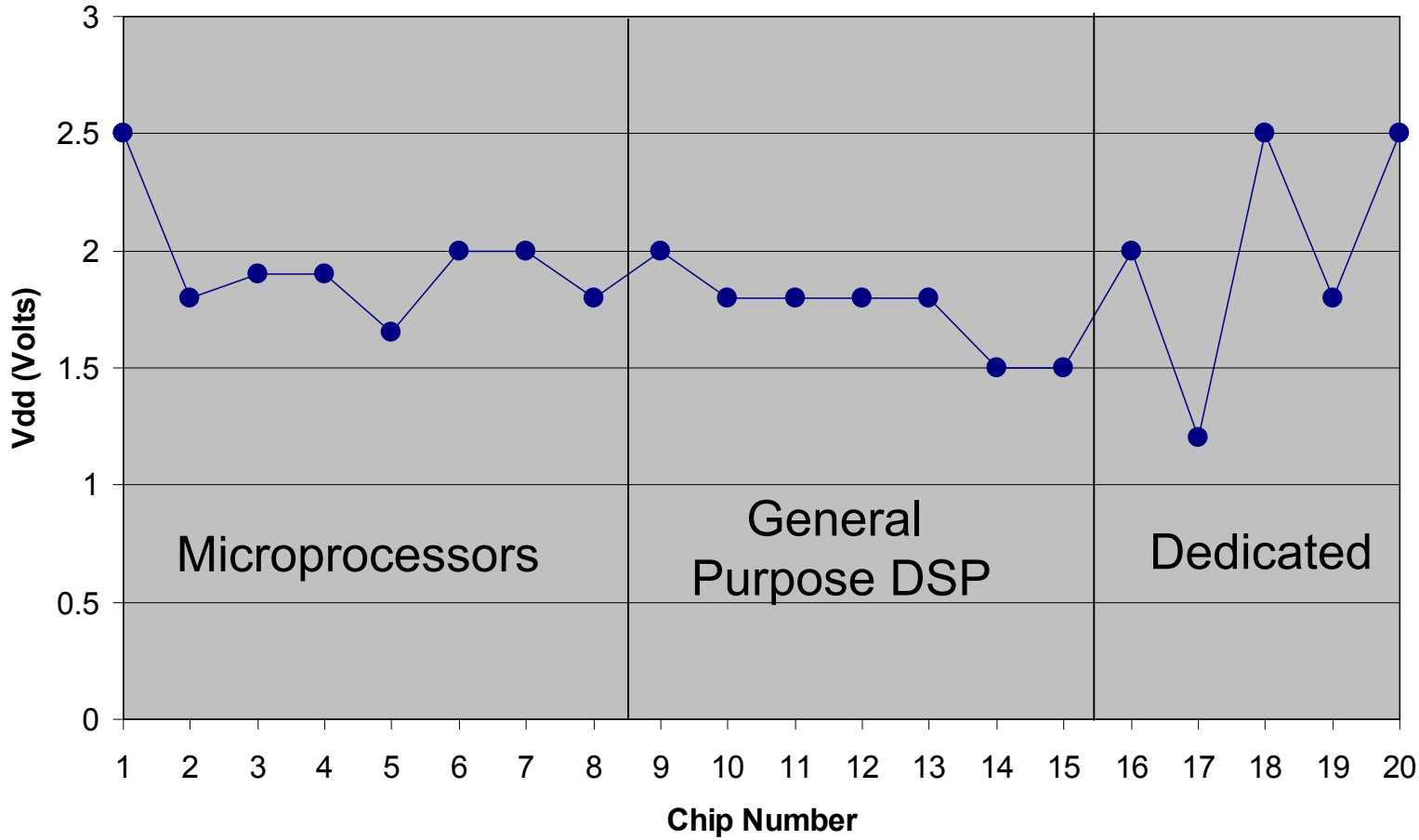
$$= (f_{clk} * N_{op}) / A_{chip} * C_{sw} * (V_{dd})^2 * f_{clk}$$

Simplifying,

$$MOPS/mW = 1/(A_{op} * C_{sw} * V_{dd}^2)$$

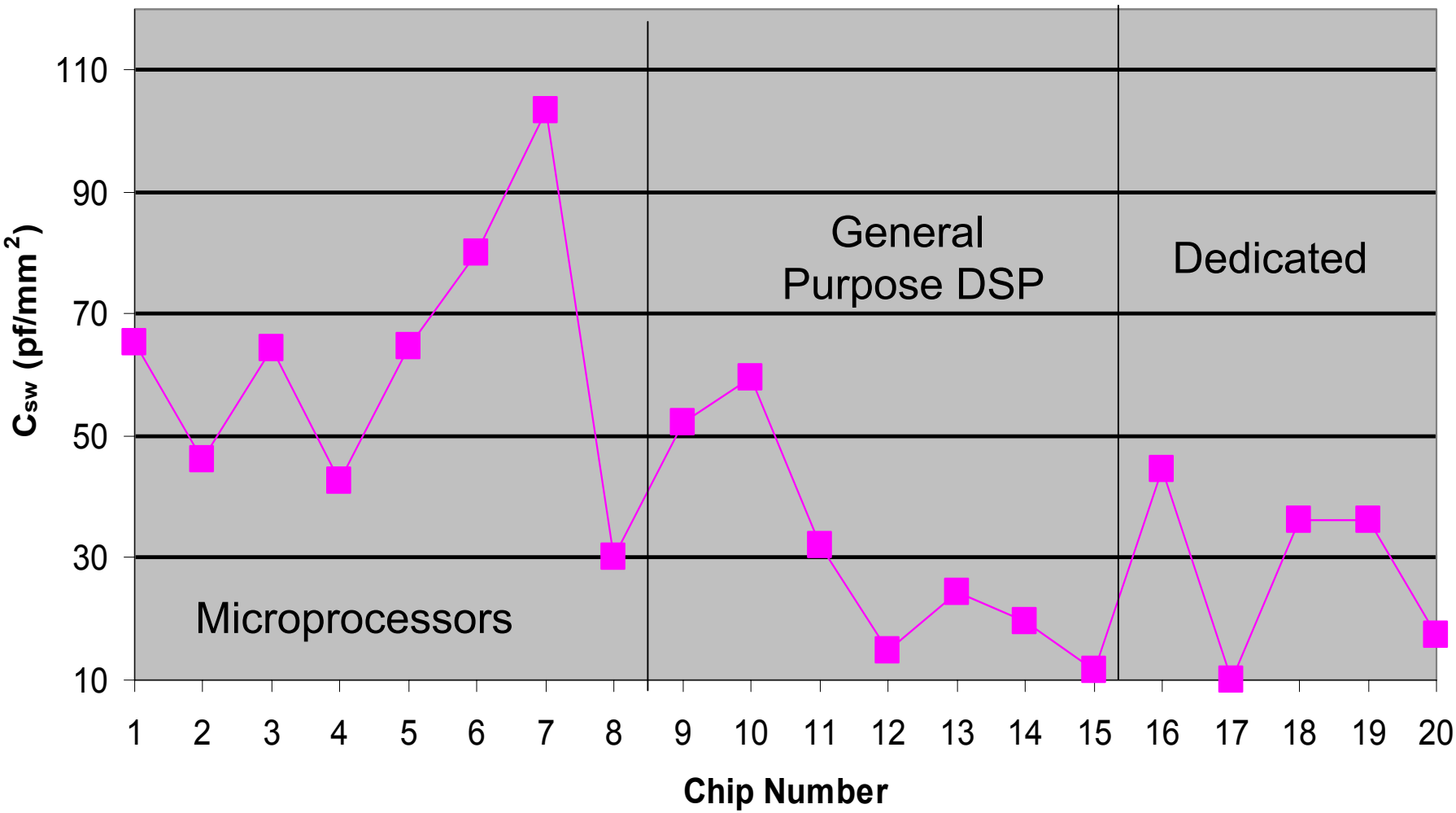
So lets look at the 3 components – V_{dd}, C_{sw} and A_{op}

Supply Voltage, V_{dd}



Supply voltage isn't the cause of the difference, actually a bit higher for the dedicated chips

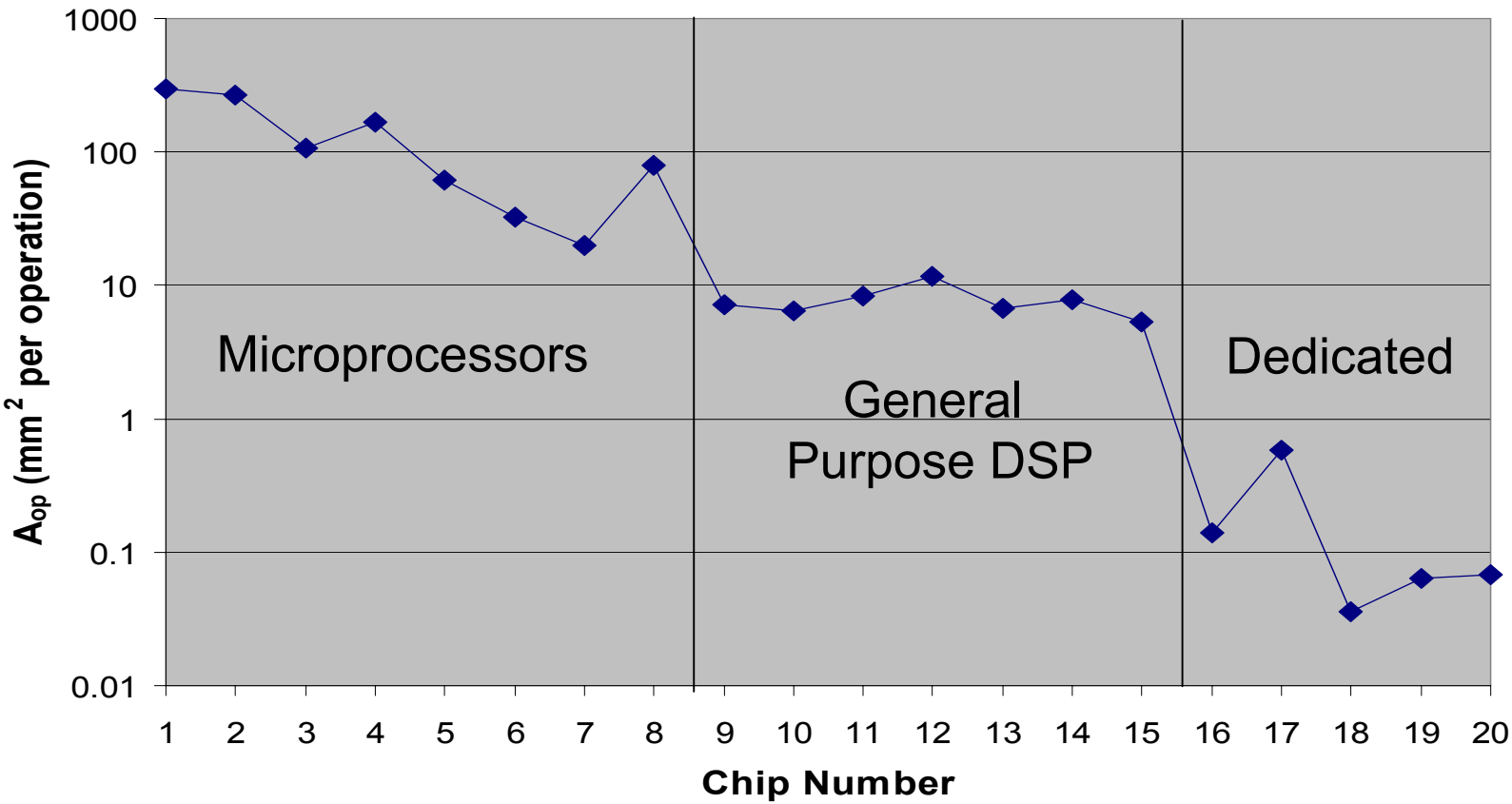
Switched Capacitance, C_{sw} (pF/mm^2)



C_{sw} is lower for dedicated, but only by a factor of 2 to 3

A_{op} = Area per operation (A_{chip}/N_{op})

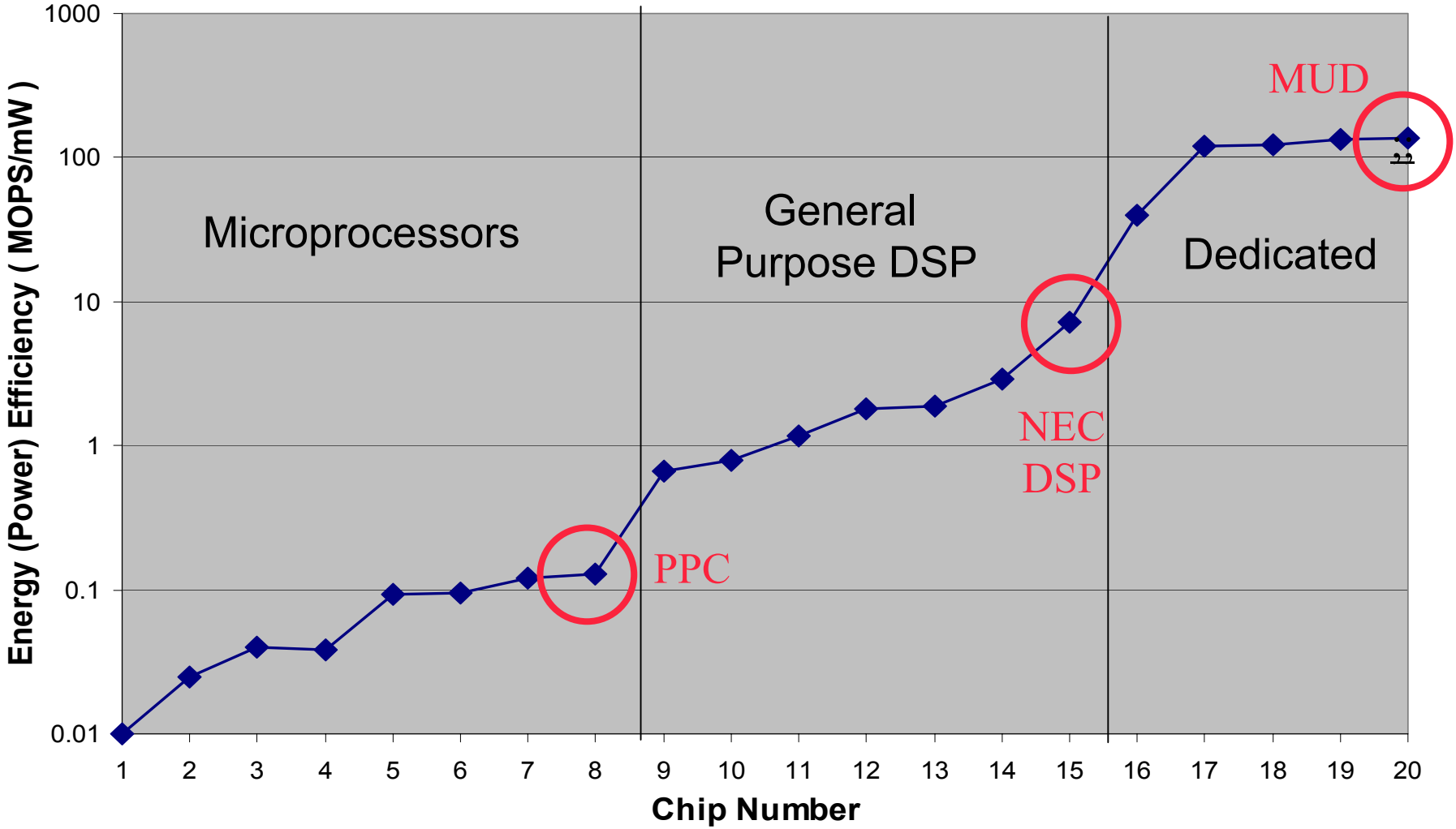
MOPS/mW =$1/(A_{op}*C_{sw}*V_{dd}^2)$; $A_{op} = A_{chip}/N_{op}$



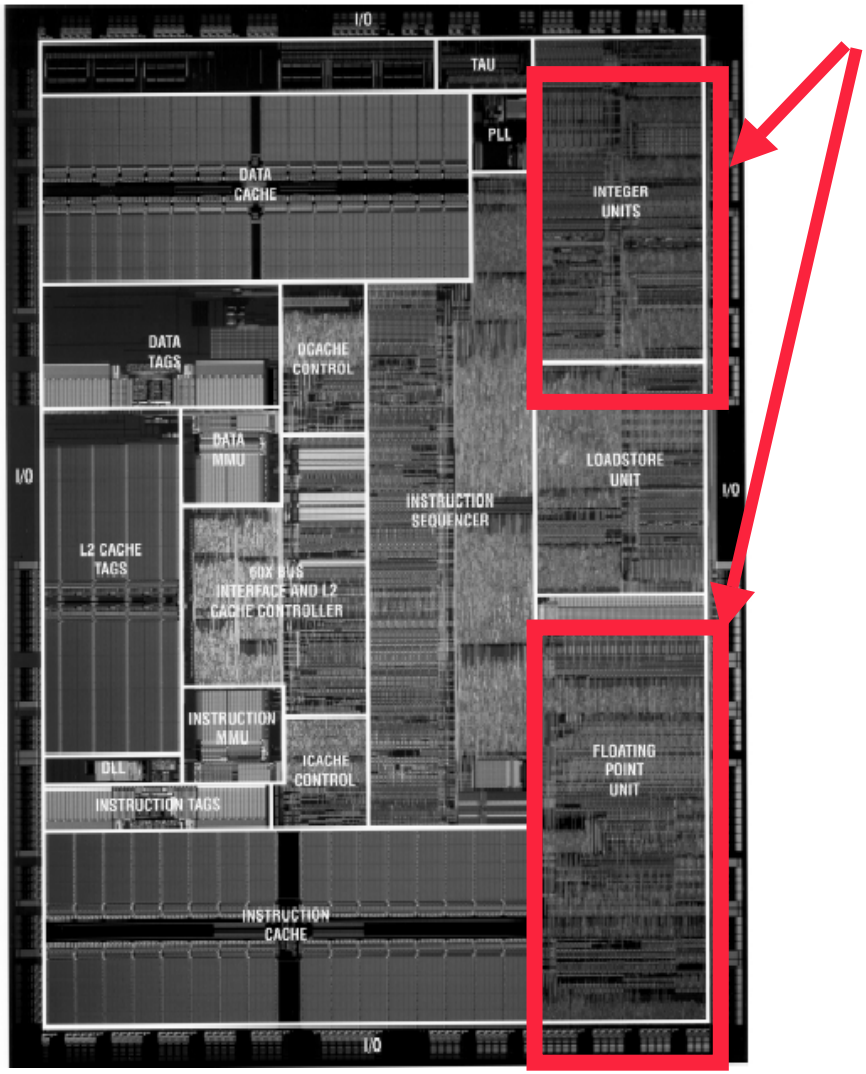
Here is the one that explains the difference, lower due to more parallelism (higher N_{op}) in a smaller chip area (less overhead)

Lets look at some chips to actually see the different architectures

We'll look at one from each category…



Microprocessor: MOPS/mW=.13



The only circuitry which supports "useful operations" All the rest is overhead to support the time multiplexing

$N_{op} = 2$

f_{clock} = 450 MHz (2 way)

