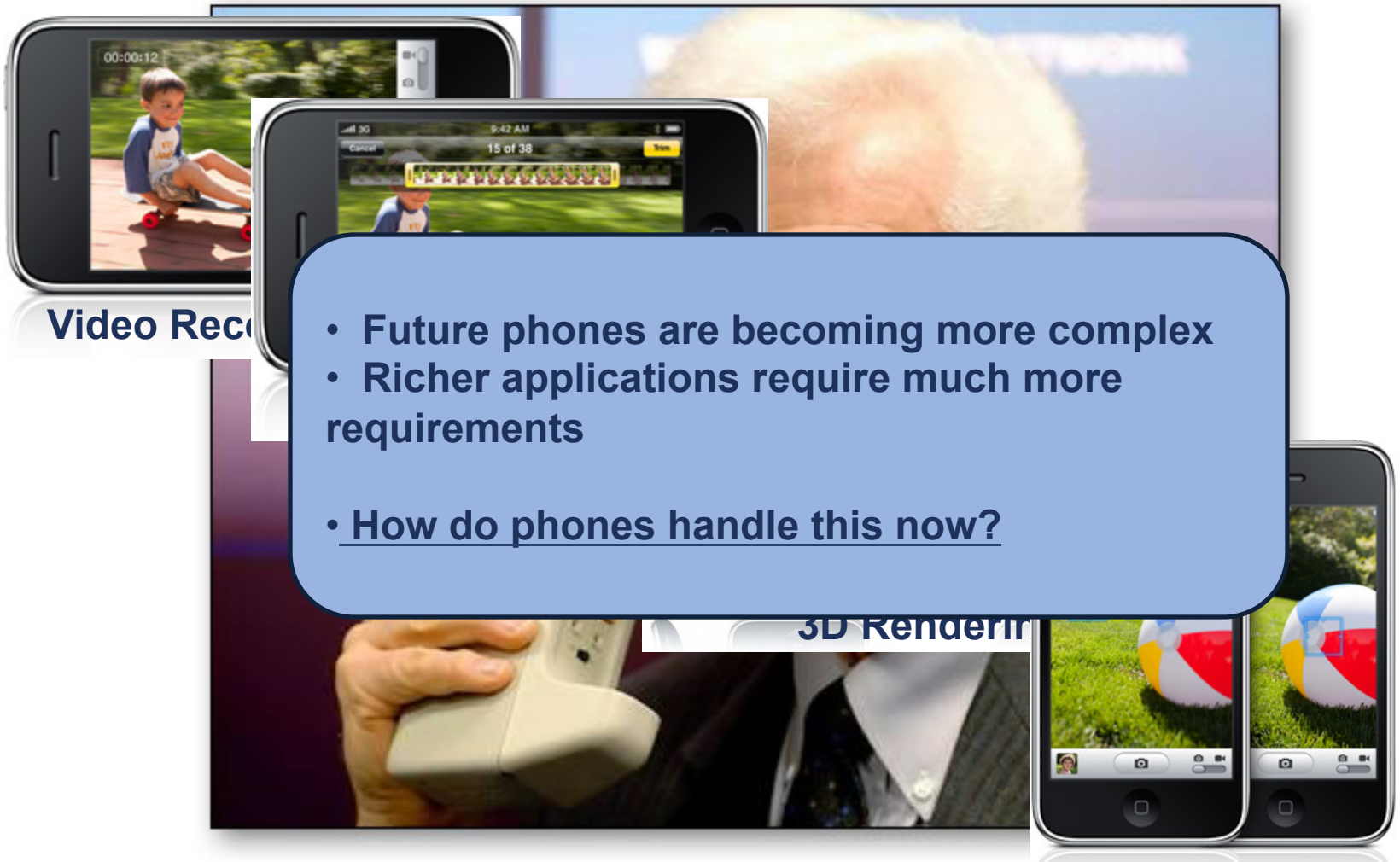
AnySP: Anytime Anywhere Anyway Signal Processing

Mark Woh[1], Sangwon Seo[1], Scott Mahlke[1],Trevor Mudge[1],
Chaitali Chakrabarti[2], Krisztian Flautner[3]

University of Michigan – ACAL[1]
Arizona State University[2]
ARM, Ltd.[3]

The Modern Mobile Phone

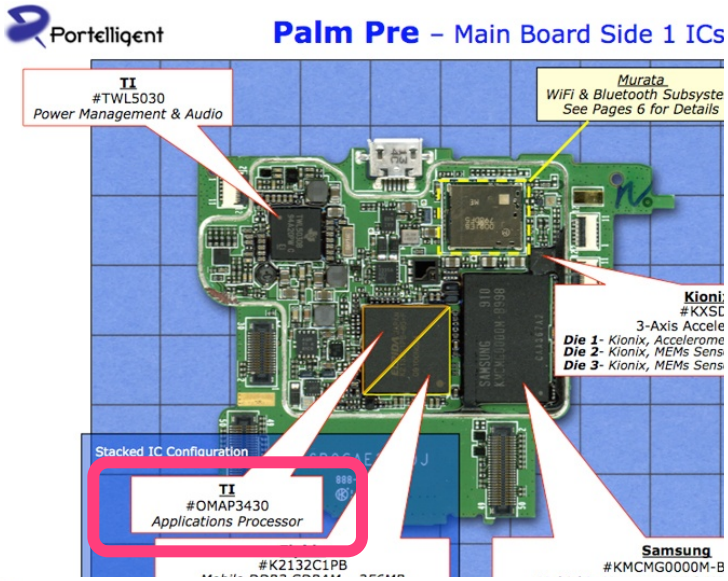


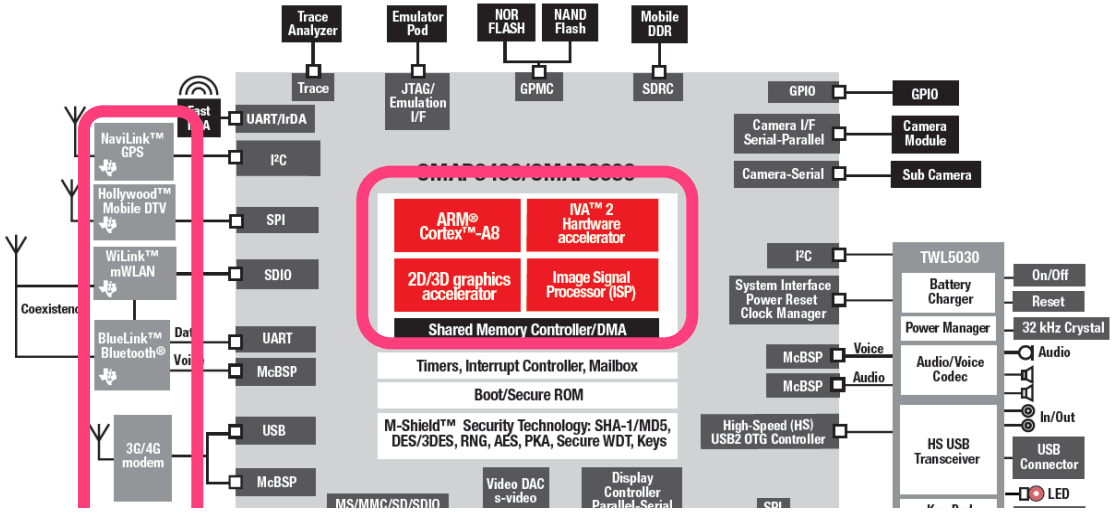
- Future phones are becoming more complex
- Richer applications require much more requirements

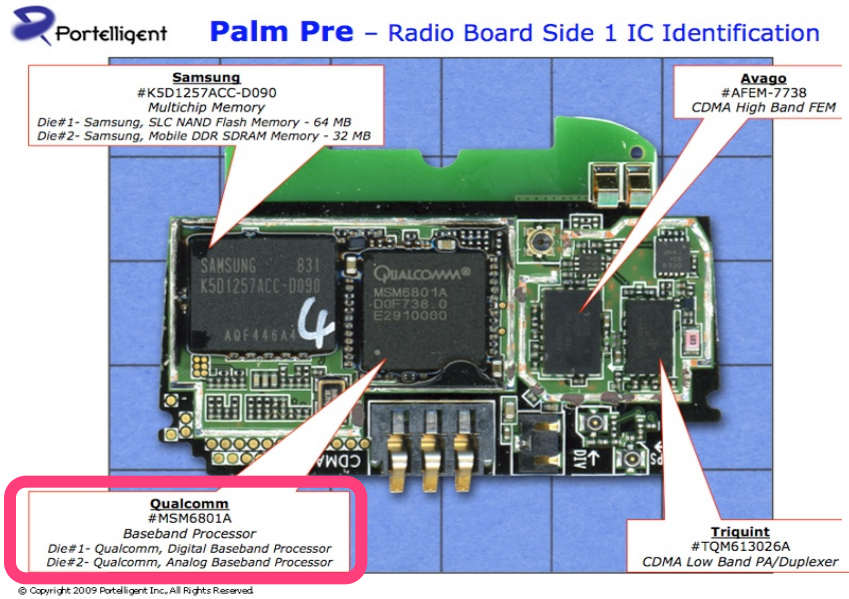
- How do phones handle this now?

