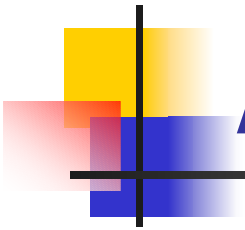


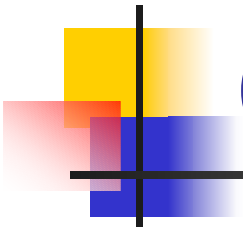
Aspect Ratio Project

James Bonanno

Suzanne Rivoire

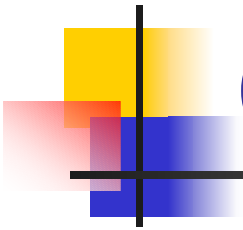
Rex Petersen

June 4, 2002



Overview

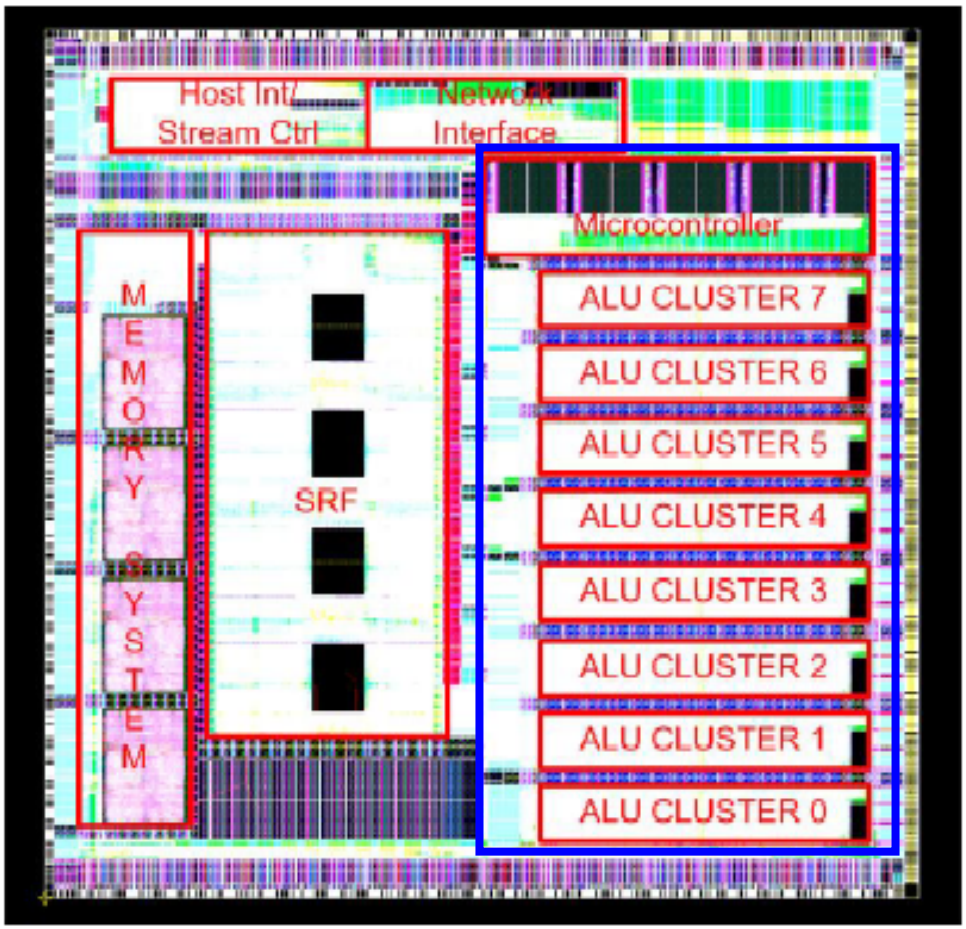
- How Cost Scales
- How Performance Scales
- How Performance/Cost Scales
- Recommendations for future work



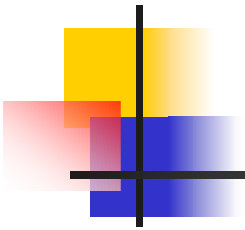
Cost Model

- Based on Area
  - *Included:* clusters, internal cluster switch, cluster communication, microcontroller
  - *Not included*: SRF, memory system, and other interfaces
- Units
  - Roughly in terms of mm^2
  - Scaled so that cost of Imagine is 100