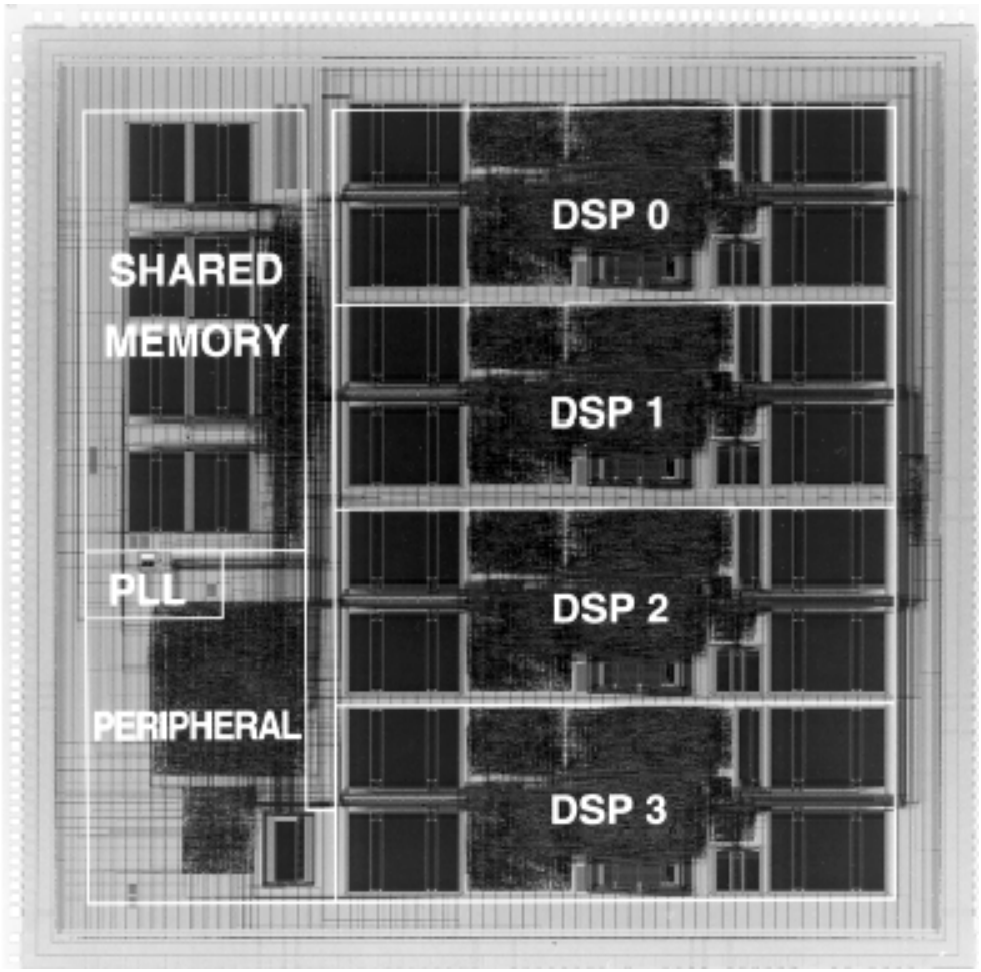
= 900 MIPS

Two operations each clock cycle, so

$A_{op} = A_{chip}/2 =$ 42mm^2

Power = 7 Watts

DSP: MOPS/mW=7



Same granularity (a datapath), more parallelism

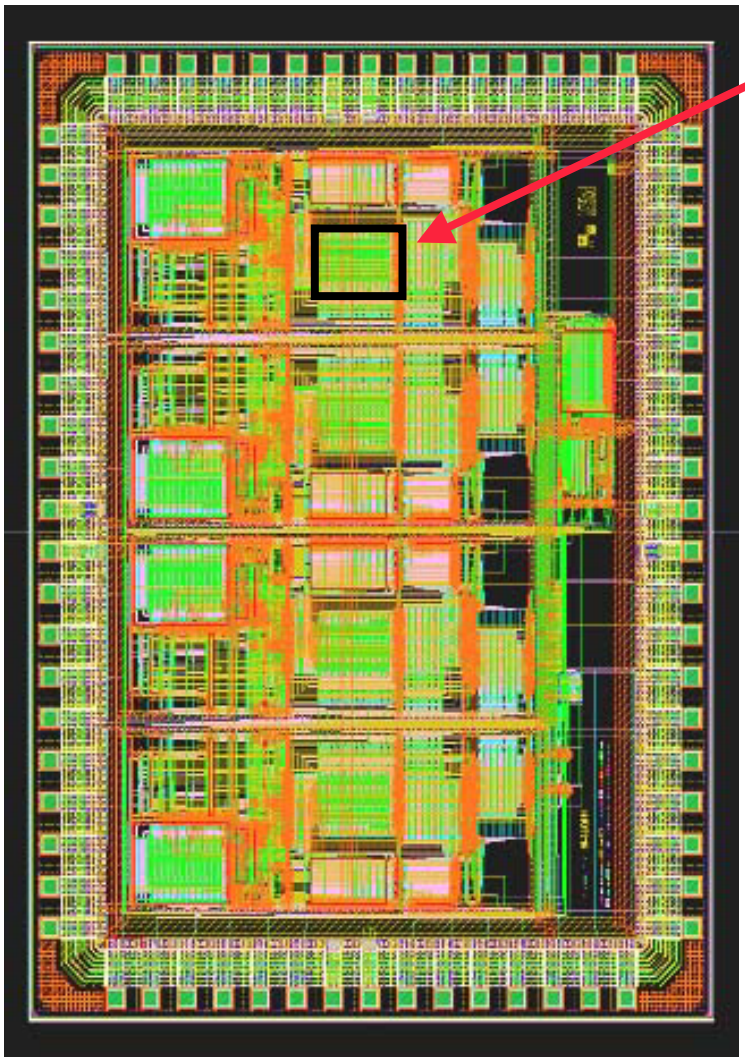
4 Parallel processors (4 ops each)
$N_{op} = 16$

50 MHz clock
=> 800 MOPS

Sixteen operations each clock cycle, so
$A_{op} = A_{chip}/16 = 5.3mm^2$

Power = 110 mW.

Dedicated Design: MOPS/mW=200



Fully parallel mapping of adaptive correlator algorithm. No time multiplexing.

N_{op} = 96

Clock rate = 25 MHz => 2400 MOPS

A_{op} = 5.4 mm^2/96 =.15 mm^2

Power = 12 mW

The Basic Problem is Time Multiplexing

- Processor architectures obtain performance by increasing the clock rate, because the parallelism is low
- Results in ever increasing memory on the chip, high control overhead and fast area consuming logic

But doesn't time multiplexing give better area efficiency???

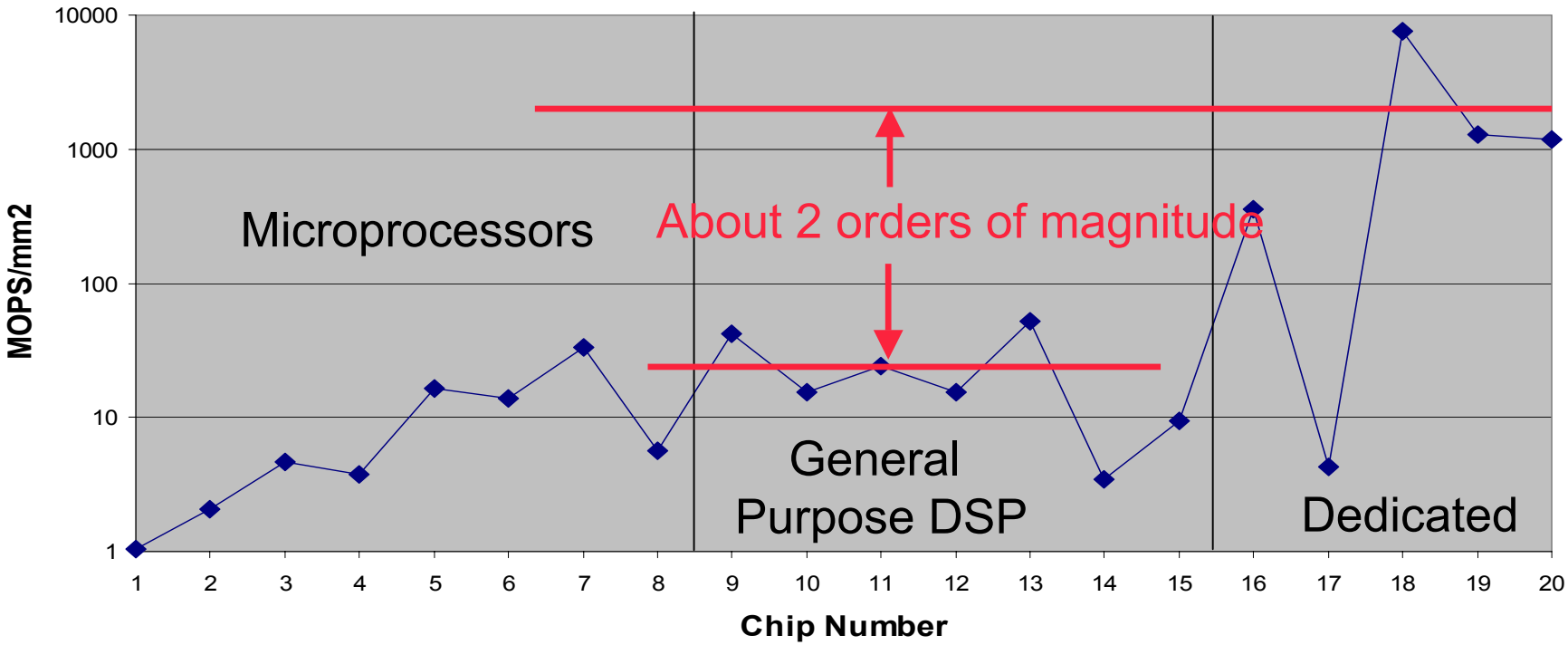
Area Efficiency

- SOC based devices are often very cost sensitive
- So we need a $ cost metric => for SOC's it is equivalent to the efficiency of area utilization
- Area Efficiency Metric:

  Computation per unit area = MOPS/mm^2

How much of a $ cost (area) penalty will we have if we put down many parallel hardware units and have limited time multiplexing?

Surprisingly the area efficiency roughly tracks the energy efficiency…



The overhead of flexibility in processor architectures is so high that there is even an area penalty

Hardware/software

Conclusion:

There is no software/hardware tradeoff.

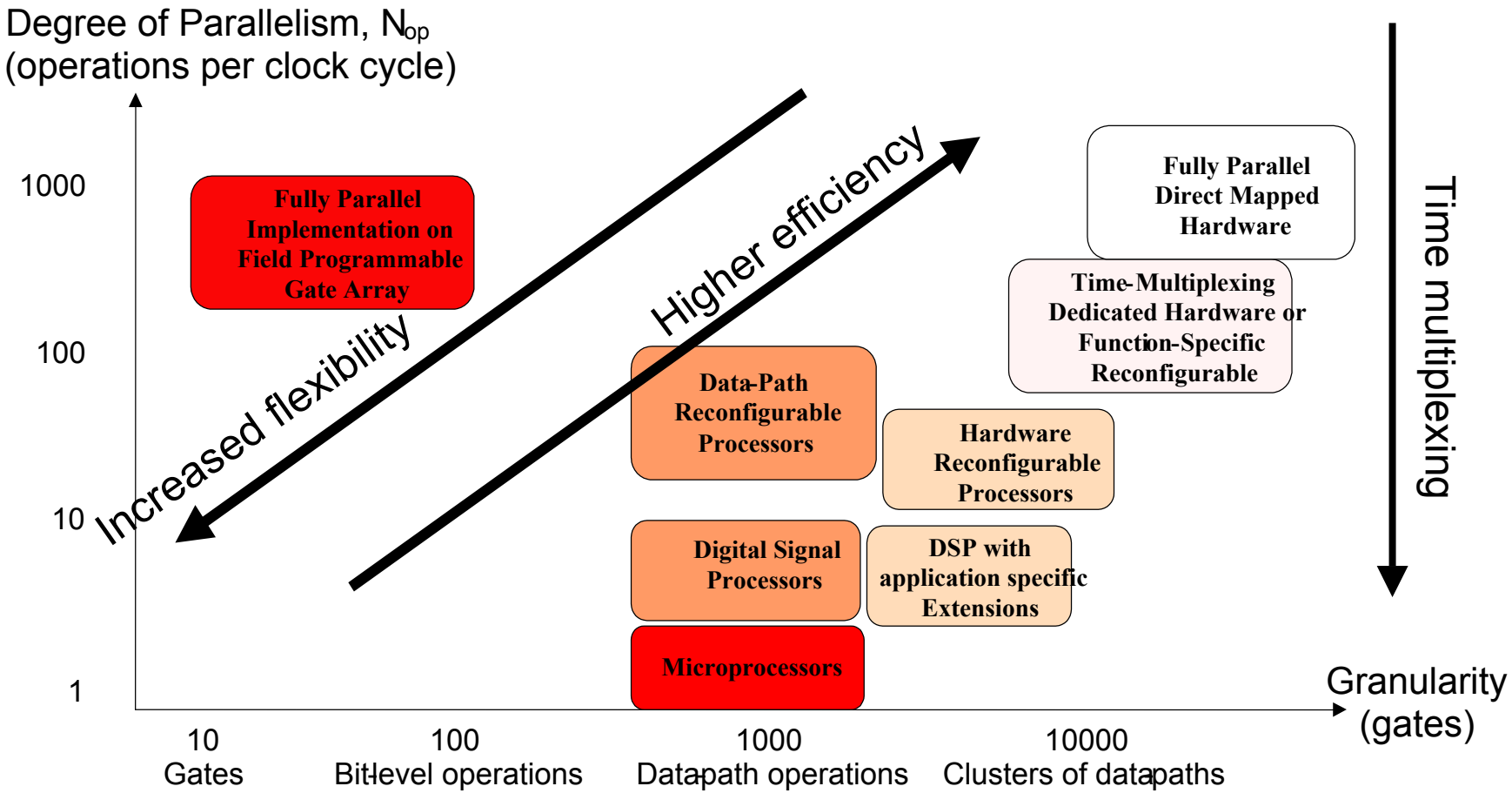
- The difference between hardware and software in performance, power and area is so large that there is no “tradeoff”.
- It is reasons other than power, energy, performance or cost that drives a software solution (e.g. business, legacy, …).
- The **“Cost of Flexibility” is extremely high**, so the other reasons better be good!

Are there better ways to provide flexibility?

- Lets say the reasons for flexibility are good enough, then are there alternatives to processor based software programmability??

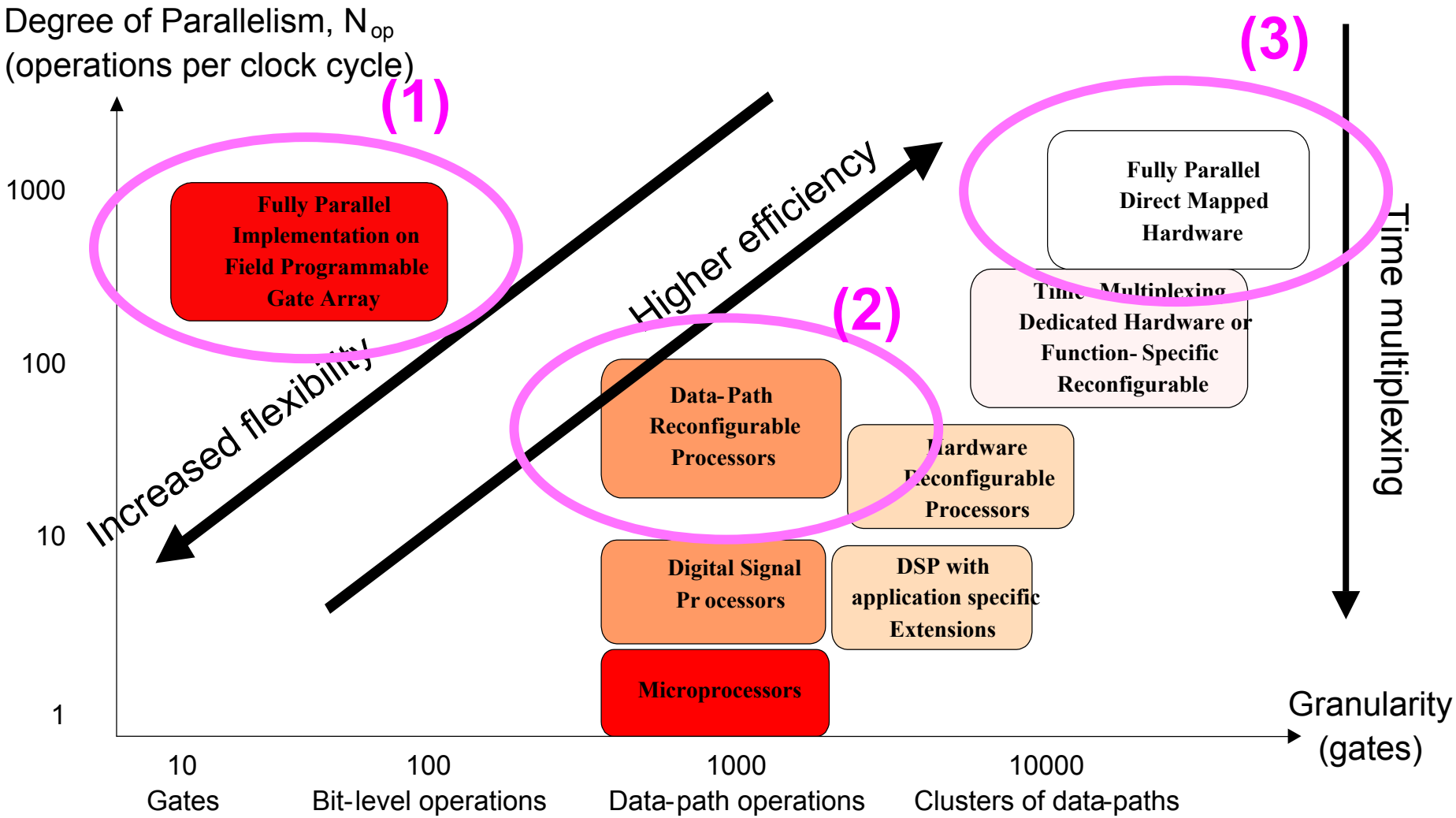
- Yes…
  - » The key is to provide flexibility along with the parallelism we get from the technology..
  - » Lots of choices…

Granularity and Parallelism

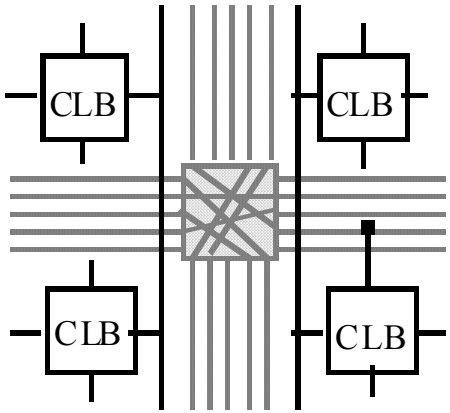


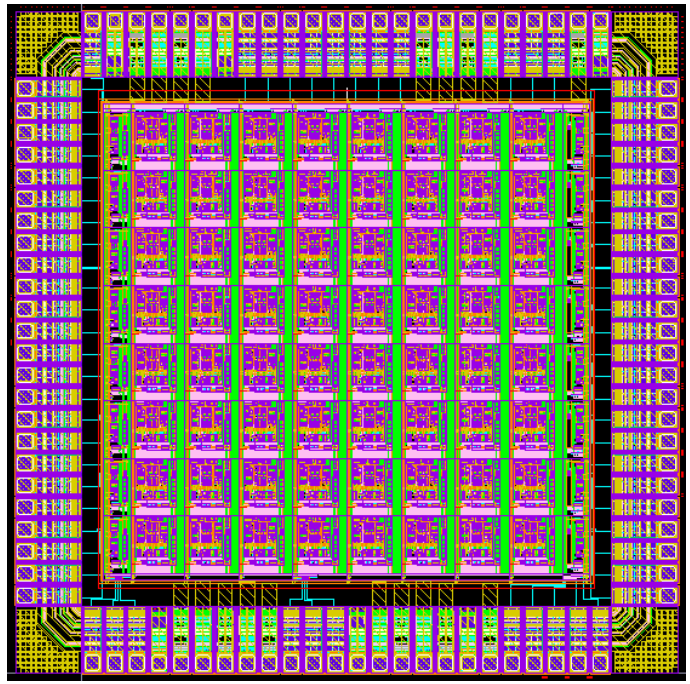
- Increased granularity and higher parallelism yields higher efficiency
- Smaller granularity and reduced parallelism yields more flexibility
- Time multiplexing is needed for performance with low parallelism

We will look at three cases…



Case (1): Reconfigurable Logic: FPGA

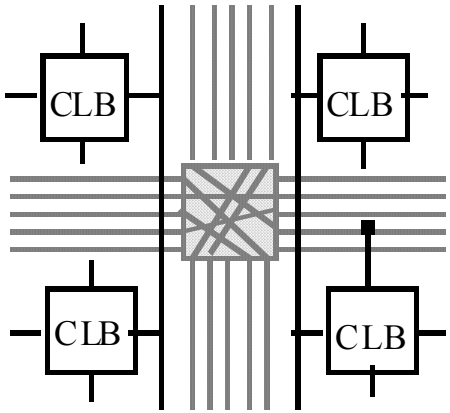


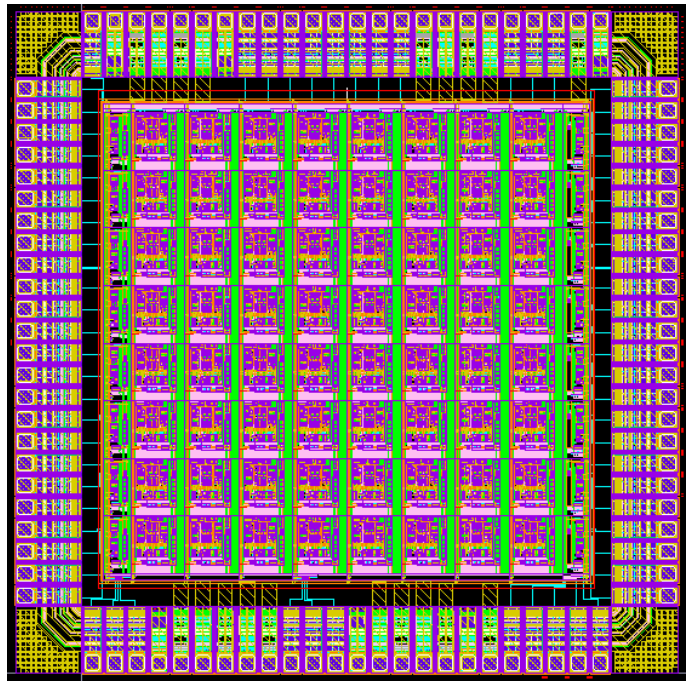


- Very low granularity (CLB's) – *improves* flexibility
- High parallelism – *improves* efficiency

But….