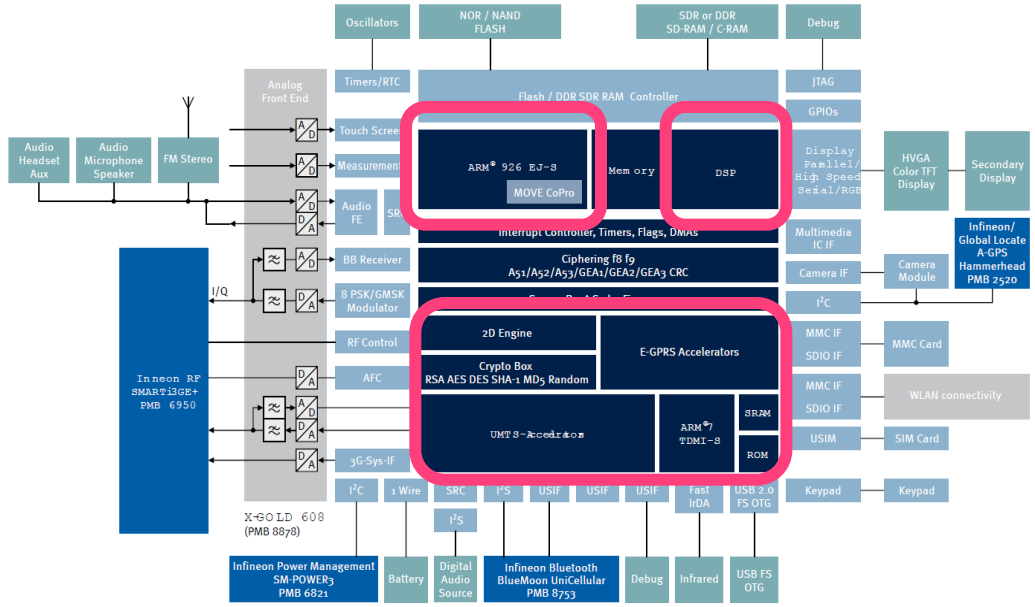
Photos From - http://www.engadget.com/2009/06/10/iphone-3g-s-supports-opengl-es-2-0-but-3g-only-supports-1-1/
http://www.apple.com/iphone

Inside Today's Smart Phones



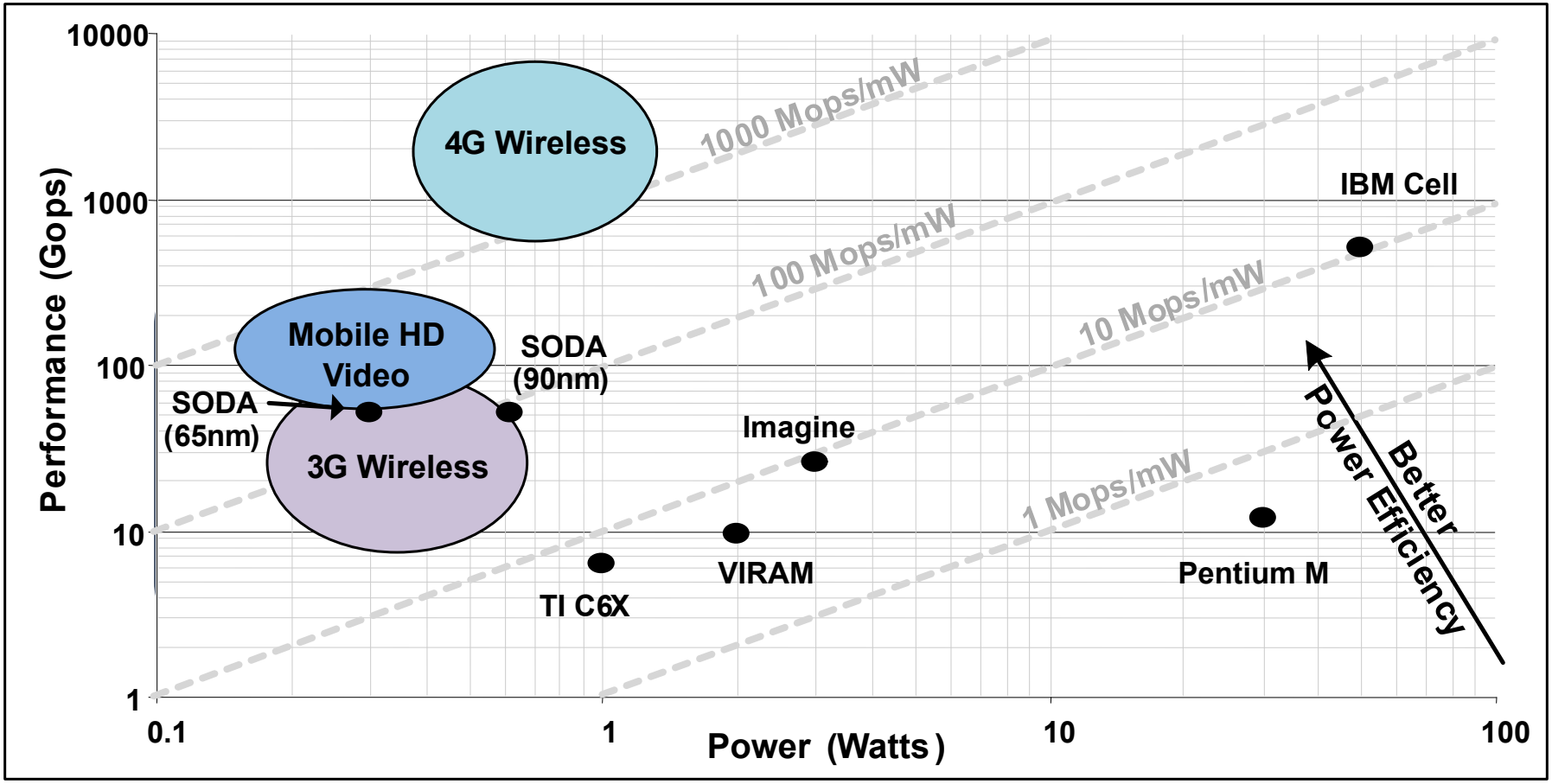






Inside the X-Gold 608 (Representation of QCOM)

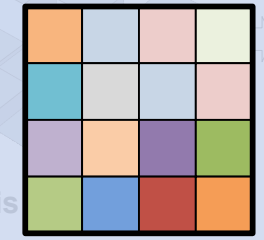
Power/Performance Requirements for Multiple Systems



The Applications

Is there anything we can learn from the applications themselves?