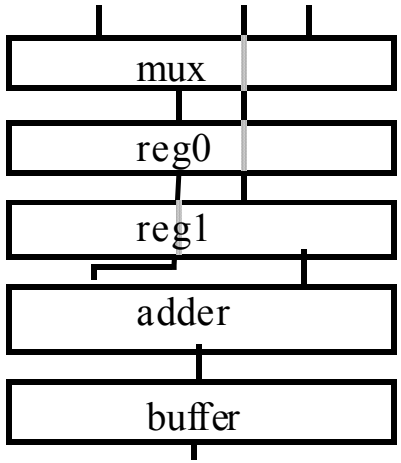
Case (1): Reconfigurable Logic: FPGA

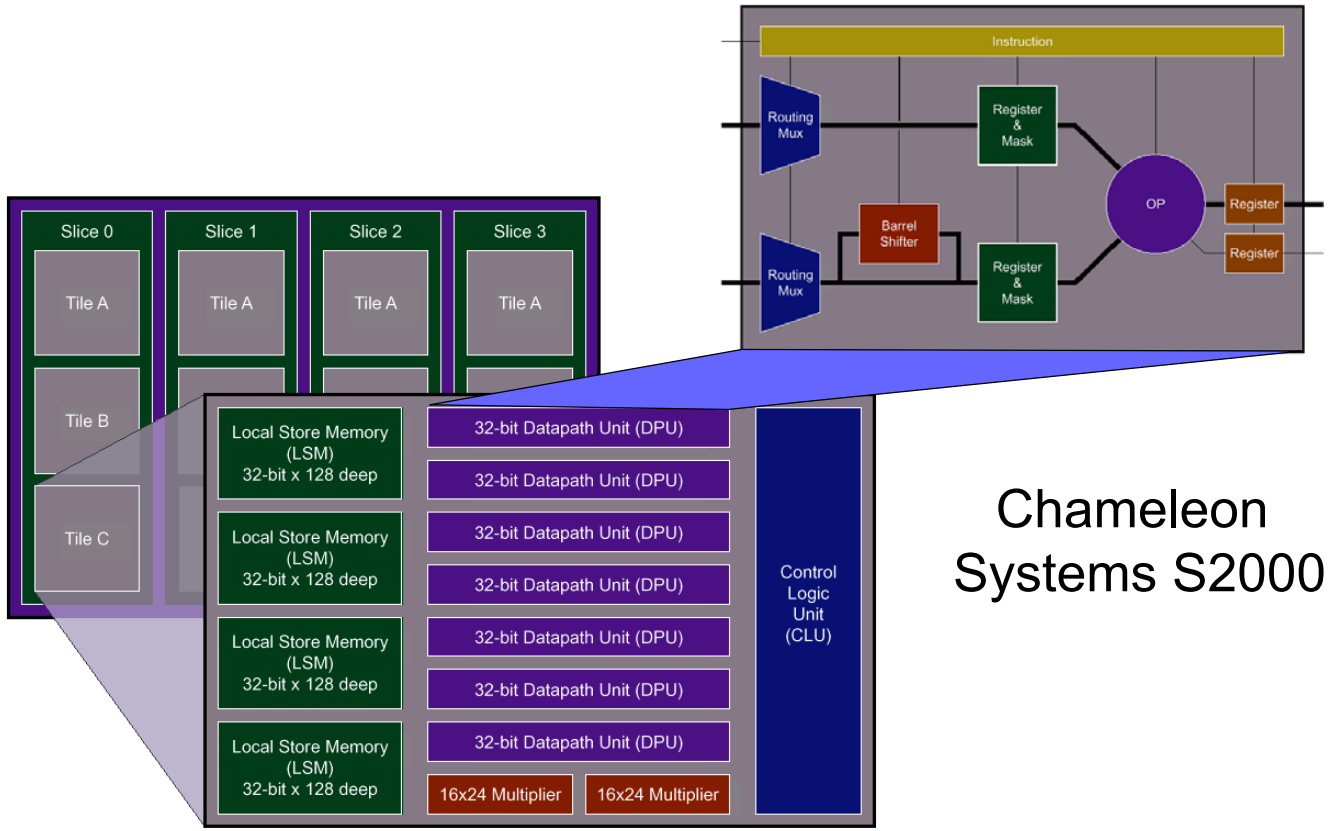




- Very low granularity (high amount of interconnect) – decreases efficiency

Case (2): Reconfiguration at a higher level of granularity



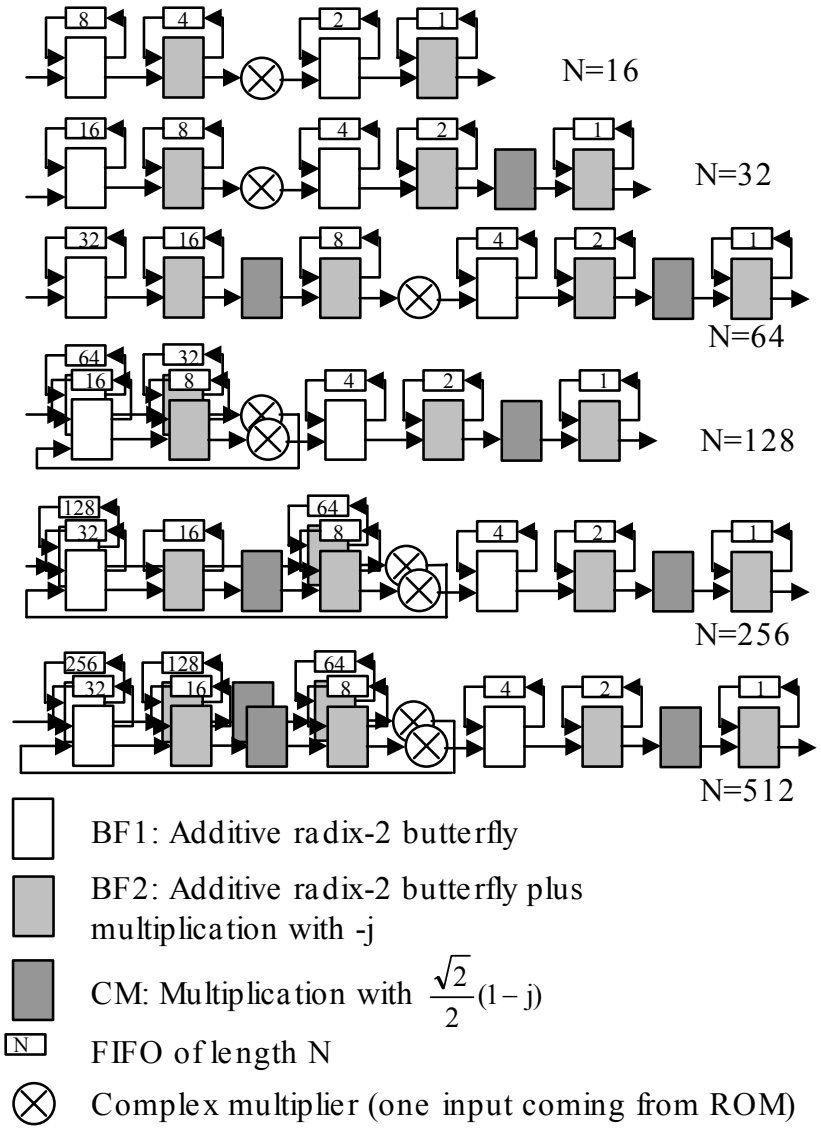


Chameleon Systems S2000

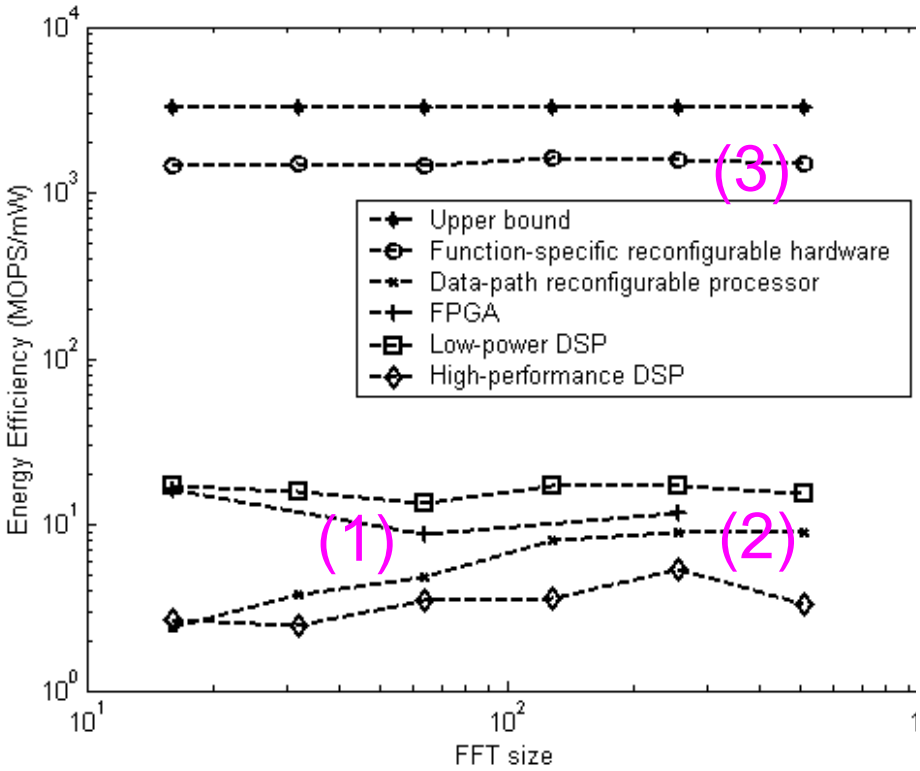
- Higher granularity – datapath units
- Higher efficiency, but lower flexibility

Case (3): Even higher granularity - "Flexible" dedicated hardware

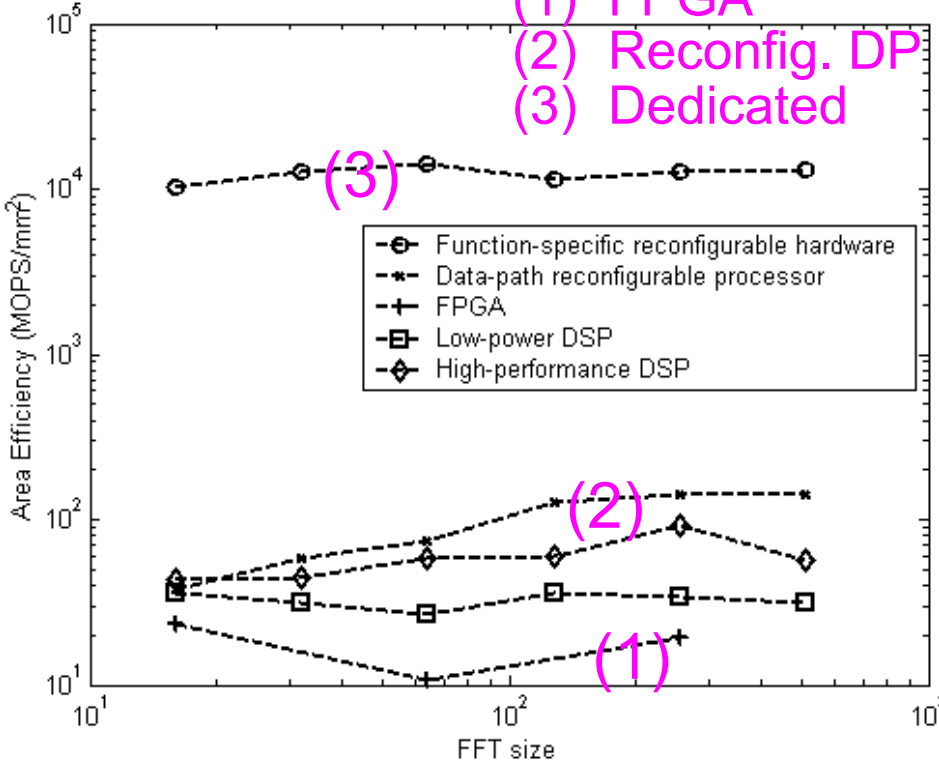
- Use a hardware architecture that has the flexibility to cover a range of parameter values
- Not much flexibility, but very high efficiency
- Example here is an FFT which can range from N=16 to 512
- Uses time multiplexing



Efficiencies for a variety of architectures for a flexible FFT



MOPS/mW
vs. FFT size



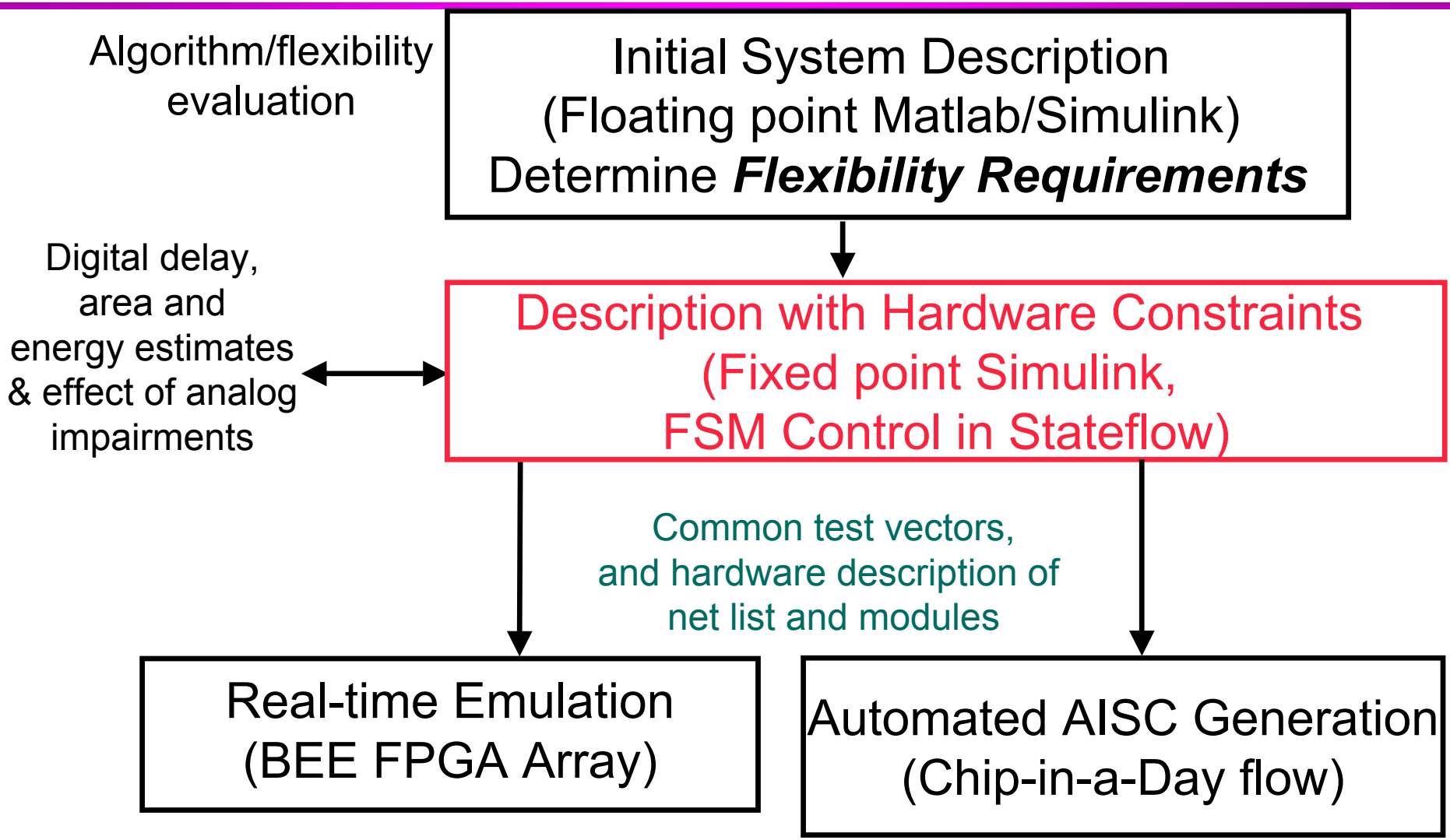
MOPS per mm^2
vs. FFT size

* All results are scaled to 0.18μm

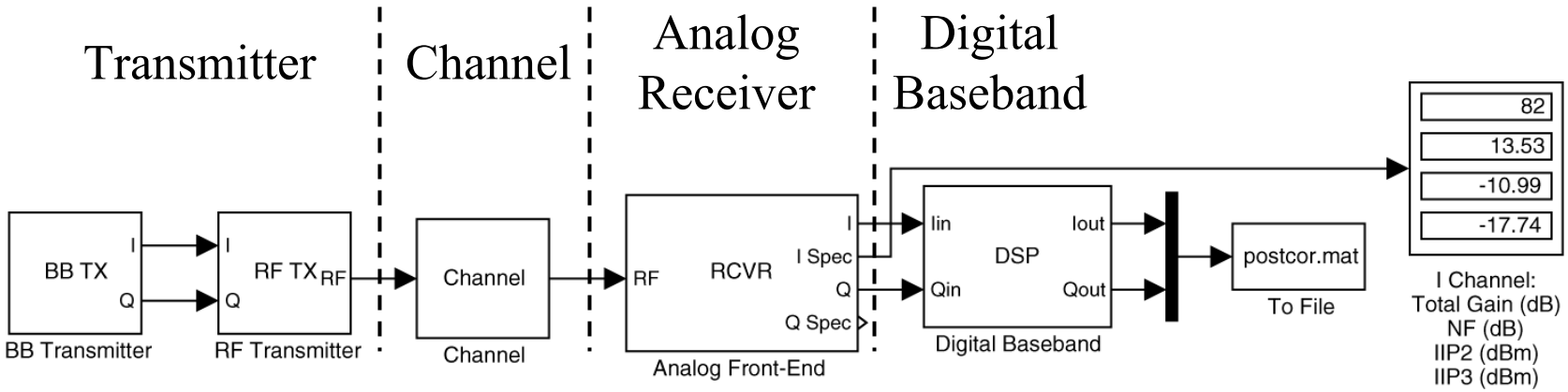
The Issues

- How much flexibility is needed and how best to include it…
- A single system description including interaction between the analog and digital domains
- “Realtime” SOC prototyping
- Automated ASIC design flow

An SOC Design Flow with Prototyping



Simulation Framework using Simulink/Stateflow (from Mathworks, Inc.)

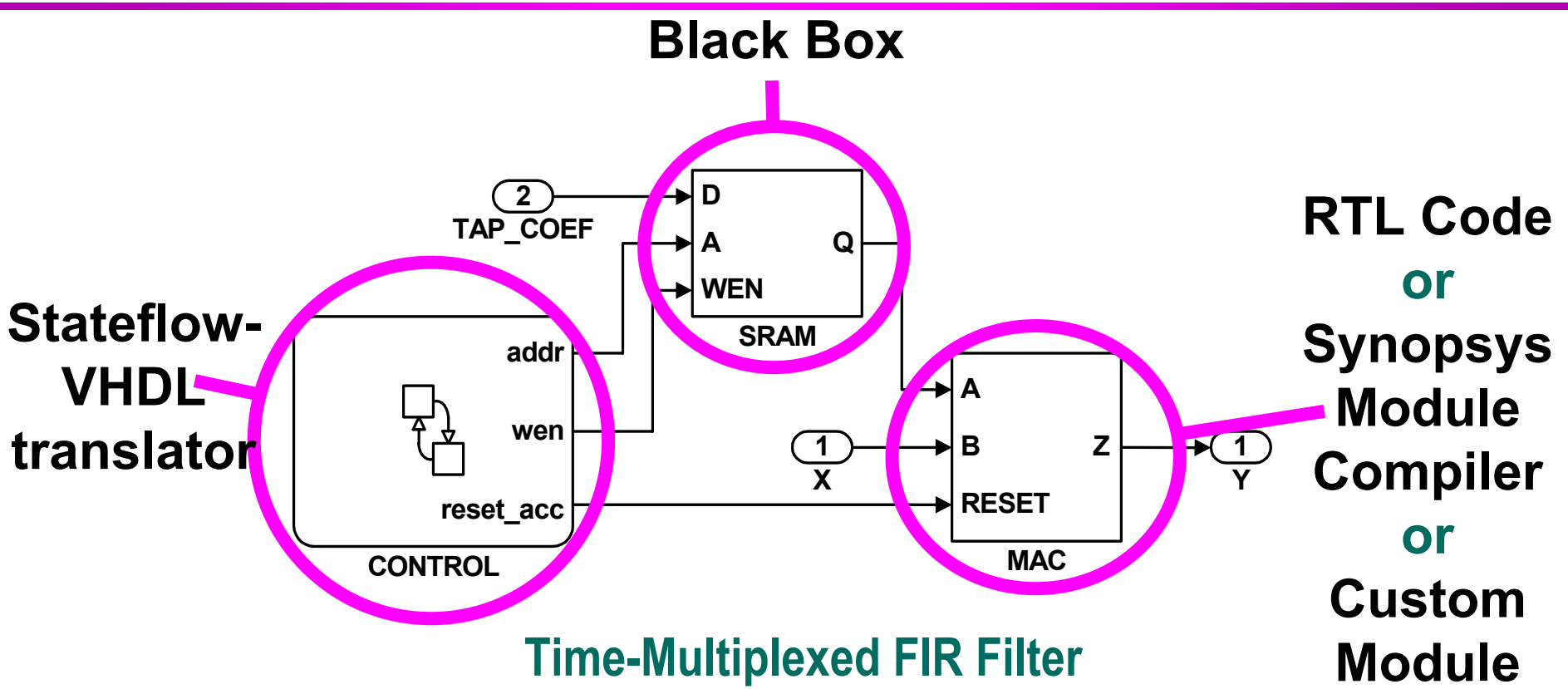


- **Techniques used to decrease simulation time:**

  Baseband-equivalent modeling of RF blocks

  Compile design using MATLAB Real-Time Workshop

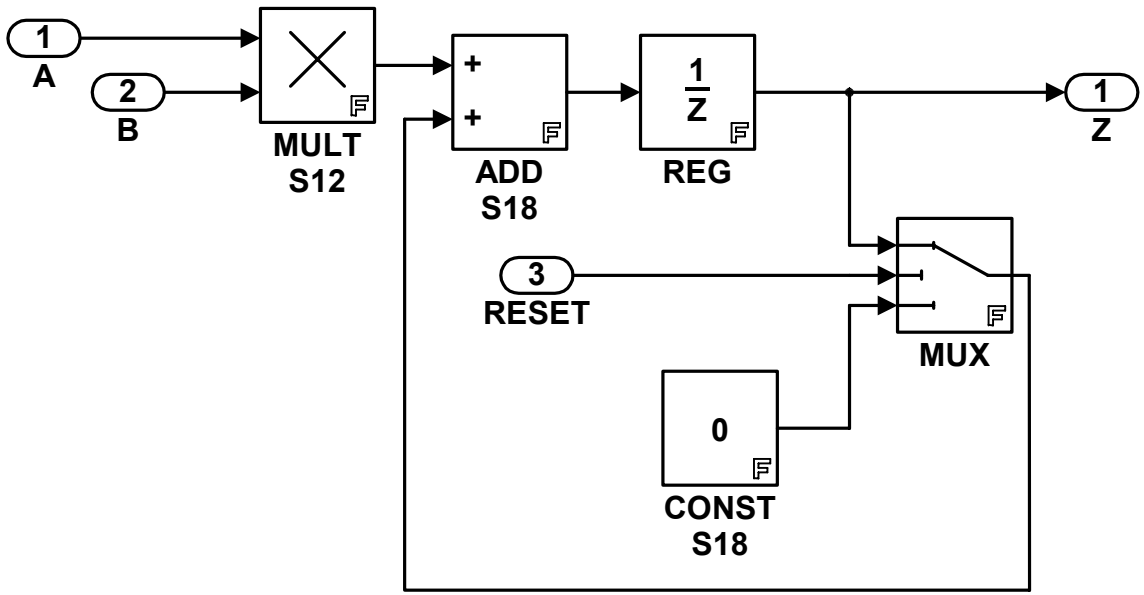
Blocks map to implementation libraries



- Implementation choices embedded in description
- Libraries of blocks are pre-verified and re-used

Timed Dataflow Graph Specification

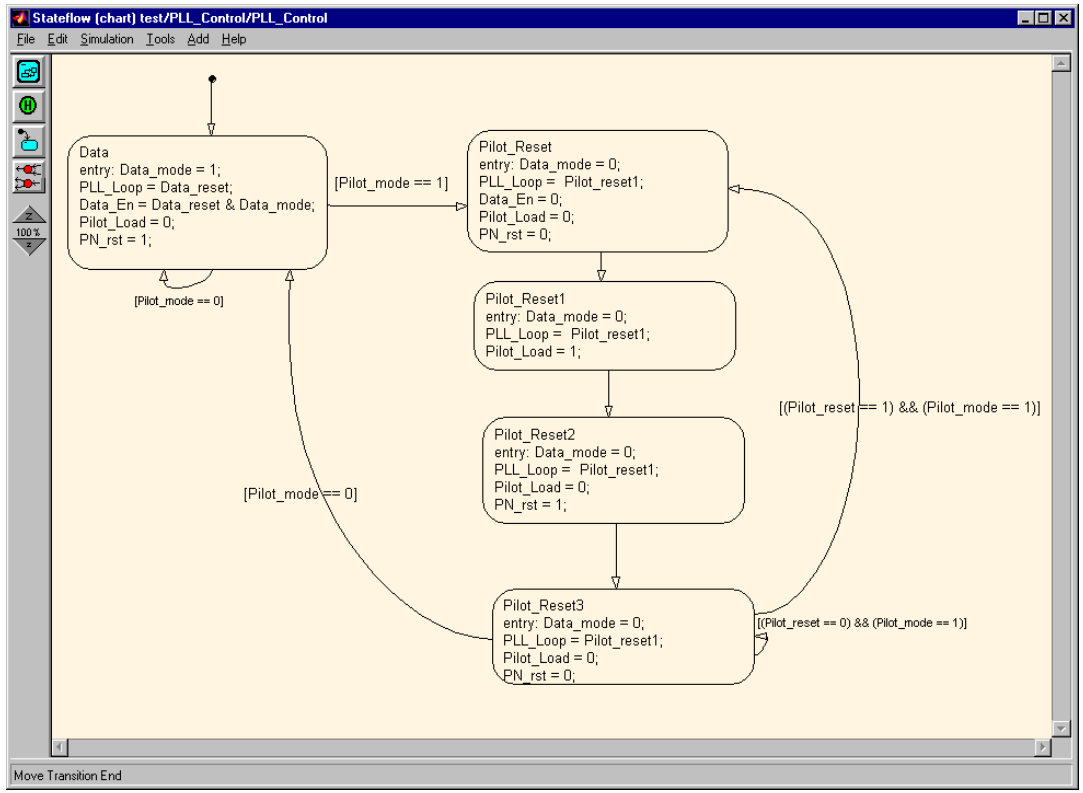
- Simulink (from Mathworks)
- Discrete-Time (cycle accurate)
- Fixed-Point Types (bit true)
- No need for RTL simulation
- Embedded implementation choices



Multiply / Accumulate

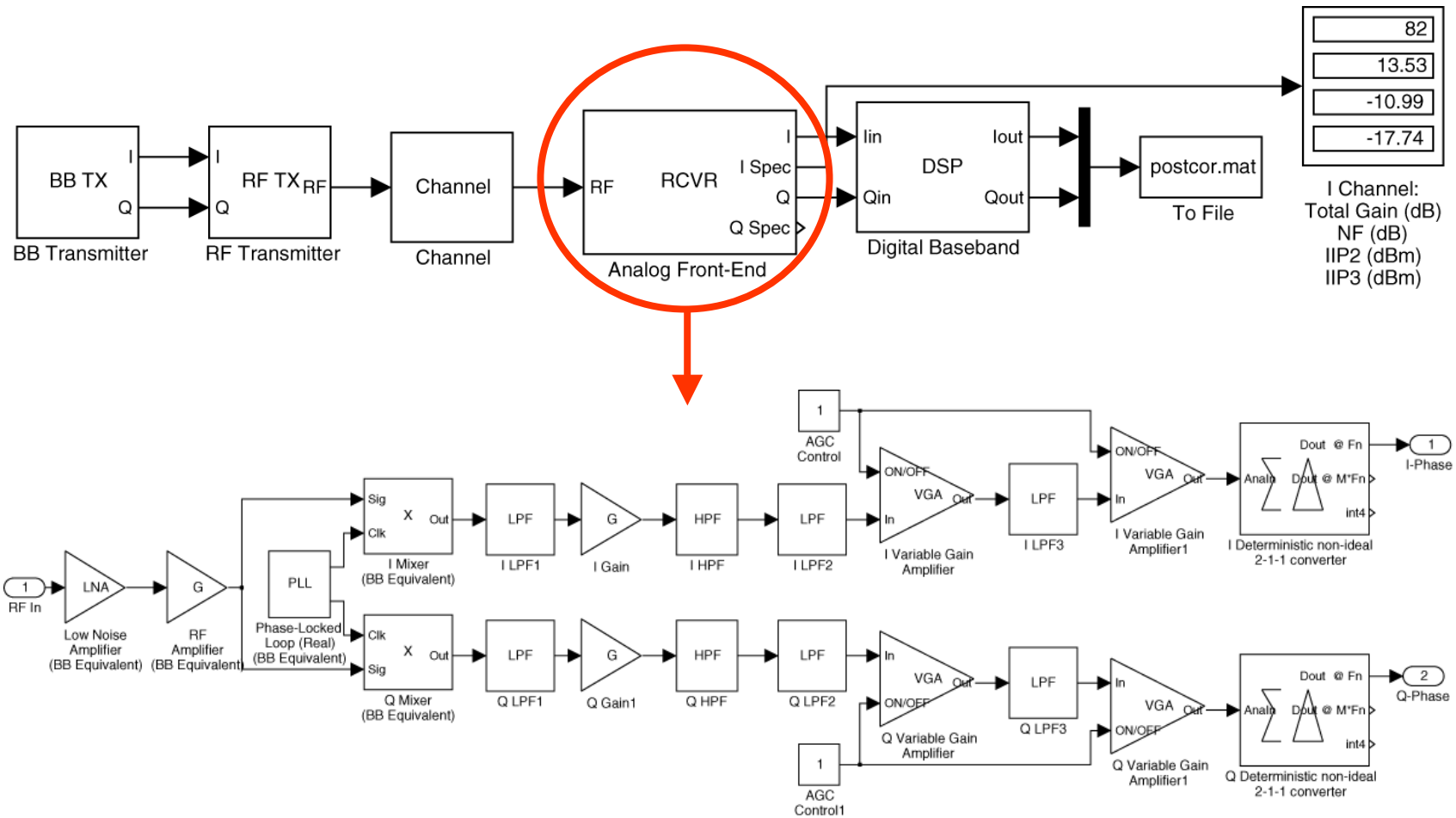
Control

- Stateflow
  - » Extended Finite State Machine
  - » Subset of Syntax
  - » Converted to VHDL
  - » Synthesized
- VHDL
  - » Synthesized directly

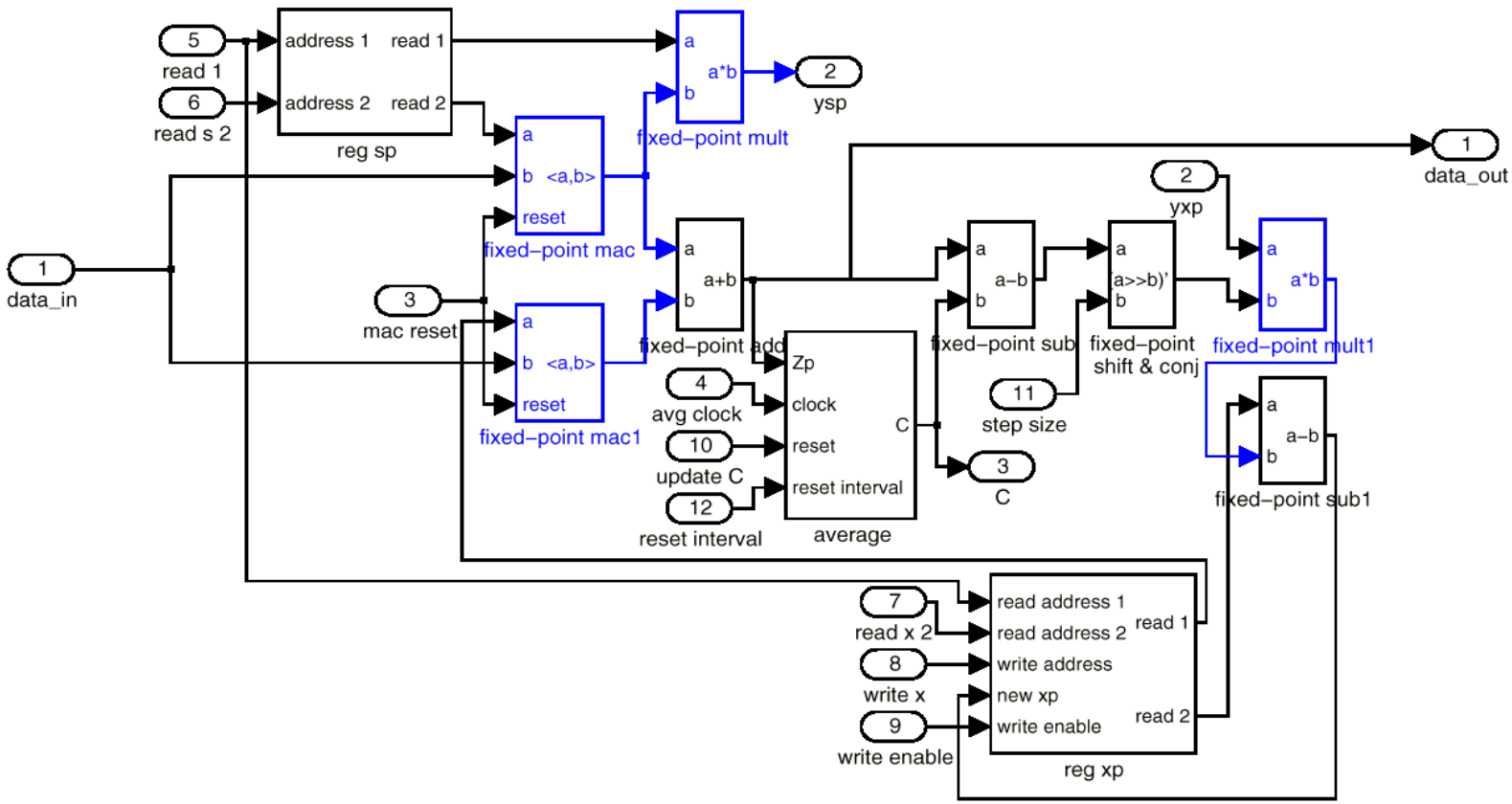


VHDL & Stateflow Macros map to a netlist of Standard Cells using standard synthesis

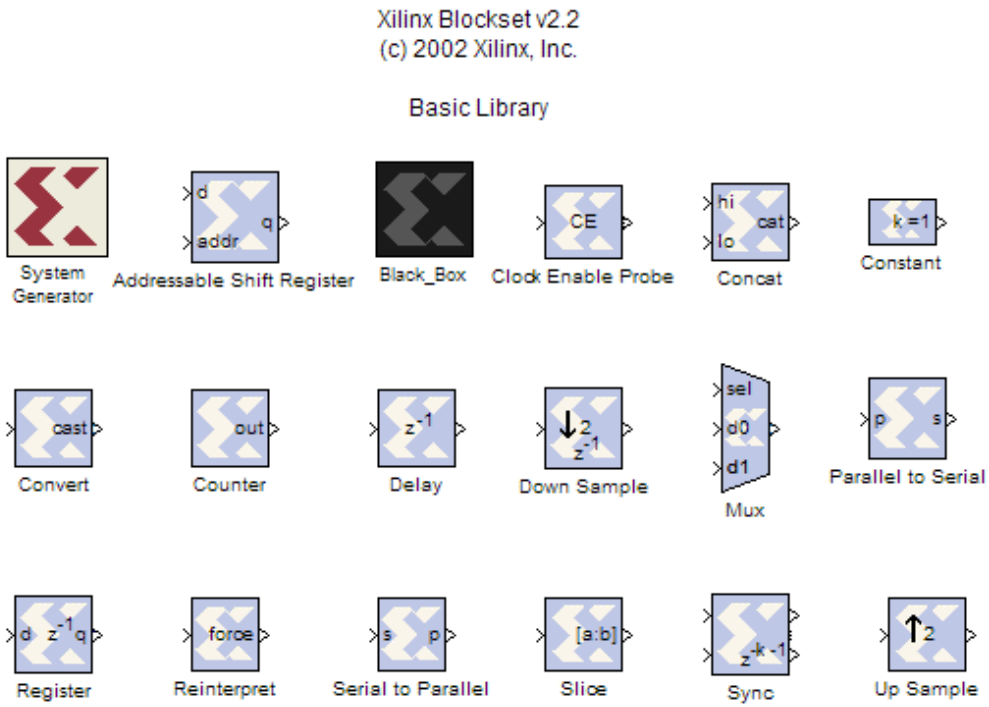
Simulink Model of Direct-Conversion Receiver