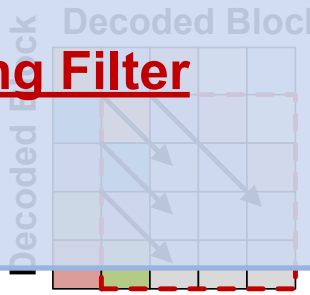
H.264 Basics



Reconstructed Macroblock

Modules	Algorithms	Power Profiling*
Entropy decoder	Context-adaptive VLD	4%
Inverse Transform	x4 integer kernel	
Motion Compensation	alf: 6-tap filter uarter: 6-tap / bilinear filter ighth: 4-tap filter	29%
Intra Prediction	irectional spatial prediction	10%
Deblocking Filter	n-loop adaptive filter	34%
Misc Kernels		23%

Deblocking Filter

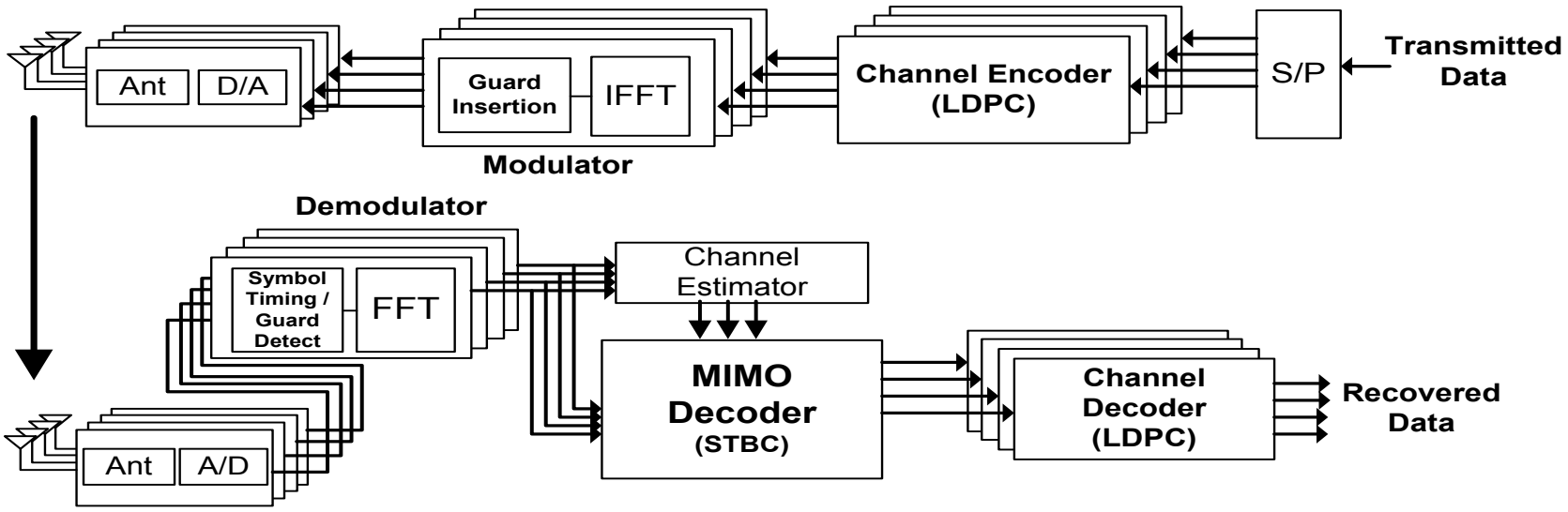


Current Uncoded Block

Intra-Prediction

T.-A. Liu, T.-M. Lin, S. -Z. Wang, et al. "A low-power dual-mode video decoder for mobile applications," *IEEE Communications Magazine*, volume 44, issue 8, pp.119-126, Aug. 2006.

4G Wireless Basics



- Three kernels make up the majority of the work
  - FFT – Extract Data from Signals
  - STBC – Combine Data into More Reliable Stream
  - LDPC – Error Correction on Data Stream

Mobile Signal Processing Algorithm Characteristics

	Algorithm	SIMD Workload (%)	Scalar Workload (%)	Overhead Workload (%)	SIMD Width (Elements)	Amount of TLP
4G	FFT	75	5	20	1024	Low
	STBC	81	5	14	4	High
	LDPC					Low
H.264	Deblo					Medium
	Intra-I					Medium
	Invers					High
	Motio					High

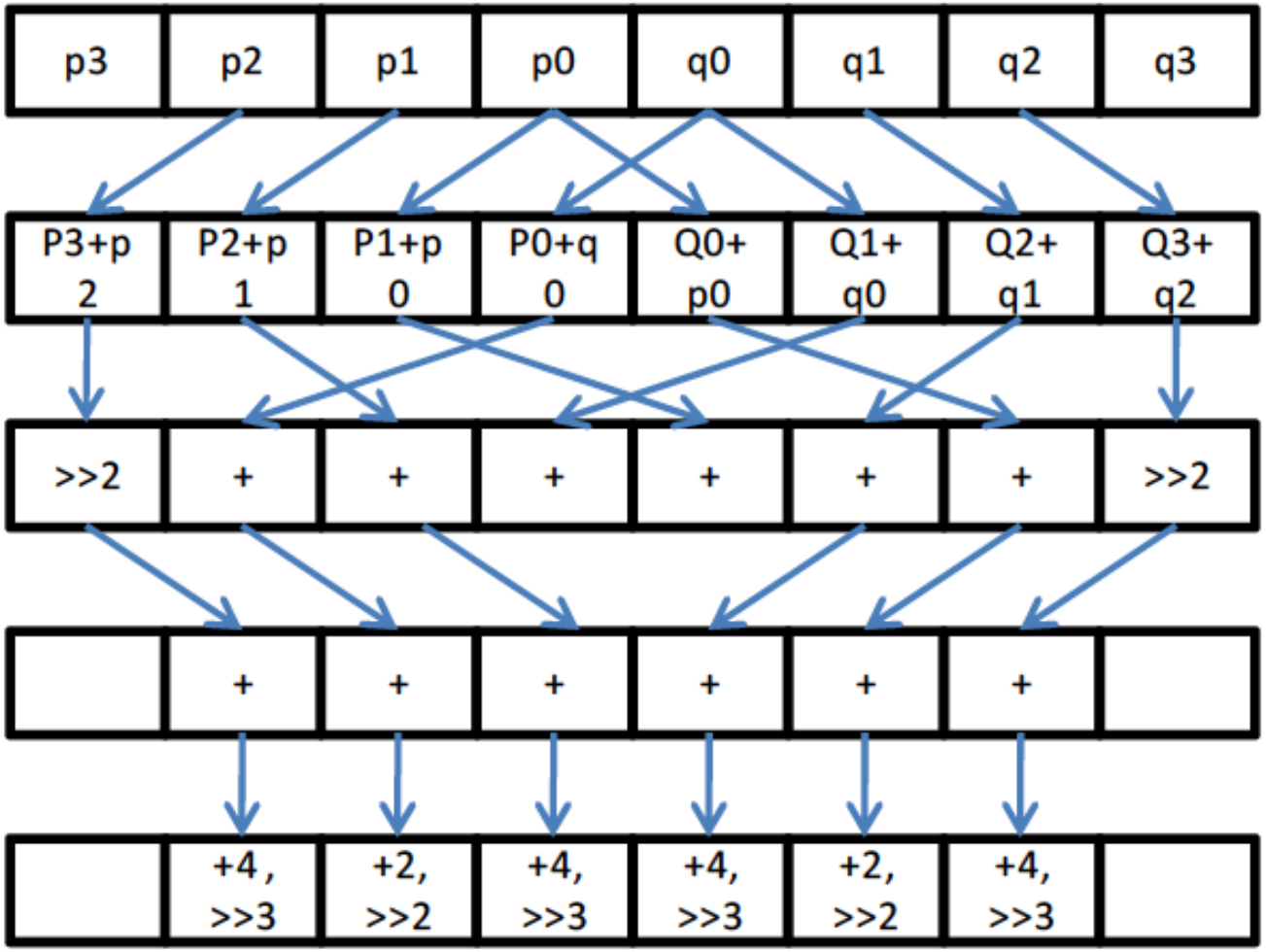
SIMD comes at a cost!
•Register File Power
•Data Movement/Alignment Cost
SIMD architectures have to deal with this!