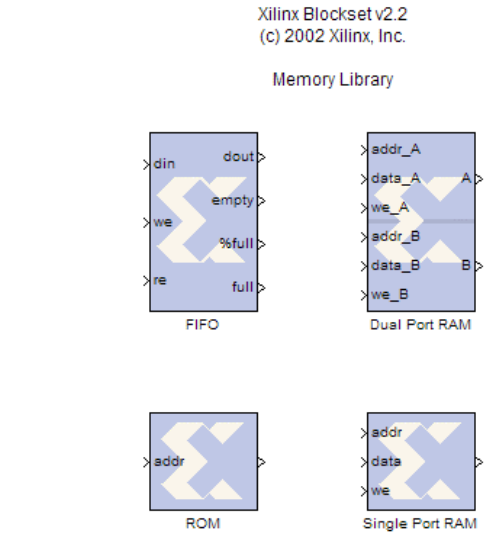


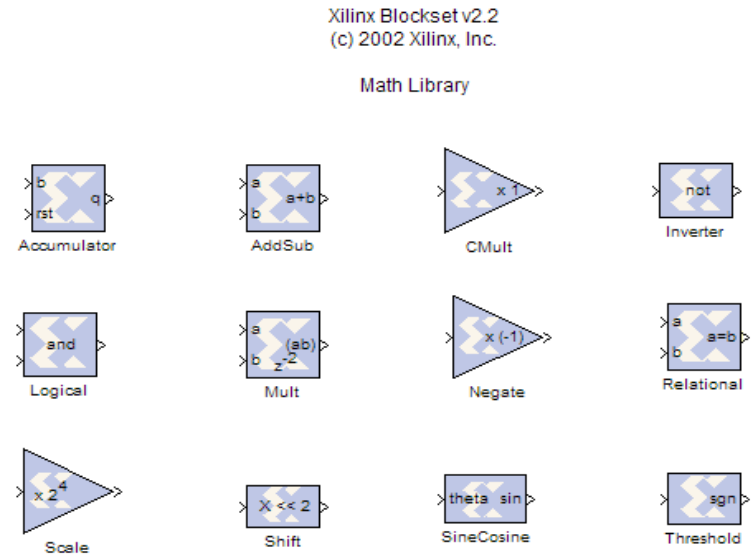
Bit true, cycle accurate digital baseband algorithms…



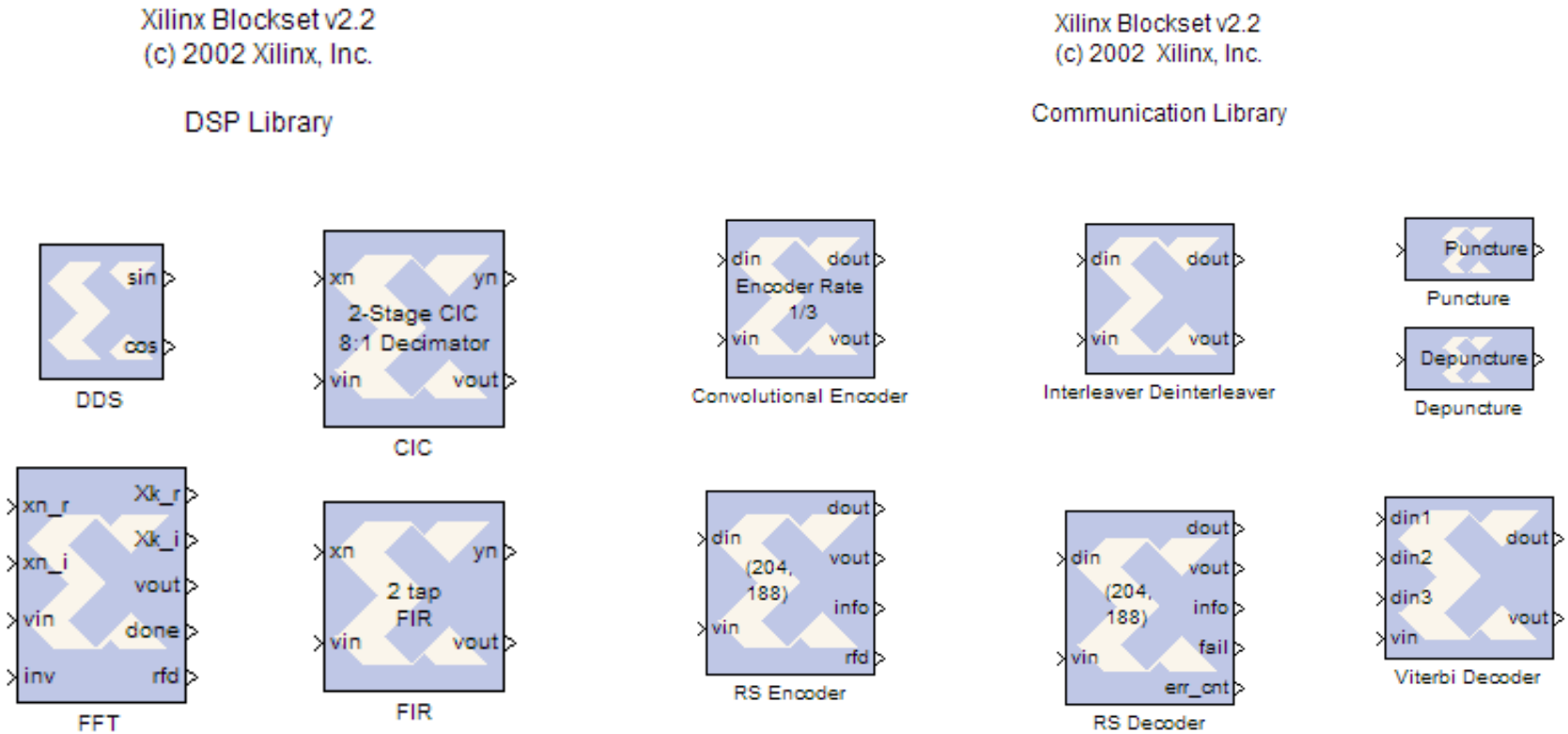
Basic Blocks based on Xilinx System Generator libraries



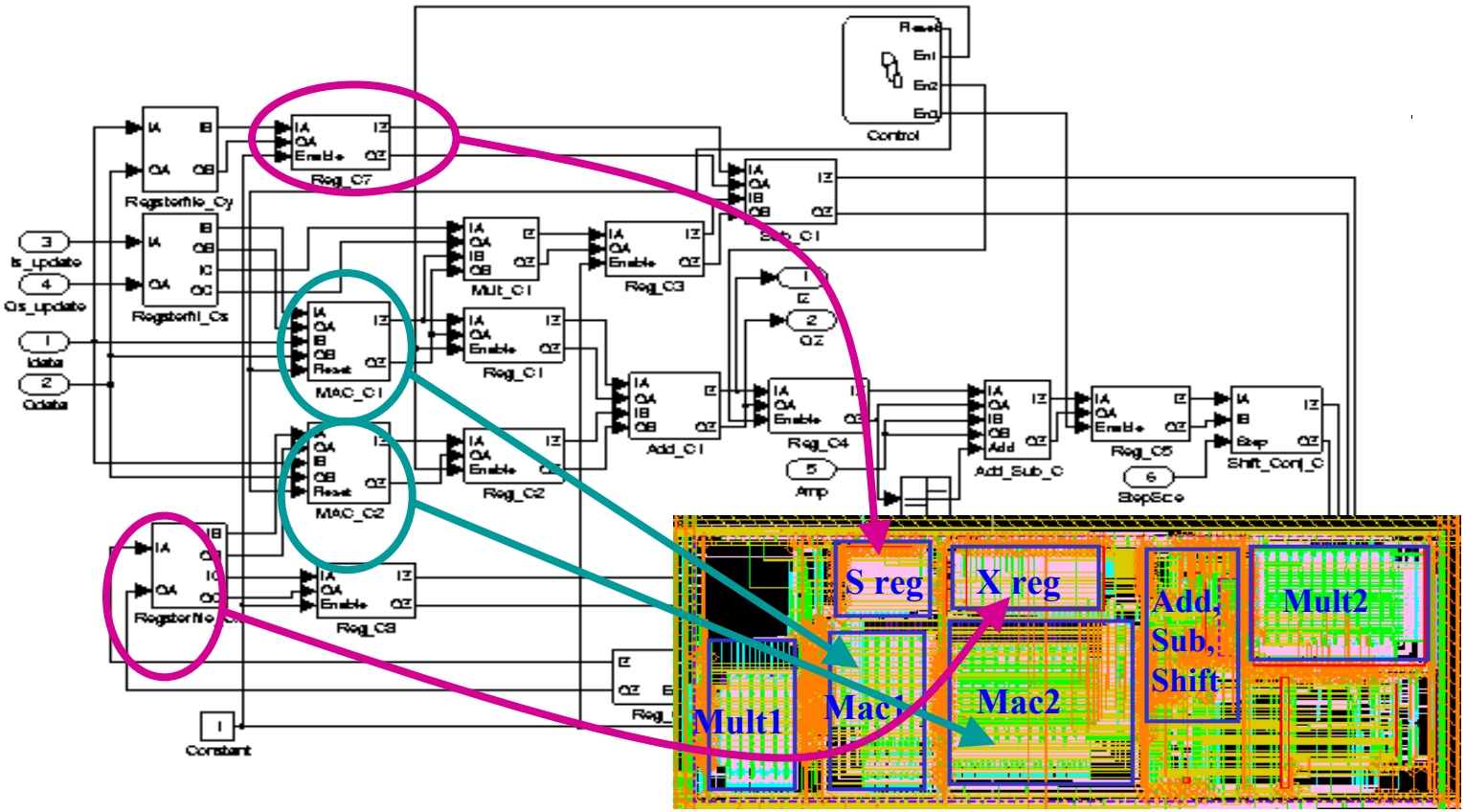




Higher level DSP Blocks



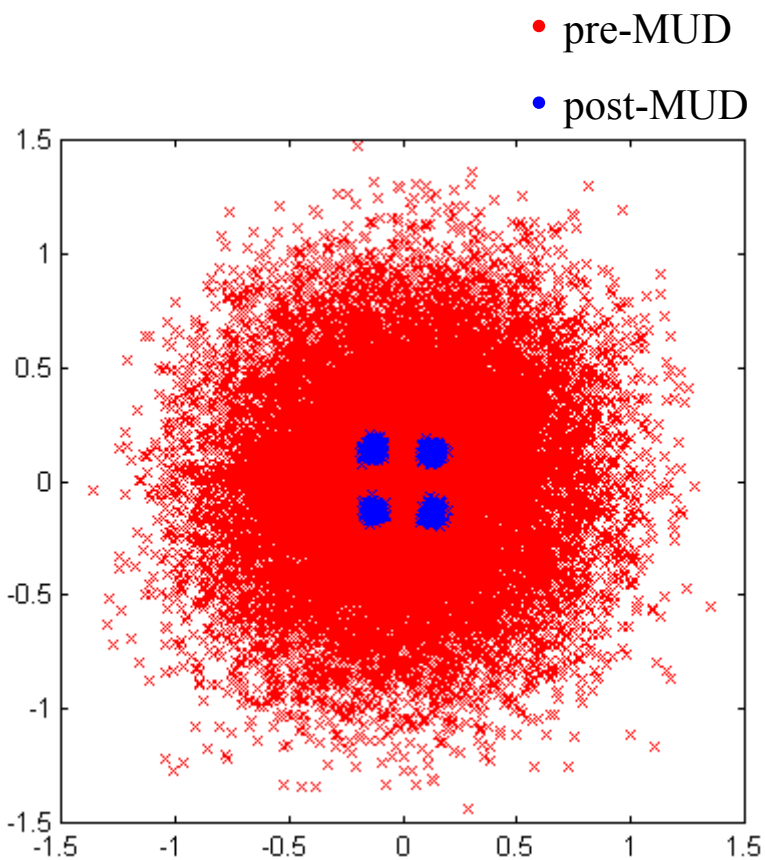
Directly map diagram into hardware since there is a one for one relationship for each of the blocks



- Results: A fully parallel architecture that can be implemented rapidly

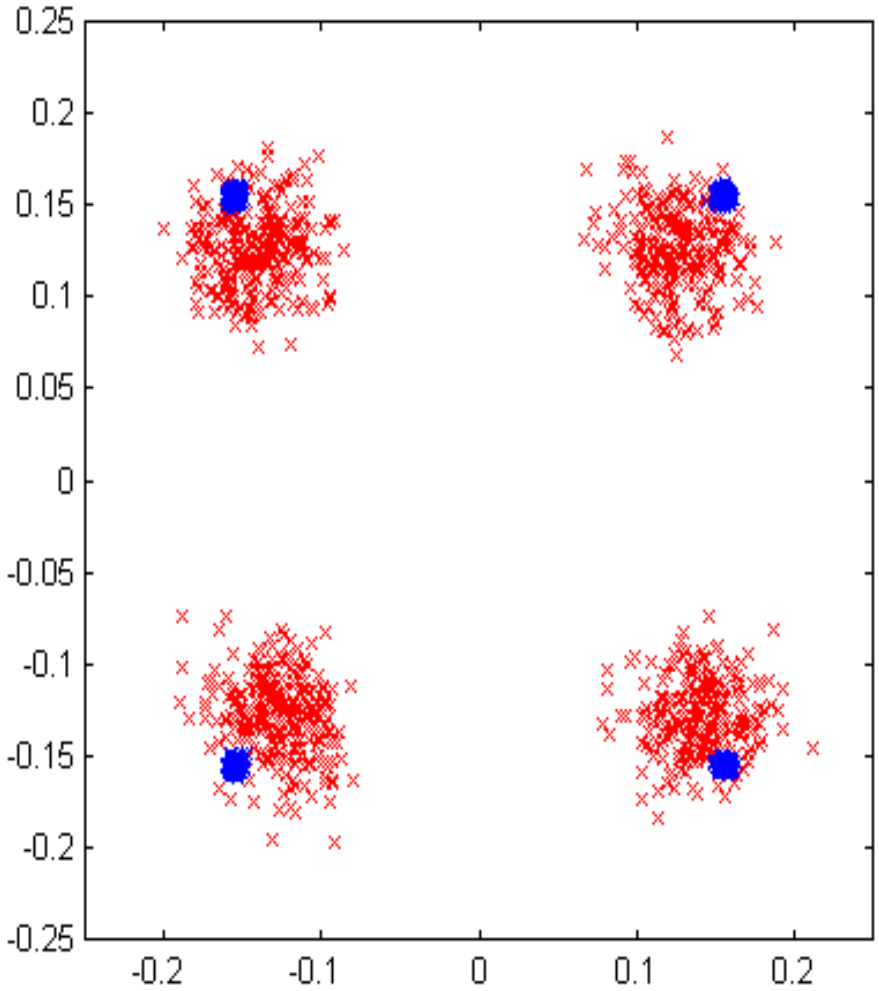
Then do a simulation: Zero-IF Receiver

- 10 users (equal power)
- 13.5dB receiver NF
- PLL: -80dBc/Hz @ 100kHz
- 2.5° I/Q phase mismatch
- 82dB gain
- 4% gain mismatch
- IIP2 = -11dBm
- IIP3 = -18dBm
- 500kHz DC notch filter
- 20MHz Butterworth LPF
- 10-bit, 200MHz Σ-Δ ADC



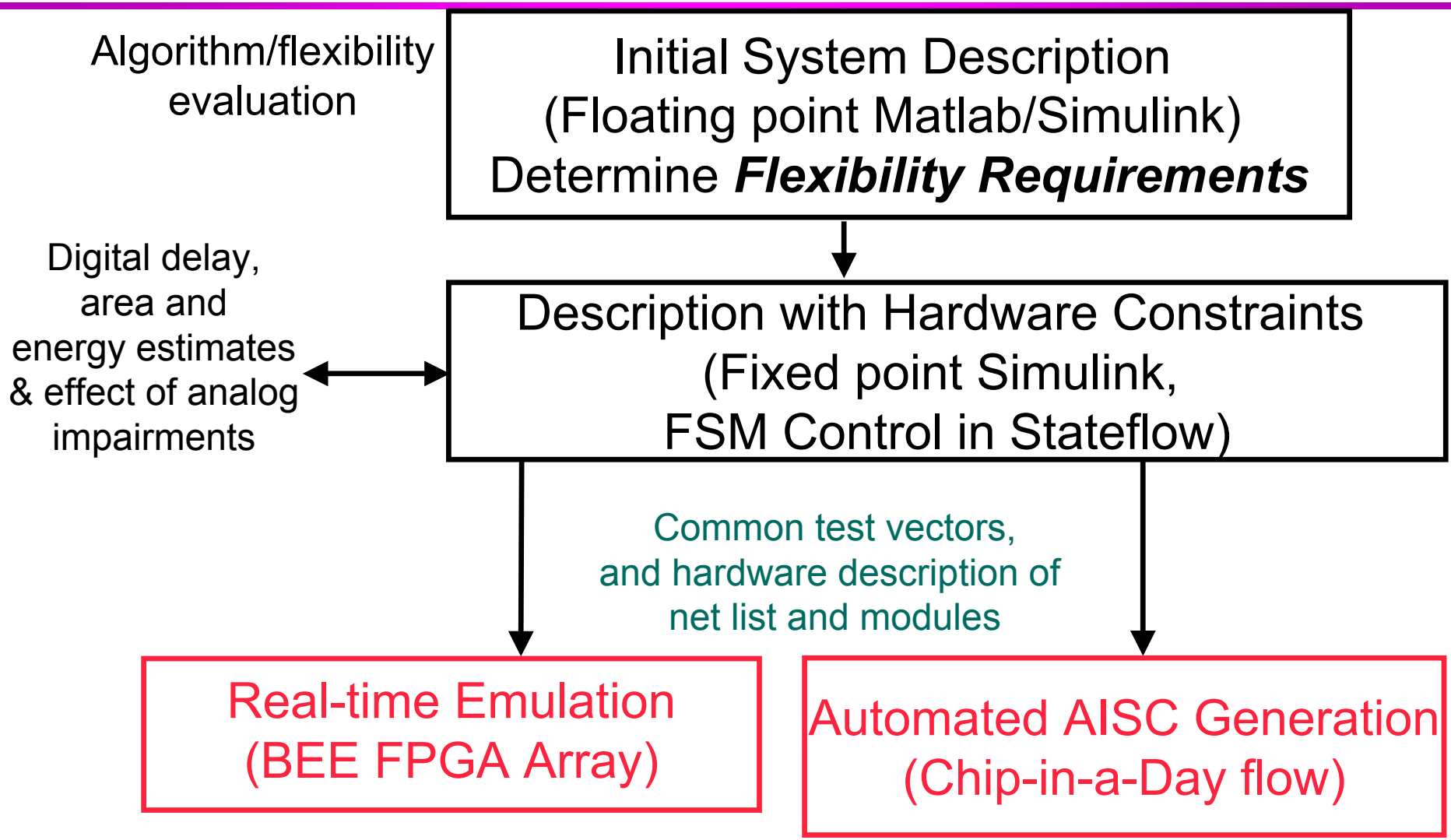
Output SNR ≈ 15dB

With Analog Impairments



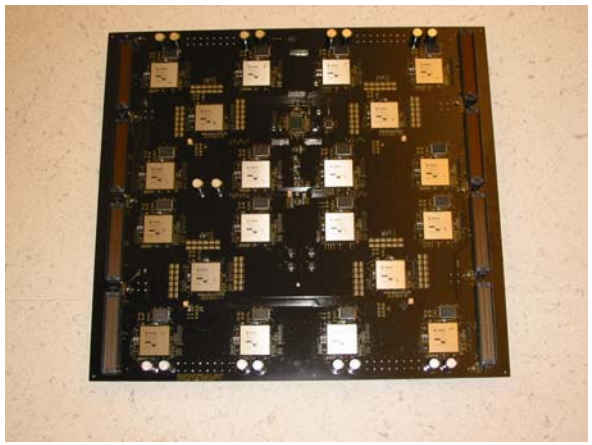
- 10 users (equal power)
- 20MHz Butterworth LPF
- 500kHz DC notch filter
- 13.5dB receiver NF
- 82dB gain
- 4% gain mismatch
- 2.5° I/Q phase mismatch
- IIP2 = -11dBm
- IIP3 = -18dBm
- PLL: -80dBc/Hz @ 100kHz
- 10-bit, 200MHz S-D ADC