- Algo
  - From very large to very small
- Though SIMD width varies all algorithms can exploit it
  - Large percentage of work can be SIMDized
- Larger SIMD width tend to have less TLP

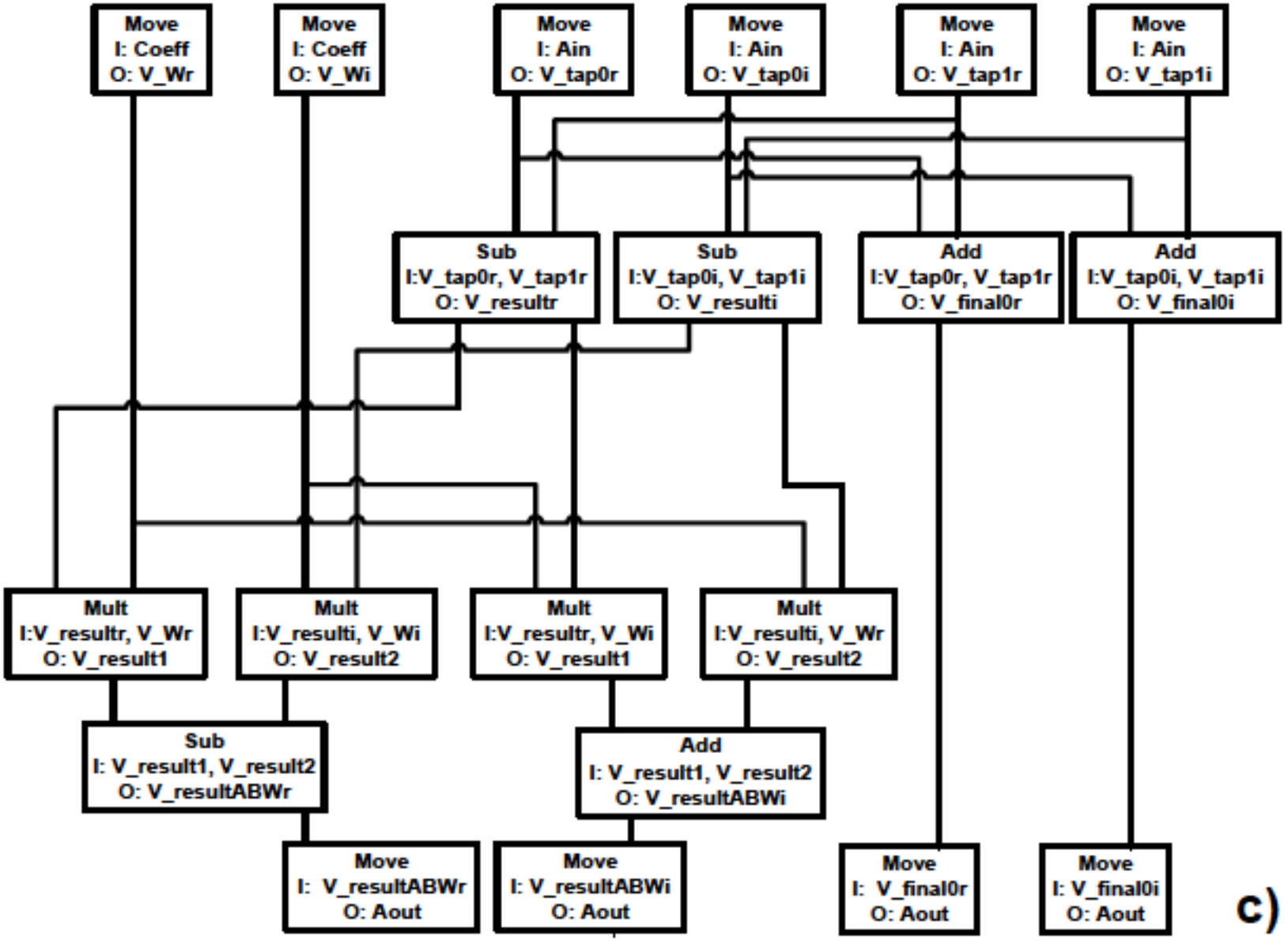
Only the instructions shown in red are MMX computations. All other instructions are simply supporting these computations.

Pentium III – SIMD code for Discrete Cosine Transform (DCT)

lea	ebx, DWORD PTR [ebp+128]	load/address overhead
mov	DWORD PTR [esp+28], ebx	load/address overhead
$B1$2:		
xor	eax, eax	address overhead
move	dx, ecx	address overhead
lea	edi, DWORD PTR [ecx+16]	load/address overhead
mov	DWORD PTR [esp+24], ecx	load/address overhead
$B1$3:		
movq	mm1, MMWORD PTR [ebp]	load overhead
pxor	mm0, mm0	initialization overhead
pmaddwd	**mm1, MMWORD PTR [eax+esi]**	**True Computation**
movq	mm2, MMWORD PTR [ebp+8]	load overhead
pmaddwd	**mm2, MMWORD PTR [eax+esi+8]**	**True Computation**
add	eax, 16address	overhead
paddw	**mm1, mm0**	**True Computation**
paddw	**mm2, mm1**	**True Computation**
movq	mm0, mm2	load related overhead
psrlq	mm2, 32	SIMD reduction overhead
povd	ecx, mm0	SIMD load overhead
movd	ebx, mm2	SIMD load overhead
add	ecx, ebx	SIMD conversion Overhead
mov	WORD PTR [edx], cx	store overhead
add	edx, 2	address overhead
cmp	edi, edx	branch related overhead
jg	$B1$3	loop branch overhead
$B1$4:		
move	cx, DWORD PTR [esp+24]	load/address overhead
add	ebp, 16	address overhead
add	ecx, 16	address overhead
move	ax, DWORD PTR [esp+28]	load/address overhead
cmp	eax, ebp	branch related overhead
jg	$B1$2	loop branch overhead

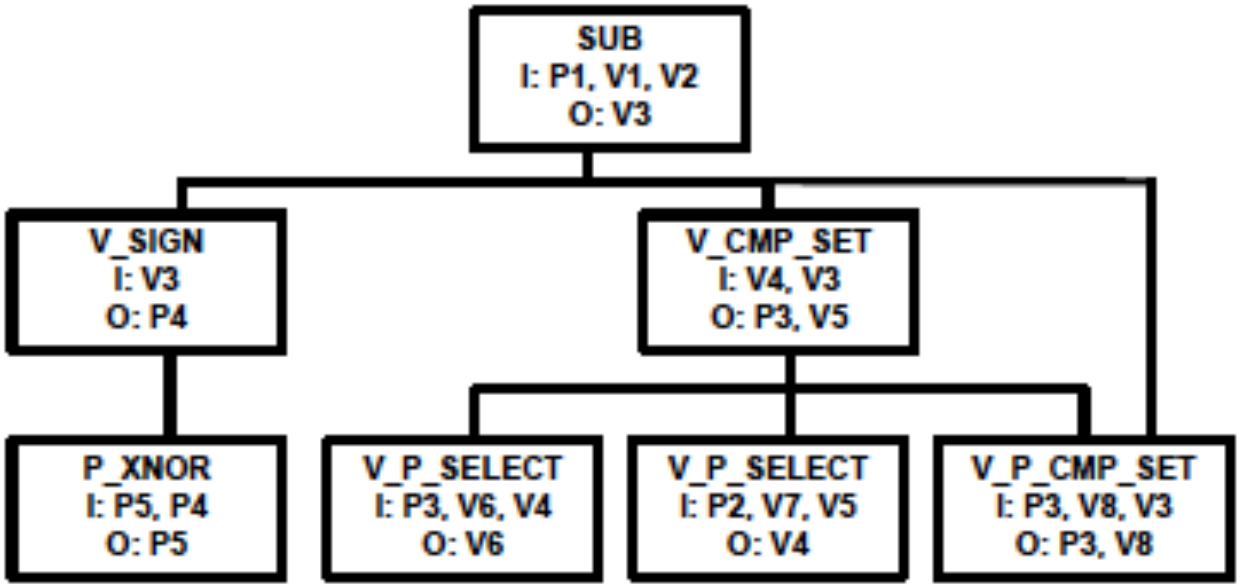


a) Deblocking Filter Subgraph



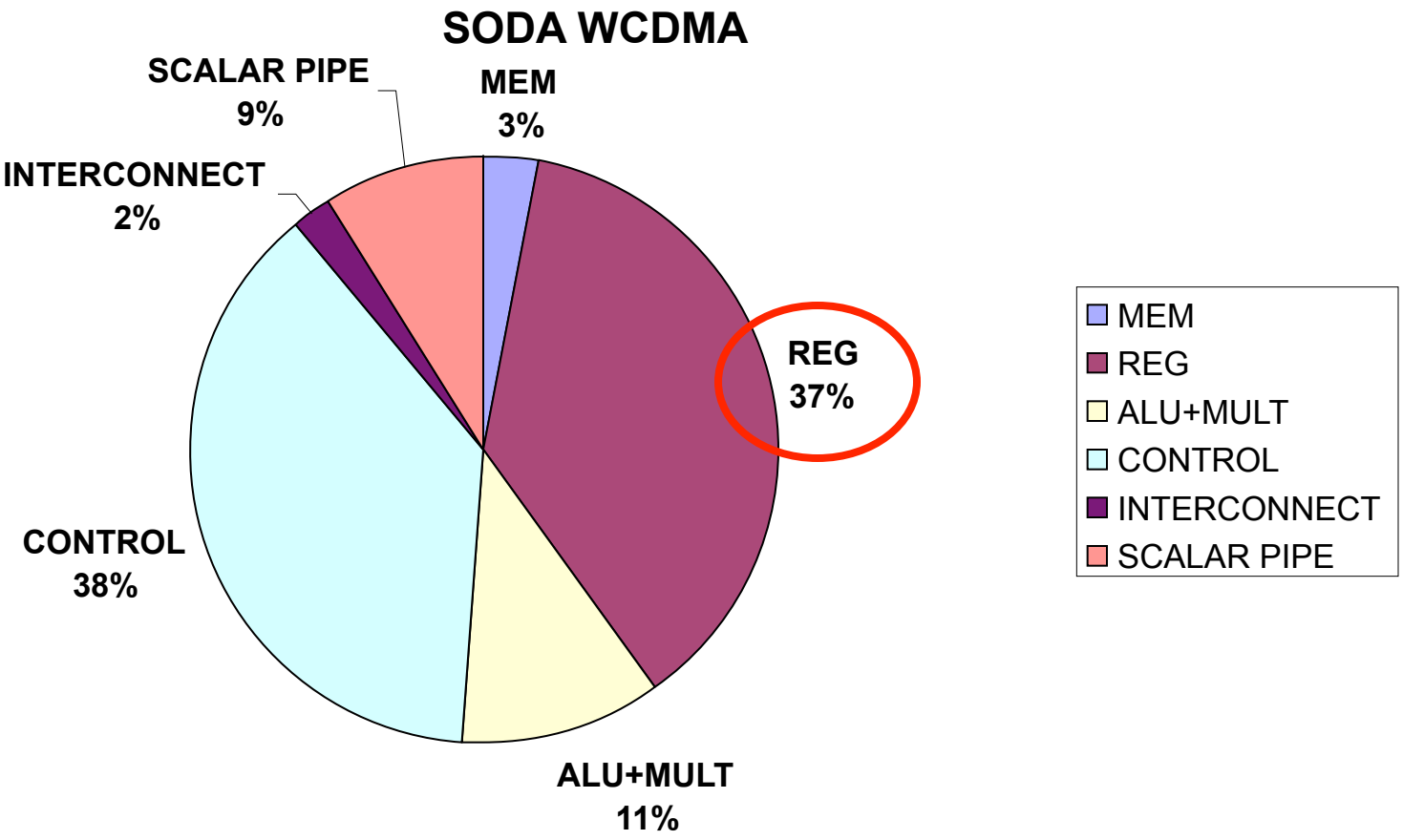
b) FFT Subgraph

c)



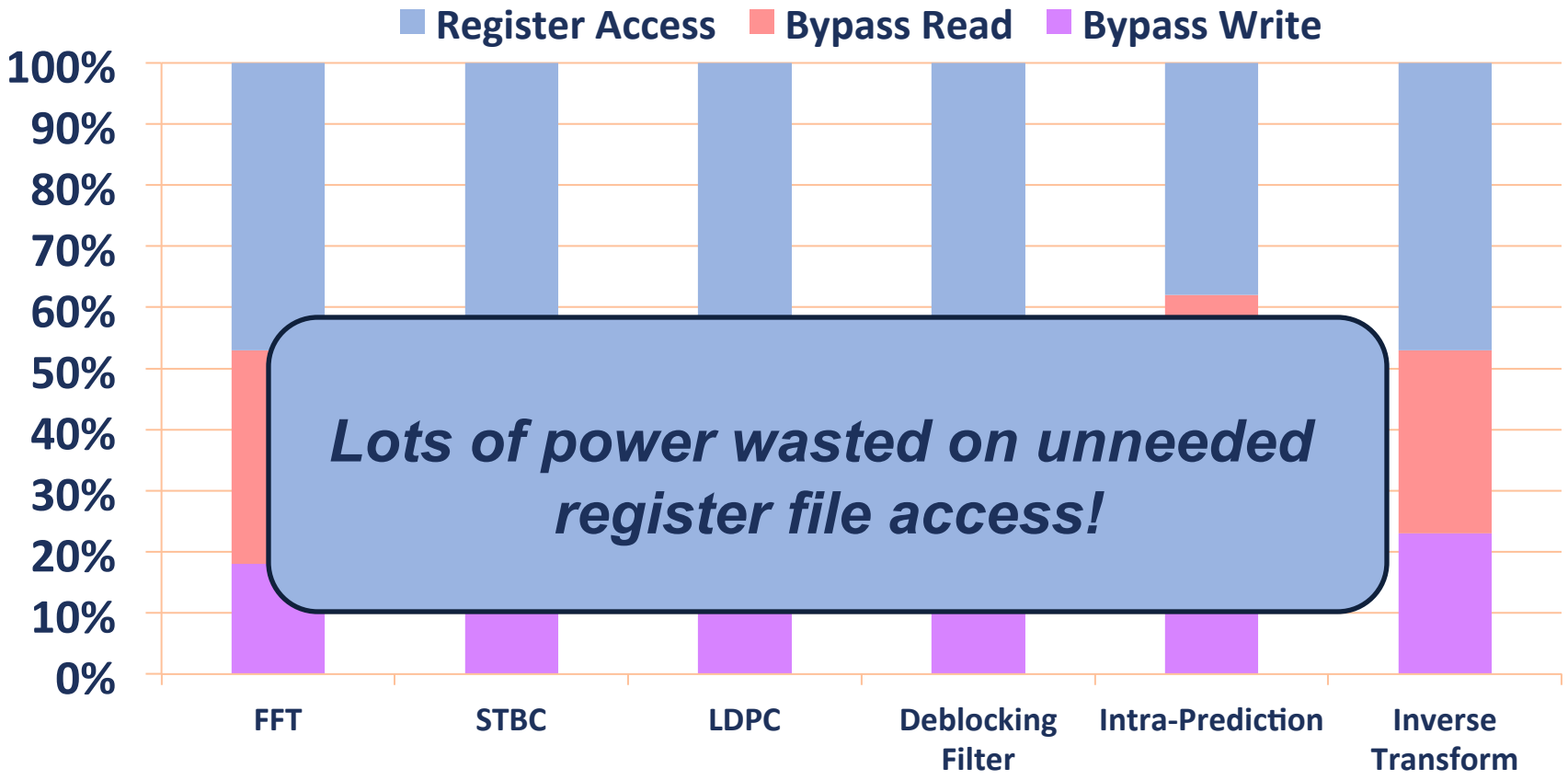
c) Subgraph for Bit Node and Check Node Operation

Traditional SIMD Power Breakdown



- Register File Power consumes a lot of power in traditional 32-wide SIMD architecture

Register File Access



- Many of the register file access do not have to go back to the main register file

Instruction Pair Frequency

	Instruction Pair	Frequency
1	multiply-add	26.71%
2	add-add	13.74%
3	shuffle-add	8.54%
4	shift right-add	6.90%
5	subtract-add	6.94%
6	add-shift right	5.76%
7	multiply-subtract	4.00%
8	shift right-subtract	3.75%
9	add-subtract	3.07%
10	Others	20.45%

a) Intra-prediction and Deblocking Filter Combined

	Instruction Pair	Frequency
1	shuffle-move	32.07%
2	abs-subtract	8.54%
3	move-subtract	8.54%
4	shuffle-subtract	3.54%
5	add-shuffle	3.54%
6	Others	43.77%