Now to implement that description

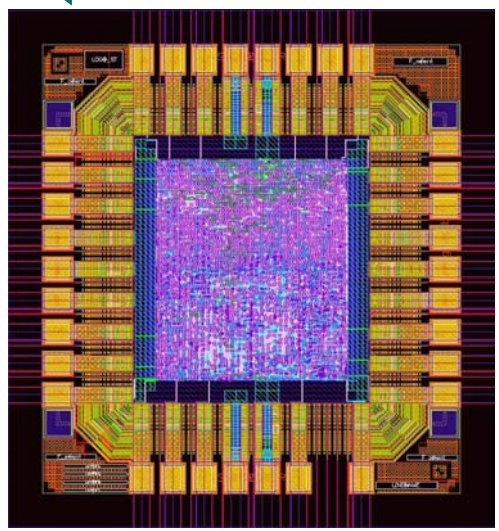


Single description – Two targets

Simulink/Stateflow Description



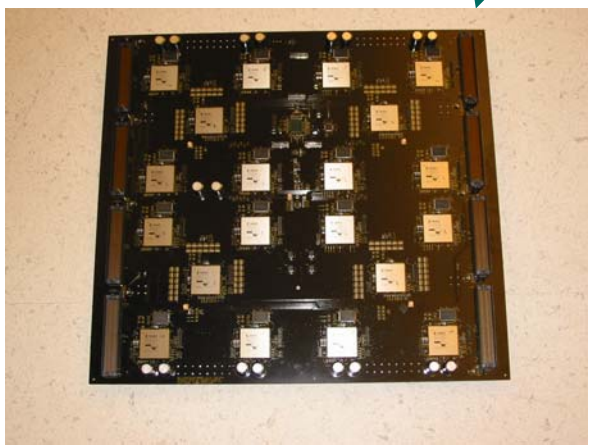
BEE
FPGA Array



ASIC Implementation
“Chip in a day”

BEE Target for Real-time emulation

Simulink/Stateflow Description



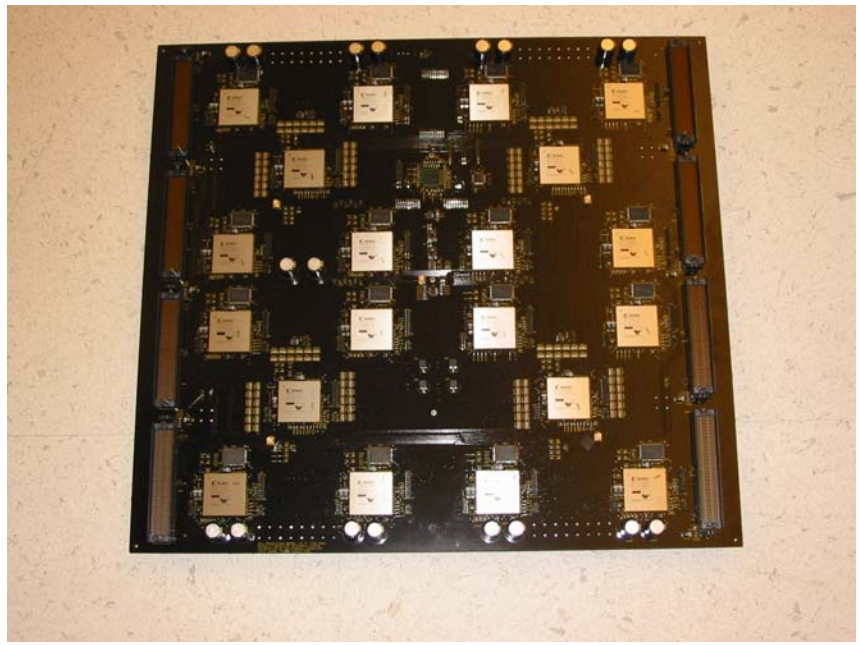
BEE
FPGA Array

BEE Design flow Goals

- Fully automatic generation of FPGA and ASIC implementations from Simulink system level design
- Cycle accurate bit-true functional level equivalency between ASIC & BEE implementation
- Real-time emulation controlled from workstation

Processing Board PCB

- Board-level Main Clock Rate: 160MHz+
- On Board connection speed:
  - » FPGA to FPGA: 100MHz
  - » XBAR to XBAR: 70MHz
- Off board connection speed: (3 ft SCSI cable loop back through riser card)
  - » LVTTL: 40MHz
  - » LVDS: 160MHz ~ 220MHz

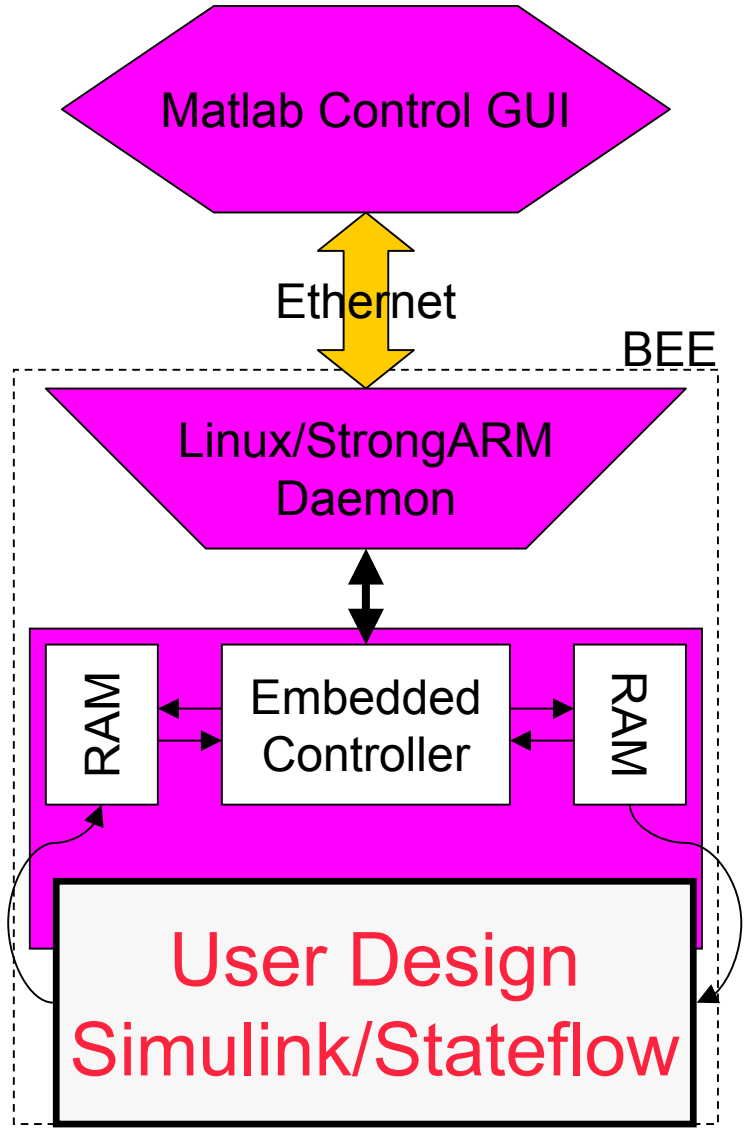


- Board Dimension: 53 X 58 cm
- Layout Area: 427 sq. in.
- No. of Layers: 26

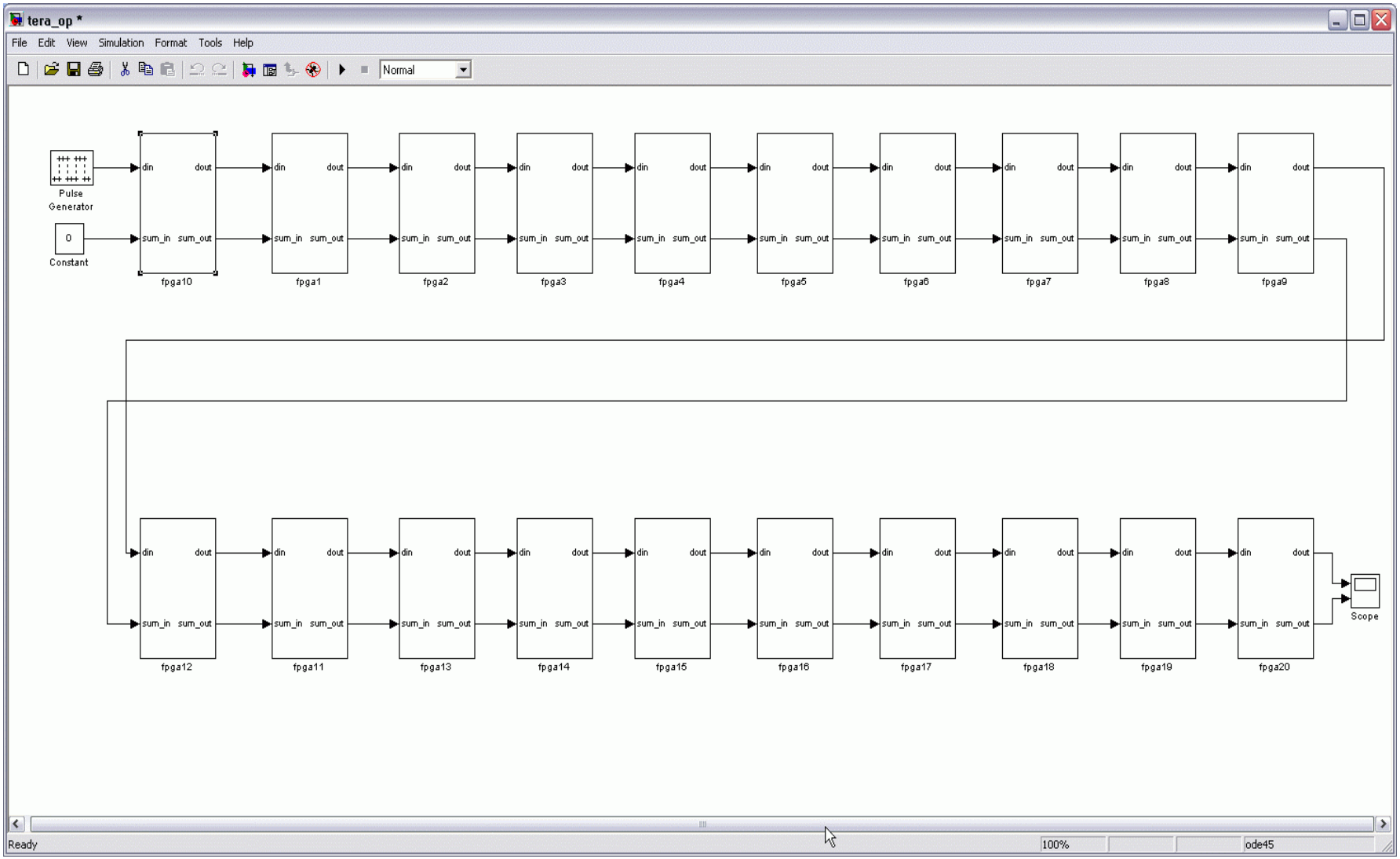
The BEE with RF transceiver I/O

Run-time Data I/O Interface

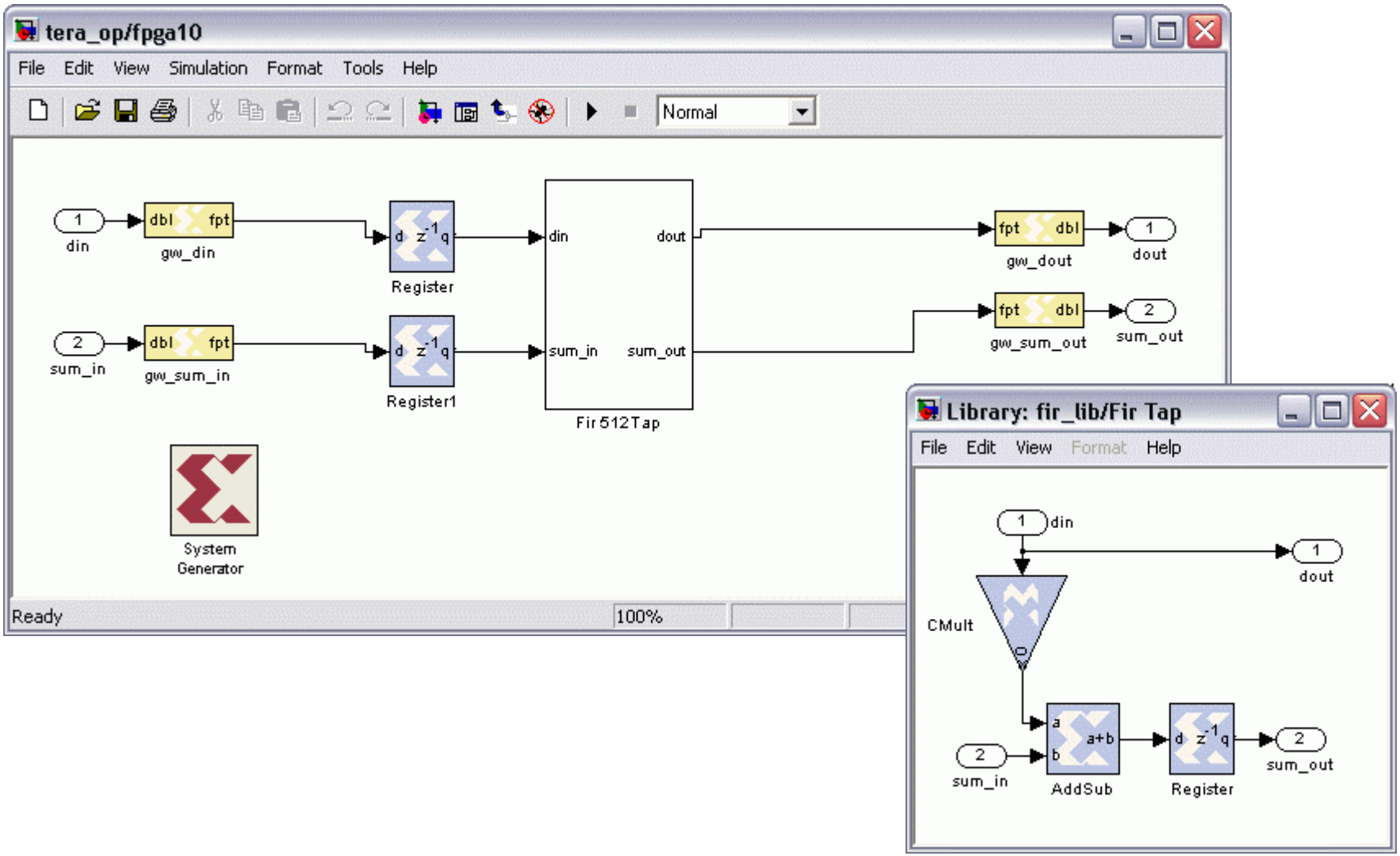
- Infrastructure for transferring data to and from the BEE
  - » The entire hardware interface is in one fully parameterized block
  - » Simply drop block into the Simulink diagram
  - » Accepts standard Simulink data structures for reuse of existing test vectors



Benchmark: 10240 Tap Fir Design



10240 Tap Fir Design (cont.)



BEE Performance

- Reference Design:
  - 10240 tap FIR filter
  - 512 taps per FPGA
- Slice utilization: 99% of 19200 slices
- Max Clock Rate: 30 Hz
- MOPS: 580,000 MOPS total (16bit add & 12bit cmult)
- Power: 2.5W per FPGA, 50W total

Comparison with an ASIC version using .13 micron chip metrics of 5000 MOPS/mm^2, 1000 MOPS/mW =>

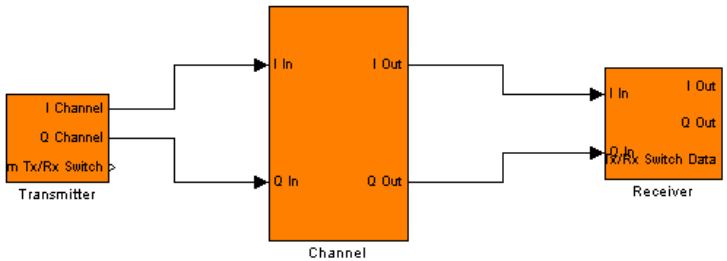
The BEE is equivalent to a single chip of 50 mm^2 with power = 500 mW.

50 Watts/500 mW => 100 times more power

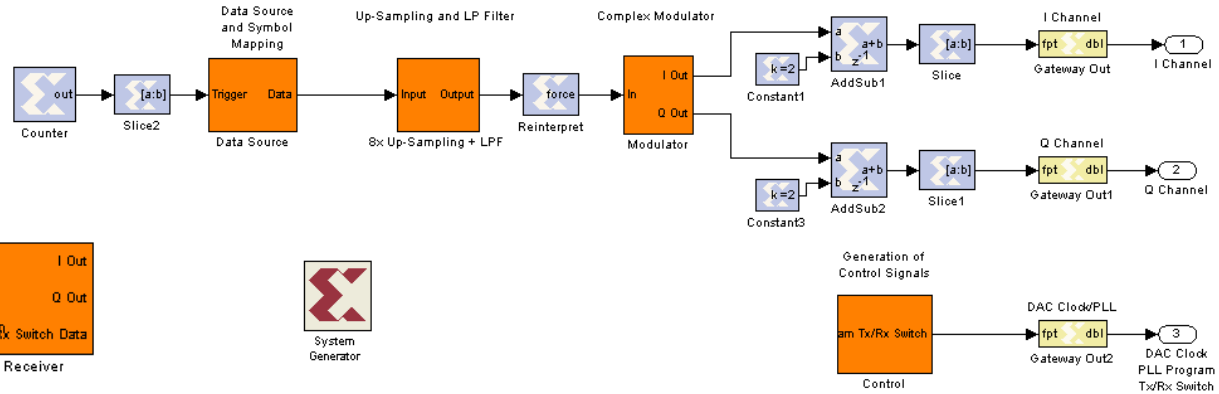
(20 ∗2 cm$^{2)}$/.5 => 100 times more area

Implementation of a Narrow-Band Radio System (Hans Bluethgen)

Complete System

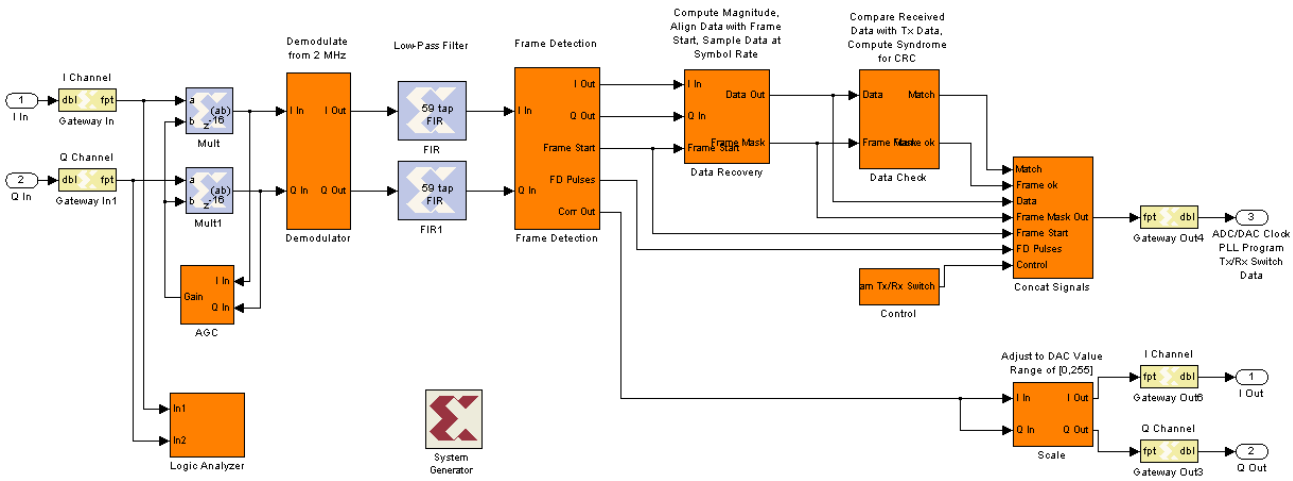


Transmitter

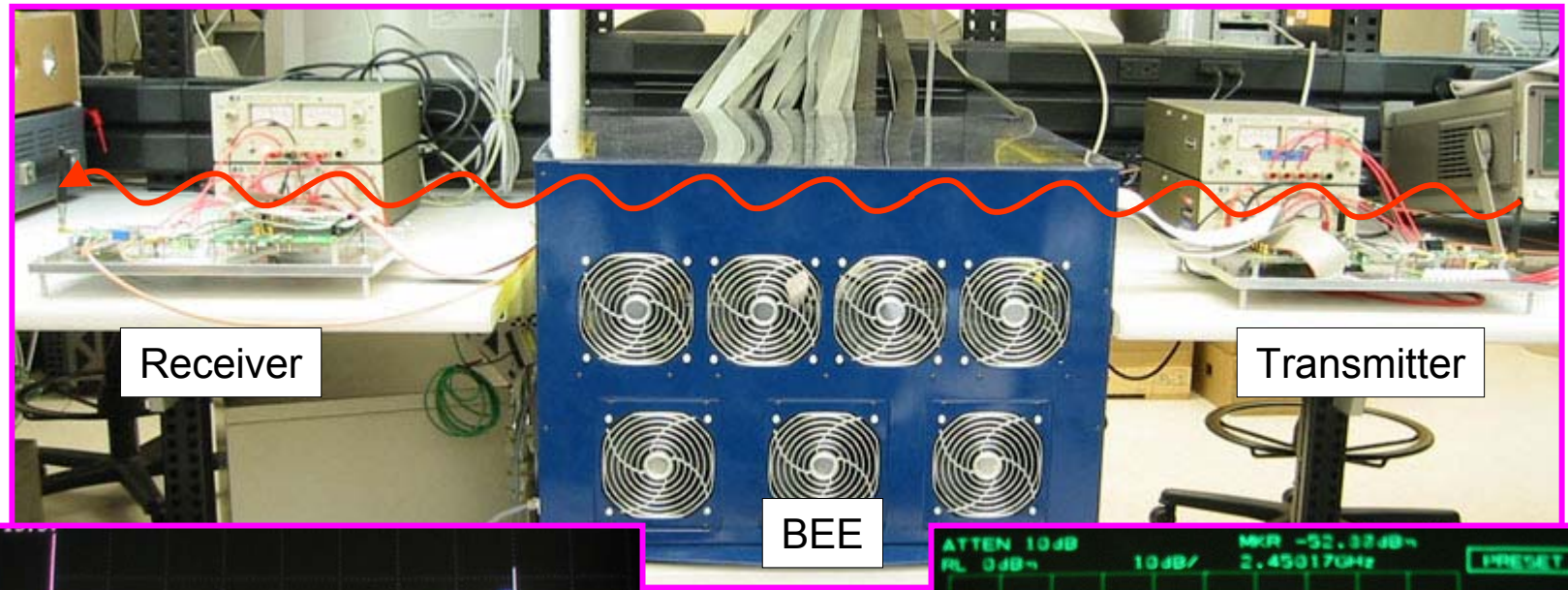


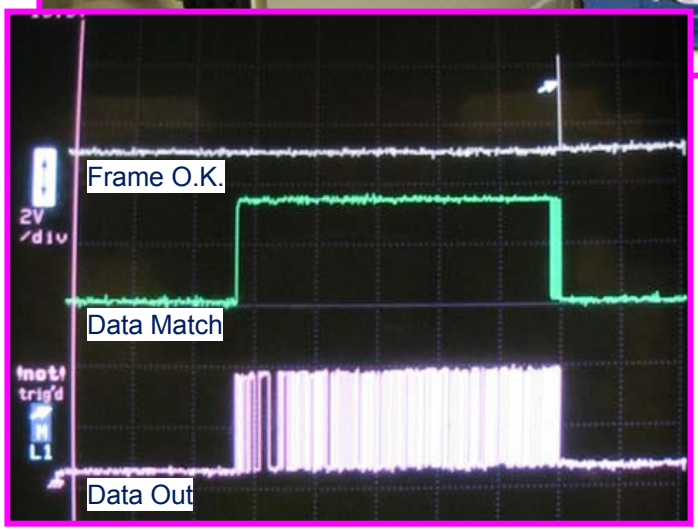
Data Rate	1 Mbit/s, 500 Kbit/s
Carrier Freq.	2.45 GHz
Bandwidth	1 MHz
Modulation	DPSK
Frame Synch.	PN Sequence

Receiver



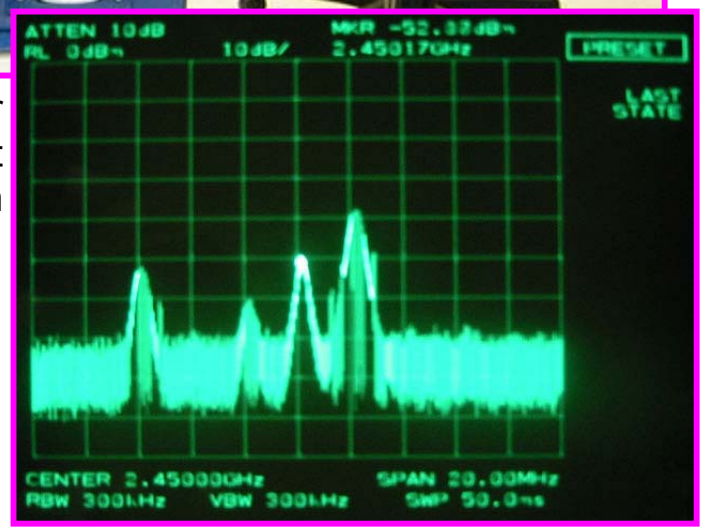
BEE Implementation of a Narrow-Band Radio



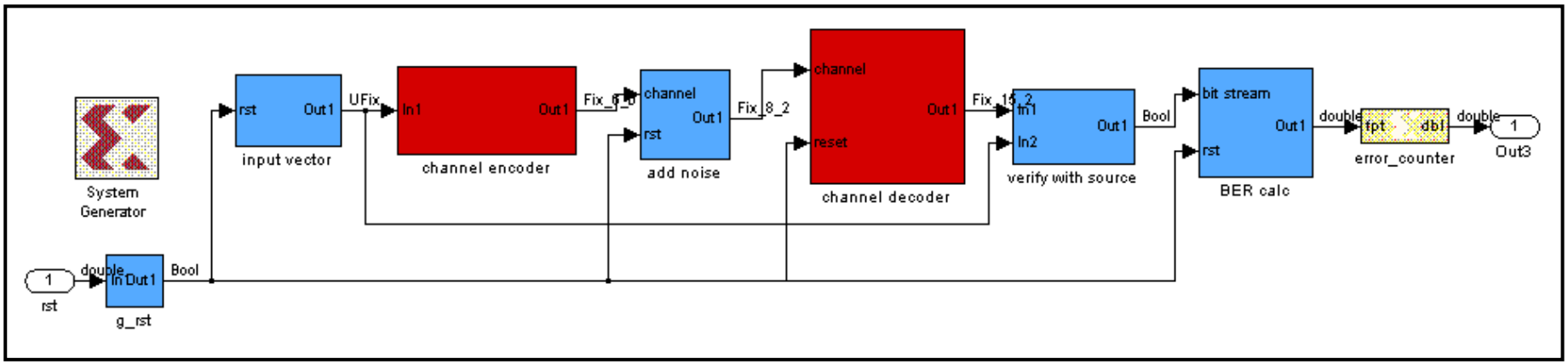


Receiver Output on SCSI Connector

Transmitter Output Spectrum

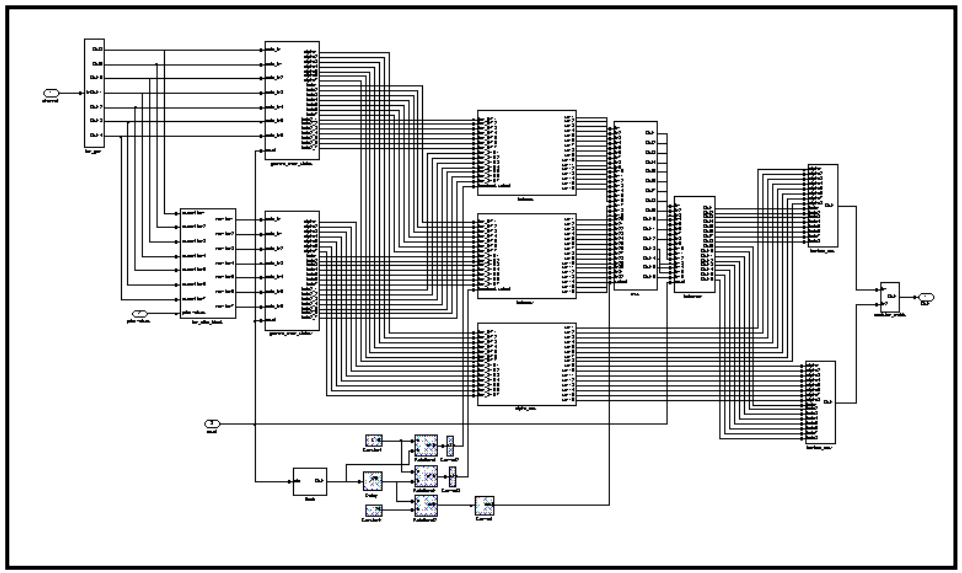


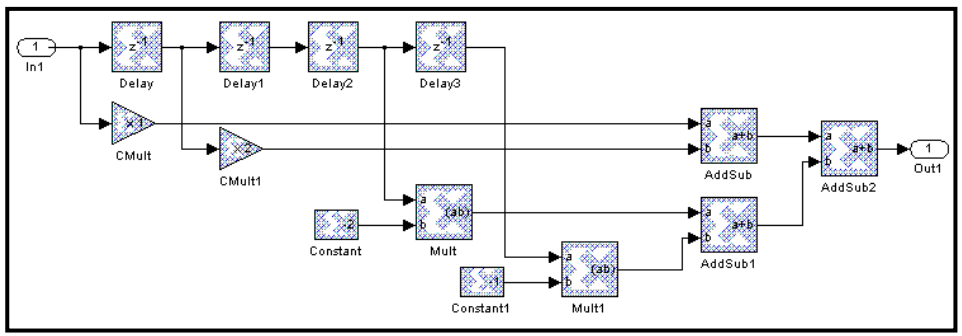
3G Turbo Decoder (Bora Nikolic)



- Complete description of ECC with variable noise levels to evaluate performance
- 10 MHz system clock
- SNR 14db $\rightarrow$ -1db
- 10^9 Samples in two minutes
- Parameterized to support variable binary point precision, SNR, number of samples for architectural evaluation

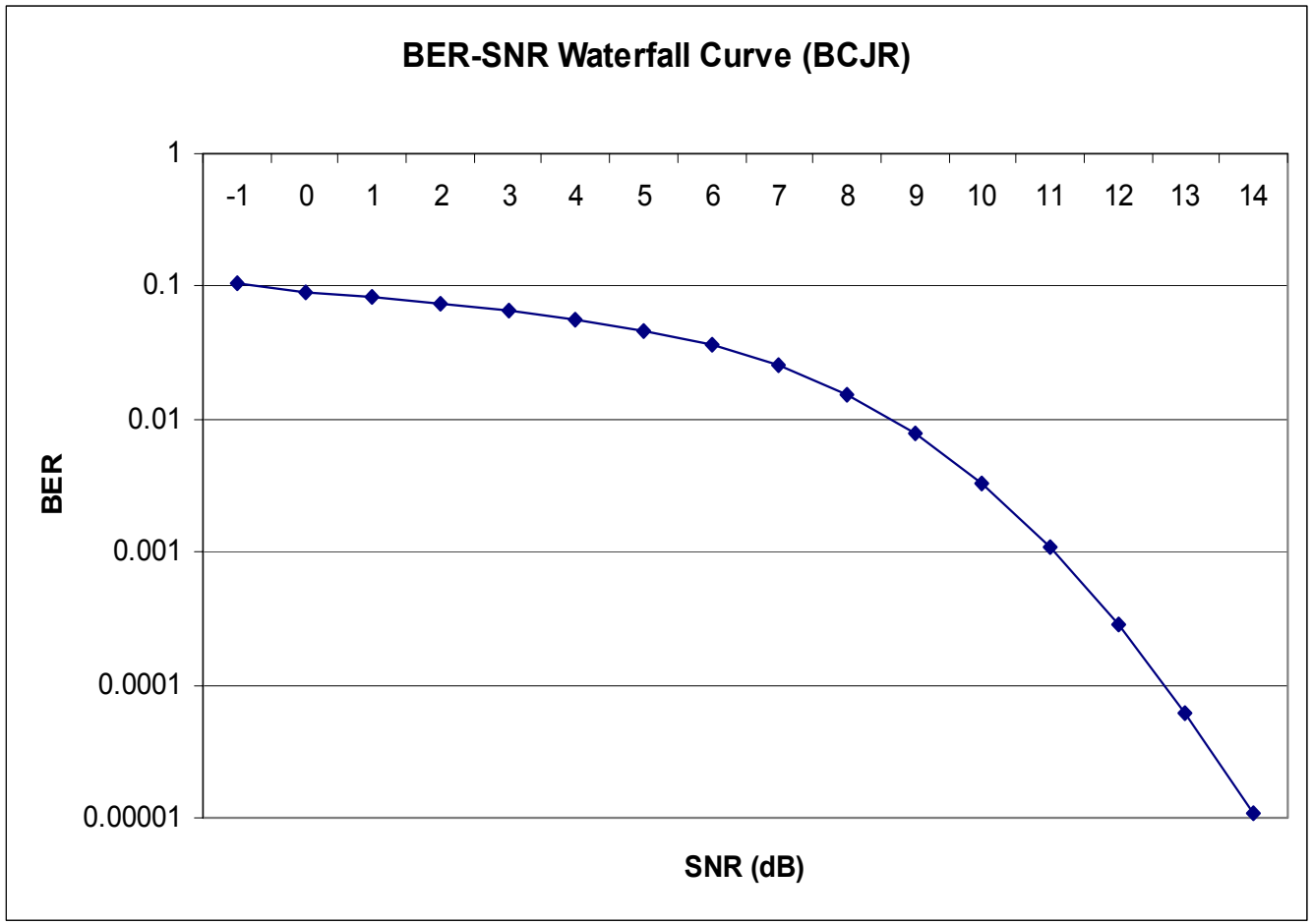
BCJR Simulink simulation





- E2PR4 Channel Encoder - Decoder
- Fully enclosed design
  - » Uniform RNG input vector
  - » Channel encoder
  - » AWGN filter
  - » Channel decoder
  - » BER collection mechanism
- Part of: Full 3G Turbo Decoder

BCJR Waterfall Curve

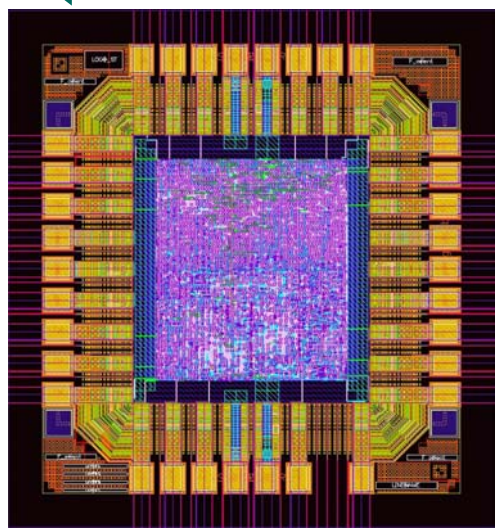


10MHz, 10^9 Samples, 1 bit binary point precision

Total simulation: approx. 10 minutes

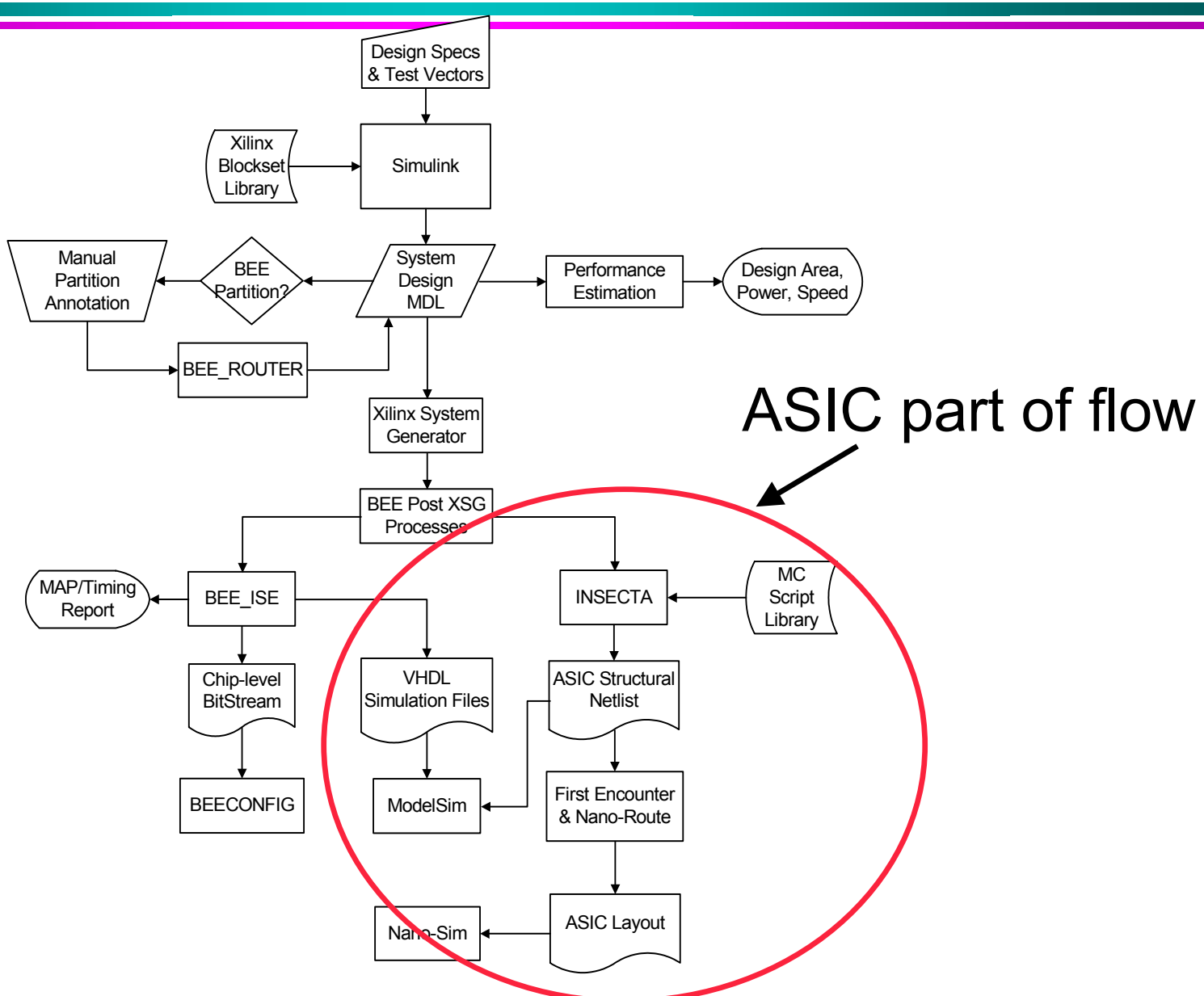
ASIC Target

Simulink/Stateflow Description



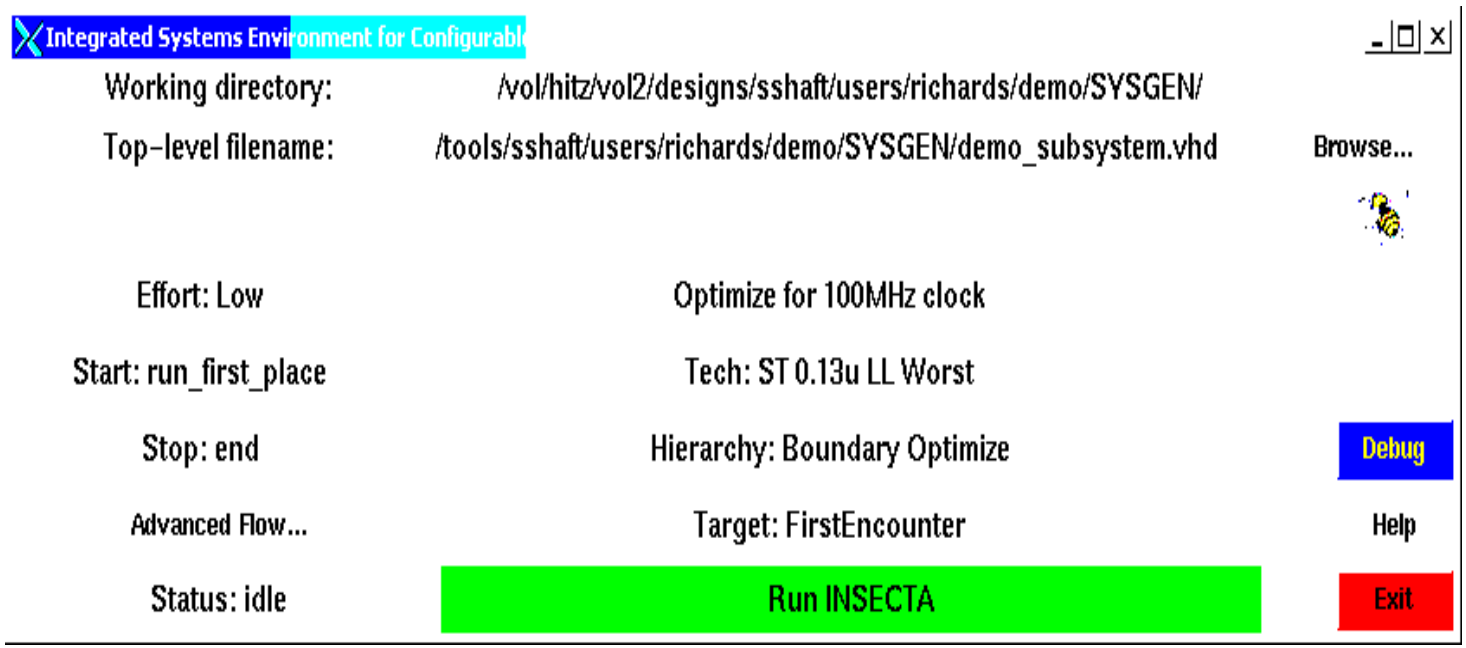
ASIC Implementation
"Chip in a day"