b) LDPC

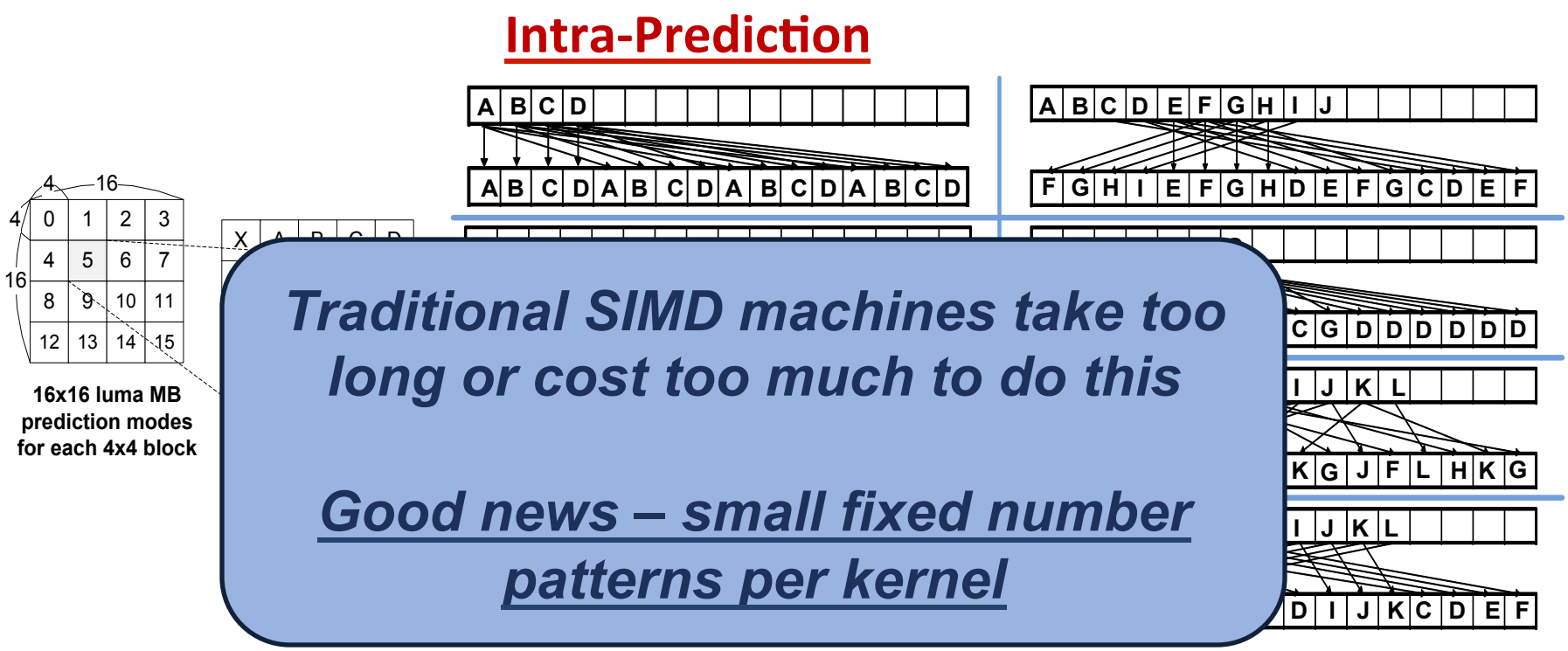
	Instruction Pair	Frequency
1	shuffle-shuffle	16.67%
2	add-multiply	16.67%
3	multiply-subtract	16.67%
4	multiply-add	16.67%
5	subtract-mult	16.67%
6	shuffle-add	16.67%

c) FFT

Like the Multiply-Accumulate (MAC) instruction there is opportunity to fuse other instructions

A few instruction pairs (3-5) make up the majority of all instruction pairs!

Data Alignment Problem!



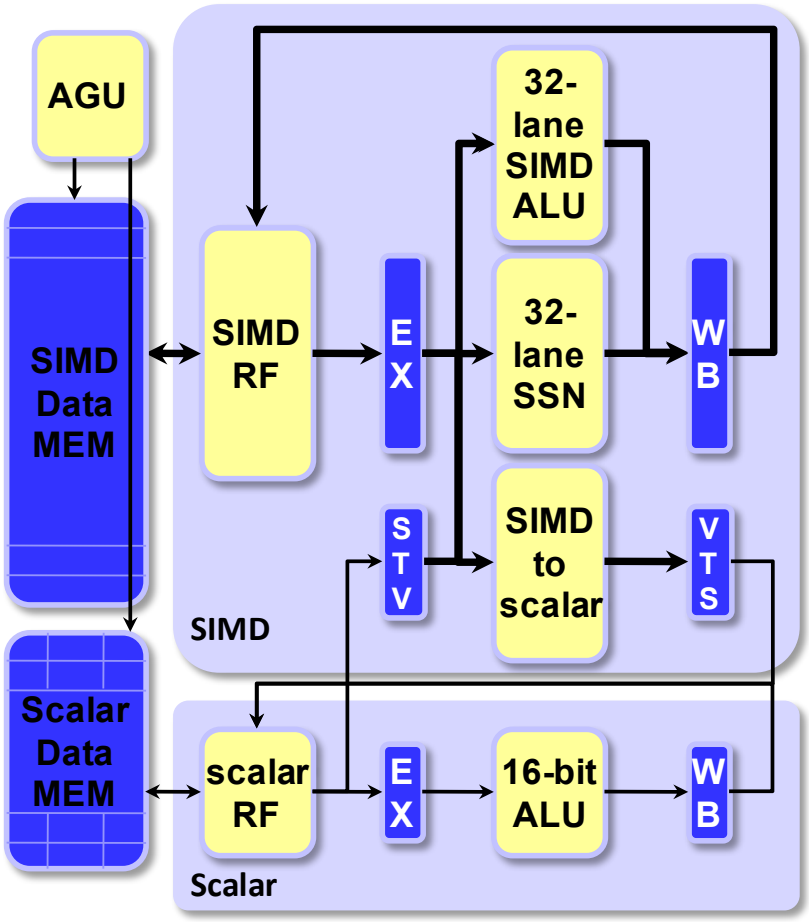
- H.264 Intra-prediction has 9 different prediction modes
  - Each prediction mode requires a specific permutation

Summary

- Conclusion about 4G and H.264
  - Lots of different sized parallelism
    - From 4 wide to 96 wide to 1024 wide SIMD
      - Which means many different SIMD widths need to be supported
  - Very short lived values
  - Lots of potential for instruction fusings
  - Limited set of shuffle patterns required for each kernel

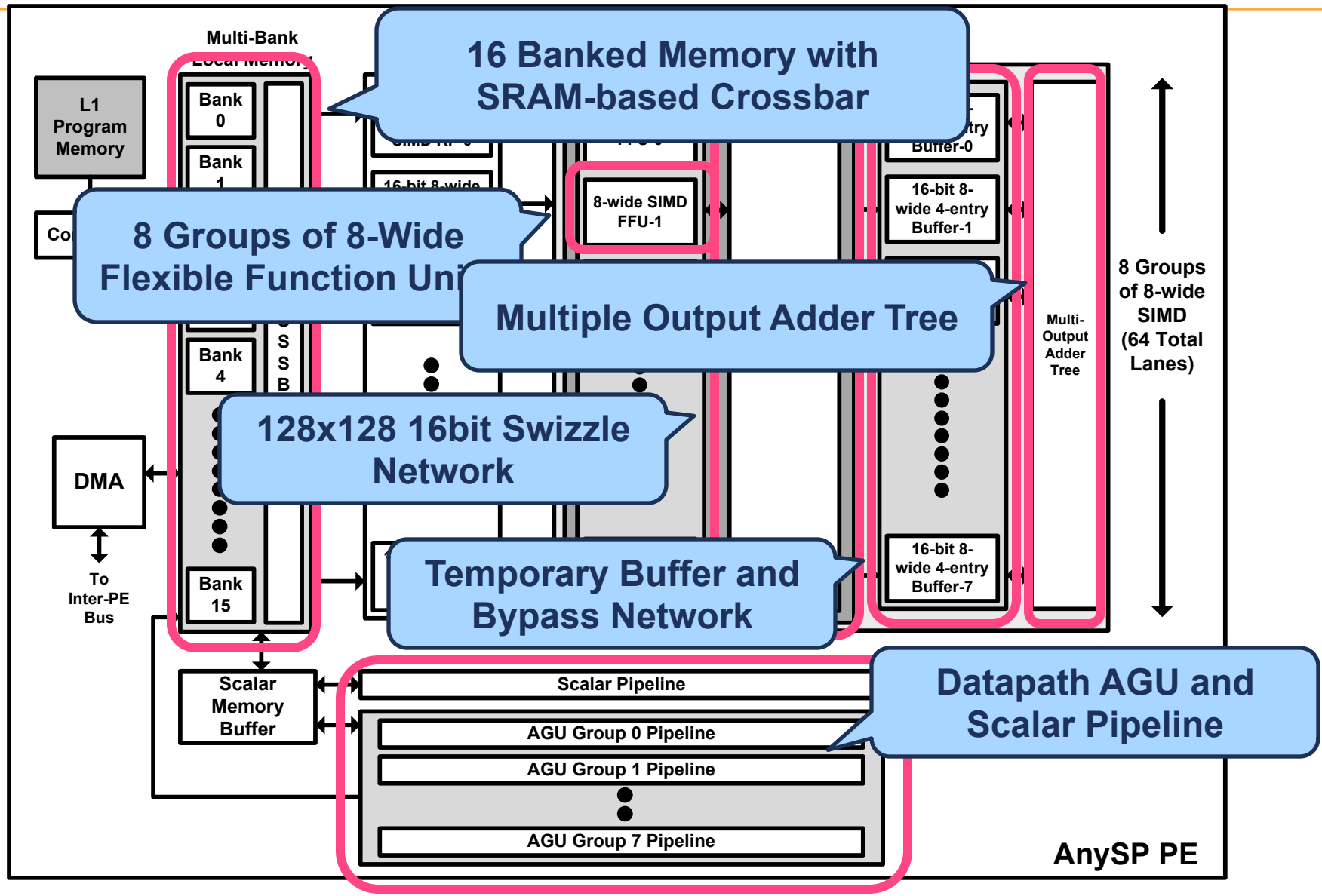
AnySP Design

Traditional SIMD Architectures

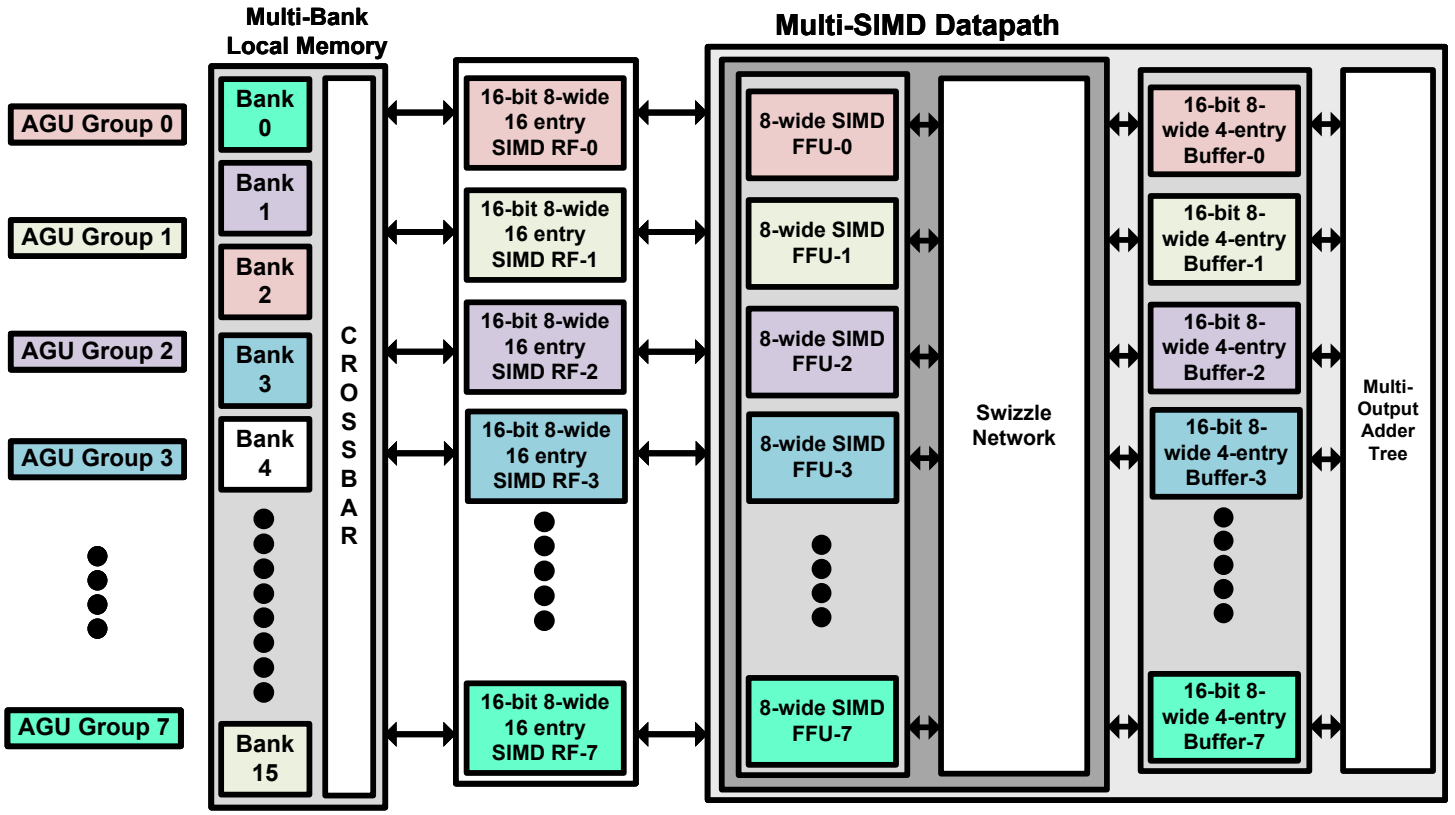


32-Wide SIMD with Simple Shuffle Network

AnySP Architecture – High Level

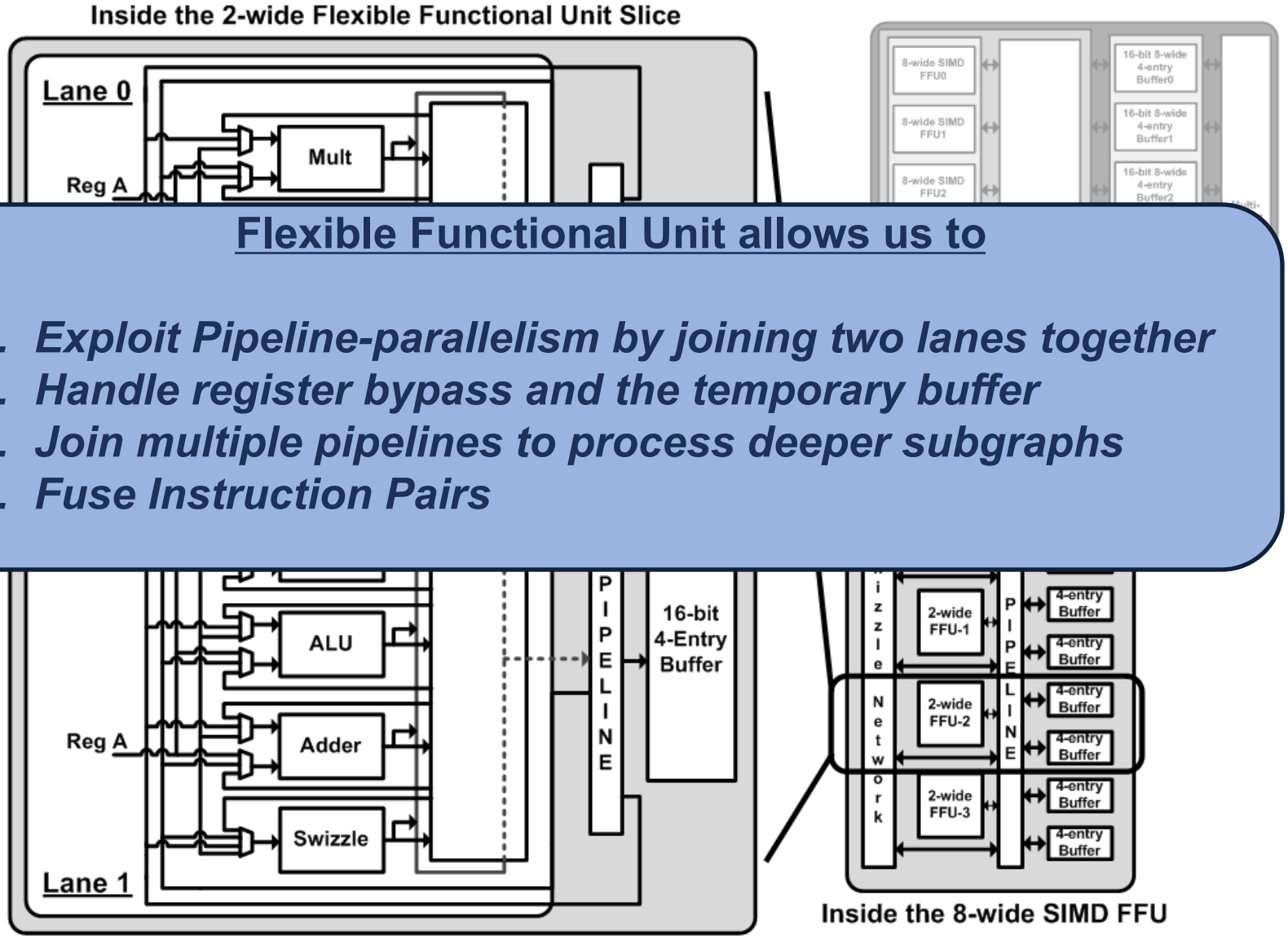


Multi-Width Support



Each 8-wide SIMD Group works on different memory locations of the same 8-wide code – AGU Offsets