Complete Design Flow

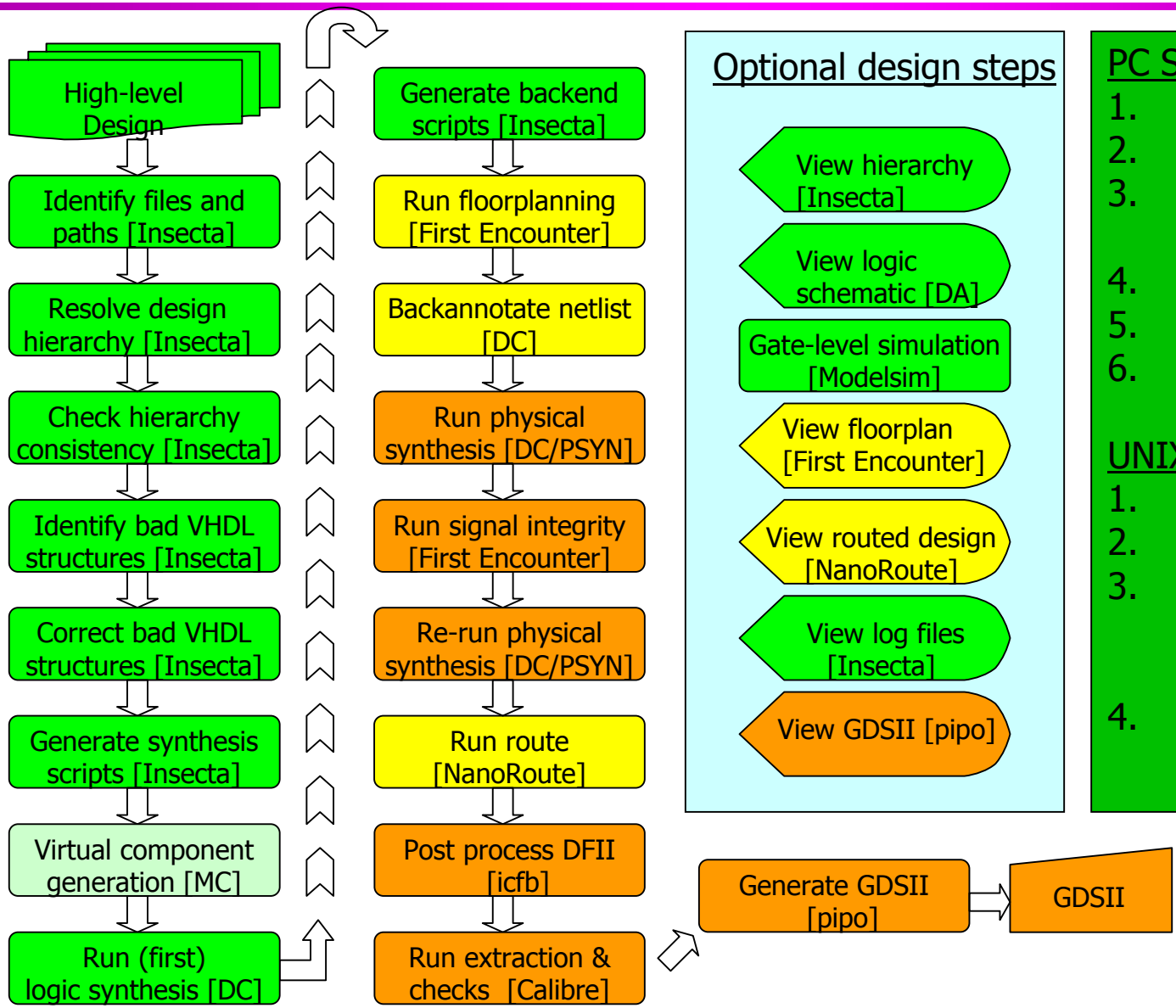


Chip-in-a-Day ASIC flow

- Tcl/Tk code drives the flow
  - » Used to drive multiple EDA tools: First Encounter, Nanoroute, Module Compiler
- GUI controls technology selection, parameter selection, flow sequencing
  - » A real “Push Button” flow…
  - » Users can refine flow-generated scripts



Automated ASIC flow tools



Optional design steps

- View hierarchy [Insecta]
- View logic schematic [DA]
- Gate-level simulation [Modelsim]
- View floorplan [First Encounter]
- View routed design [NanoRoute]
- View log files [Insecta]
- View GDSII [pipo]

PC Software

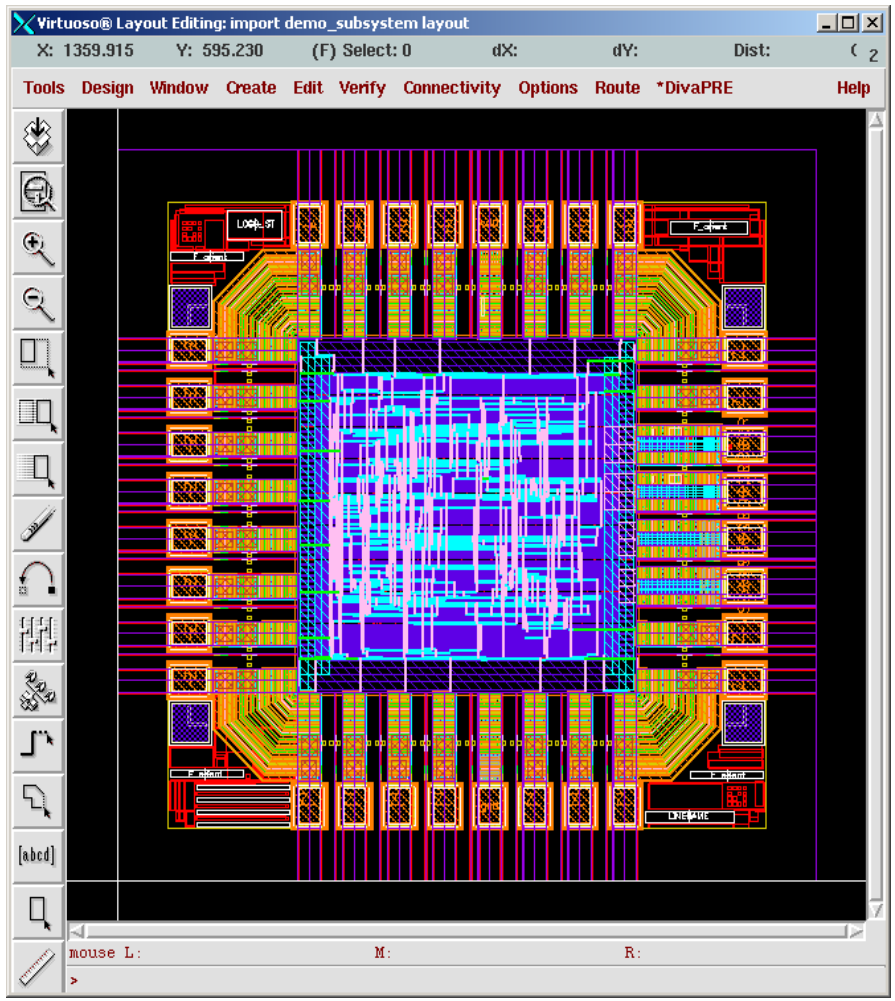
1. Matlab R13 (6.5)
2. Xilinx ISE
3. Xilinx System Generator 2.2
4. BEE ISE
5. Xilinx ChipScope
6. Xilinx Parallel Cable

UNIX SW Versions

1. TCL/TK 8.3
2. Synopsys 2002.05
3. Cadence SoC Encounter 2.2 (Nanoroute)
4. Modelsim 5.6

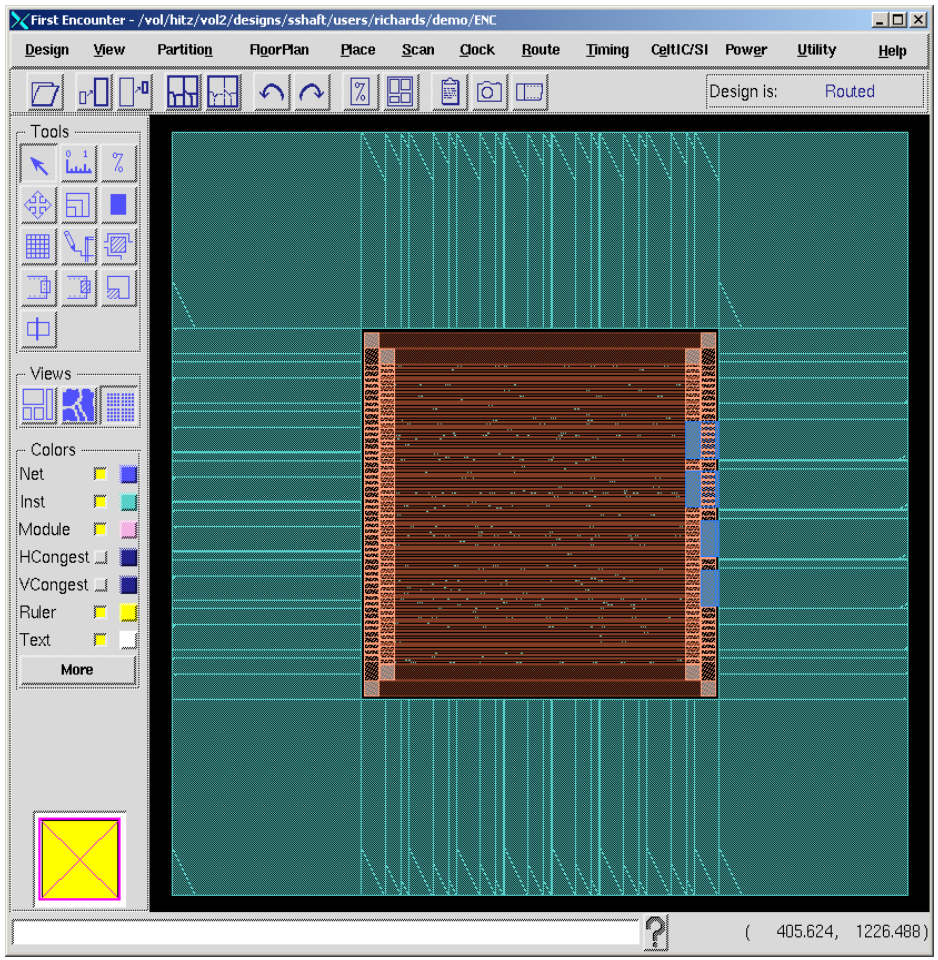
ASIC Flow: Back-end

- Using Unicad (ST Microelectronics) backend directly for DRC, LVS, Antenna rule checking
  - » Easier to track technology updates from foundry.
  - » Critical for evaluating internally developed technology files for FE, Nanoroute



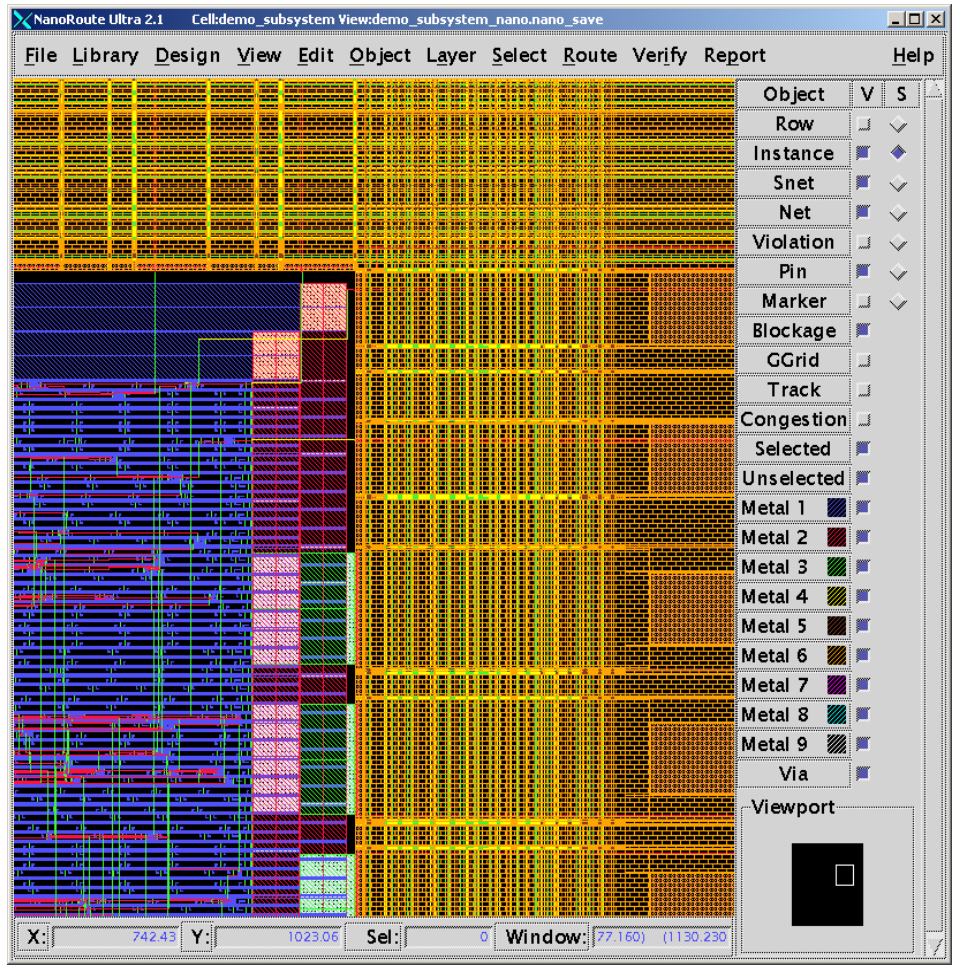
ASIC Tool Flow: Placement

- Cadence based flow
  - » First Encounter (FE)
  - » Nanoroute
- Timing Driven!
  - » FE provides accurate wire parasitic estimates
  - » Placement by FE



ASIC Flow: Routing in 130nm

- Nanoroute: Ready for 130nm, 90nm designs
  - » Stepped metal pitches
  - » Minimum area rules
  - » Complex VIA rules
  - » Avoids antenna rule violations
  - » Cross-talk avoidance: to be evaluated
- Silicon Ensemble: Fallback position
- Apollo tools: Possible alternative

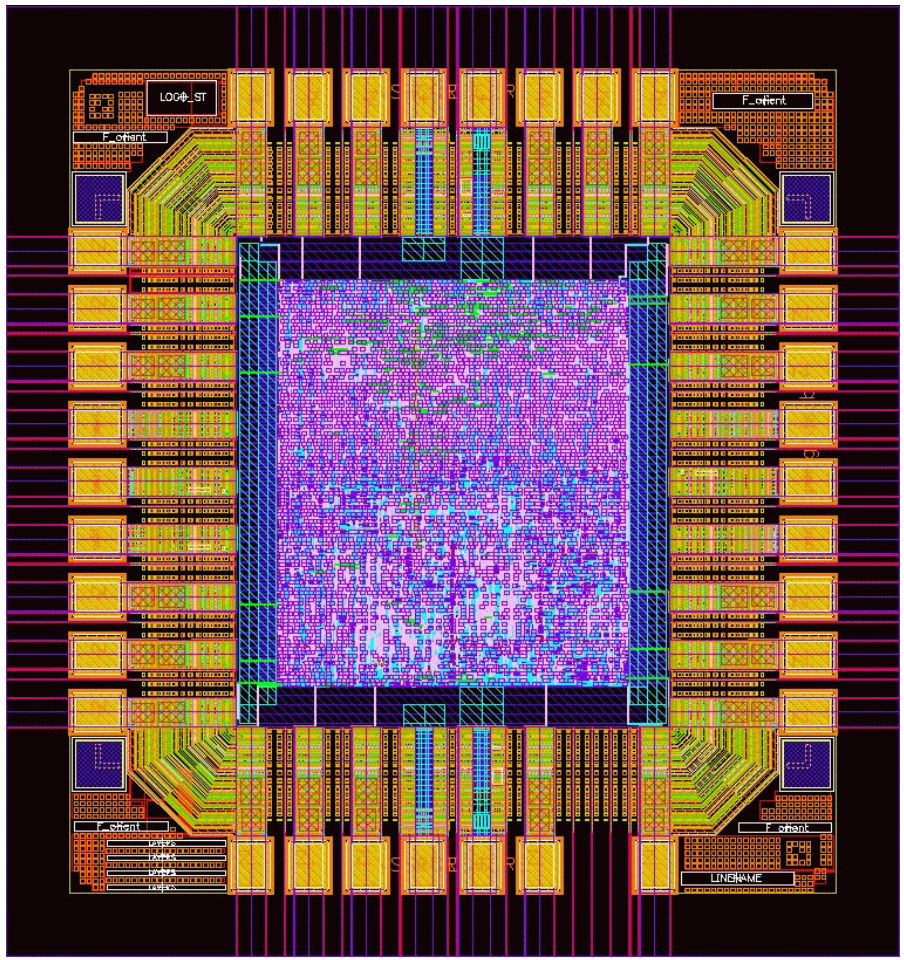


ASIC directly from Simulink – Narrowband Transmitter

CPU time: 57 min
Core Utilization: 0.344418 (Pad limited)
Size (From SoC Enconter):
  Core Height 565.8u
  Core Width 489.54u

  Die Height 1322.66u
  Die Width   1242.3u

Synopsys estimates:
Total Dynamic Power = 610.5163 uW (100%)
Cell Leakage Power = 15.9364 uW
Critical path: 9.21ns



The Issues I Addressed

- How much flexibility is needed and how best to include it…
  - » As little as possible consistent with business constraints
- A single system description including interaction between the analog and digital domains
  - » Timed dataflow plus state machines
- “Realtime” SOC prototyping
  - » FPGA configurability makes real-time prototyping possible in a fully parallel architecture.
- Automated ASIC design flow
  - » Certainly possible - the “chip in a day” flow