AnySP FFU Datapath



Flexible Functional Unit allows us to

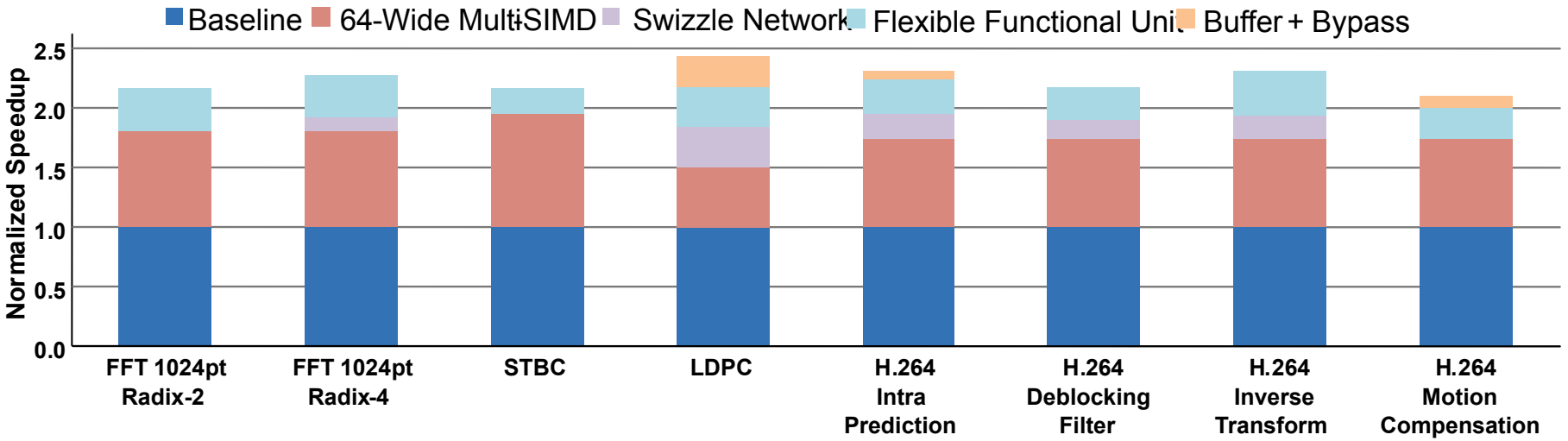
1. *Exploit Pipeline-parallelism by joining two lanes together*
2. *Handle register bypass and the temporary buffer*
3. *Join multiple pipelines to process deeper subgraphs*
4. *Fuse Instruction Pairs*

AnySP Results

Simulation Environment

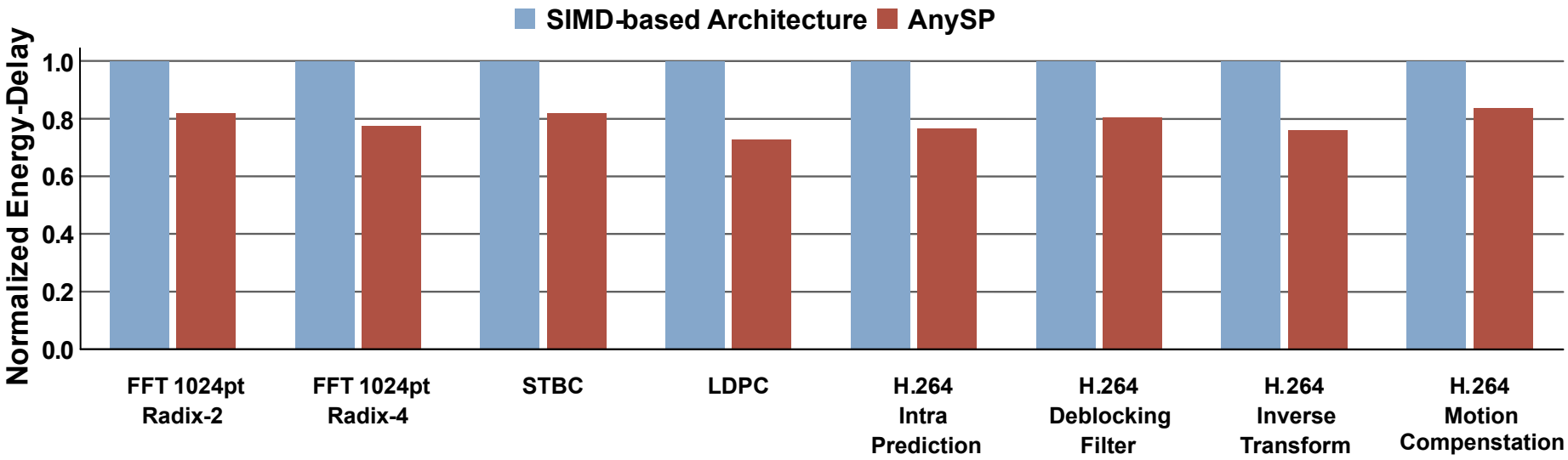
- Traditional SIMD architecture comparison
  - SODA at 90nm technology
- AnySP
  - Synthesized at 90nm TSMC
  - Power, timing, area numbers were extracted
- Performance and Power for each kernel was generated using synthesized data on in-house simulator
- 4G – based on a NTT DoCoMo 4G test setup
- H.264 – 4CIF@30fps

AnySP Speedup vs SIMD-based Architecture



- For all benchmarks we perform more than 2x better than a SIMD-based architecture

AnySP Energy-Delay vs SIMD-based Architecture



- More importantly *energy efficiency* is much better!

AnySP Power Breakdown

	Components	Units	Area		4G + H.264 Decoder	
			Area mm^2	Area %	Power mW	Power %
PE	SIMD Data Mem (32KB)	4	9.76	38.78%	102.88	7.24%
	SIMD Register File (16x1024bit)	4	3.17	12.59%	299.00	21.05%
	SIMD ALUs, Multipliers, and SSN	4	4.50	17.88%	448.51	31.58%
	SIMD Pipeline+Clock+Routing	4	1.18	4.69%	233.60	16.45%
	SIMD Buffer (128B)	4	0.82	3.25%	84.09	5.92%
	SIMD Adder Tree	4	0.18	<1%	10.43	<1%
	Intra-processor Interconnect	4	0.94	3.73%	93.44	6.58%
	Scalar/AGU Pipeline & Misc.	4	1.22	4.85%	134.32	9.46%
System	ARM (Cortex-M3)	1	0.6	2.38%	2.5	<1%
	Global Scratchpad Memory (128KB)	1	1.8	7.15%	10	<1%
	Inter-processor Bus with DMA	1	1.0	3.97%	1.5	<1%
Total	90nm (1V @300MHz)		**25.17**	100%	**1347.03**	100%
Est.	65nm (0.9V @ 300MHz)		**13.14**		**1091.09**	
	45nm (0.8V @ 300MHz)		**6.86**		**862.09**	

- We estimate that both H.264 and 4G wireless can be done in under 1 Watt at 45nm

Conclusion & Future Work

- Conclusion
  - We have presented an example architecture that could possibly meet the requirements of 100Mbps 4G and HD video on the same platform
    - Under the power budget and meeting the performance at 45nm
- Future and Ongoing Work
  - Application-specific language
  - Larger class of algorithms for AnySP
  - Better utilization of resources for non-parallel kernels
    - Speedup sequential parts