Imagine



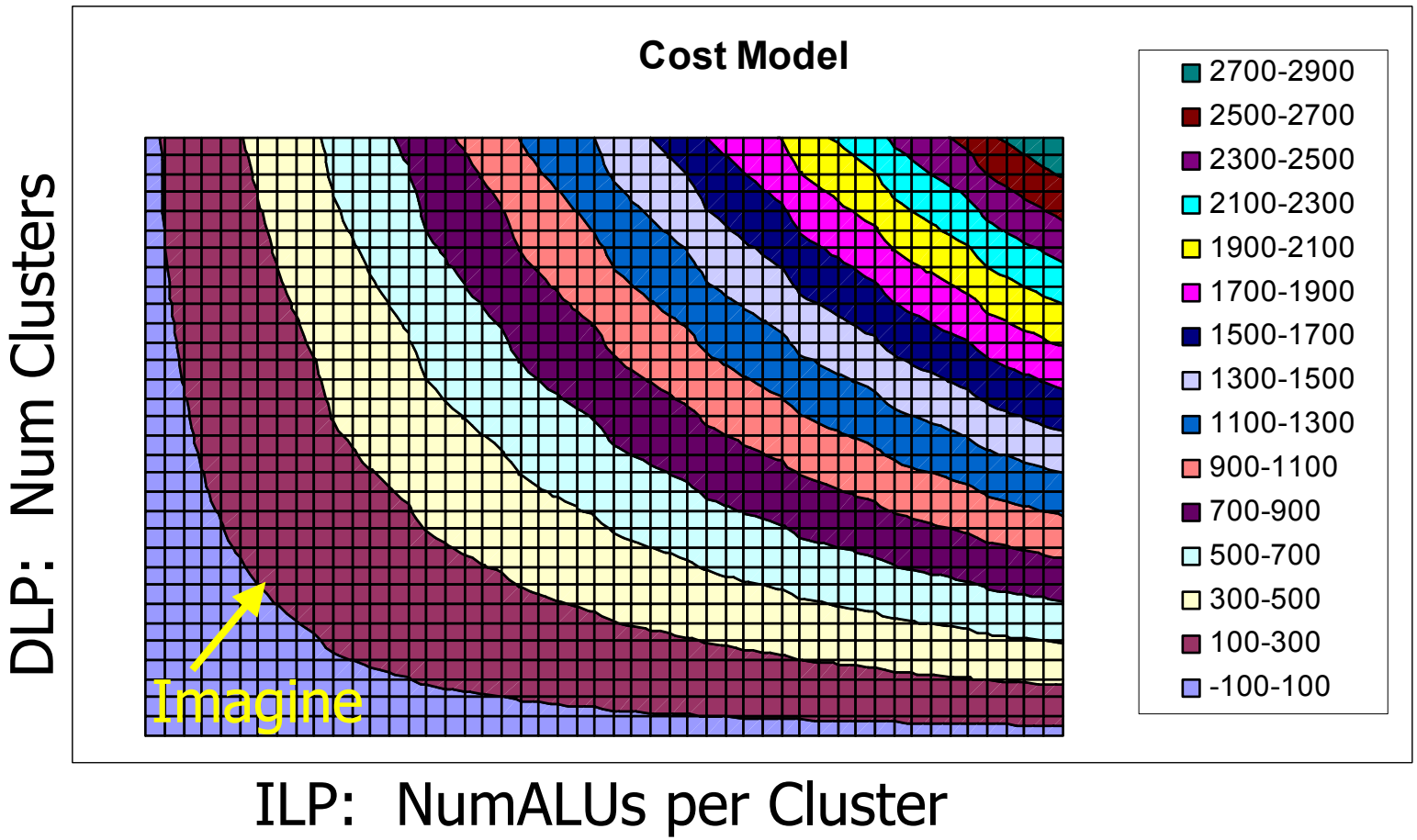
This is the area considered in the cost model



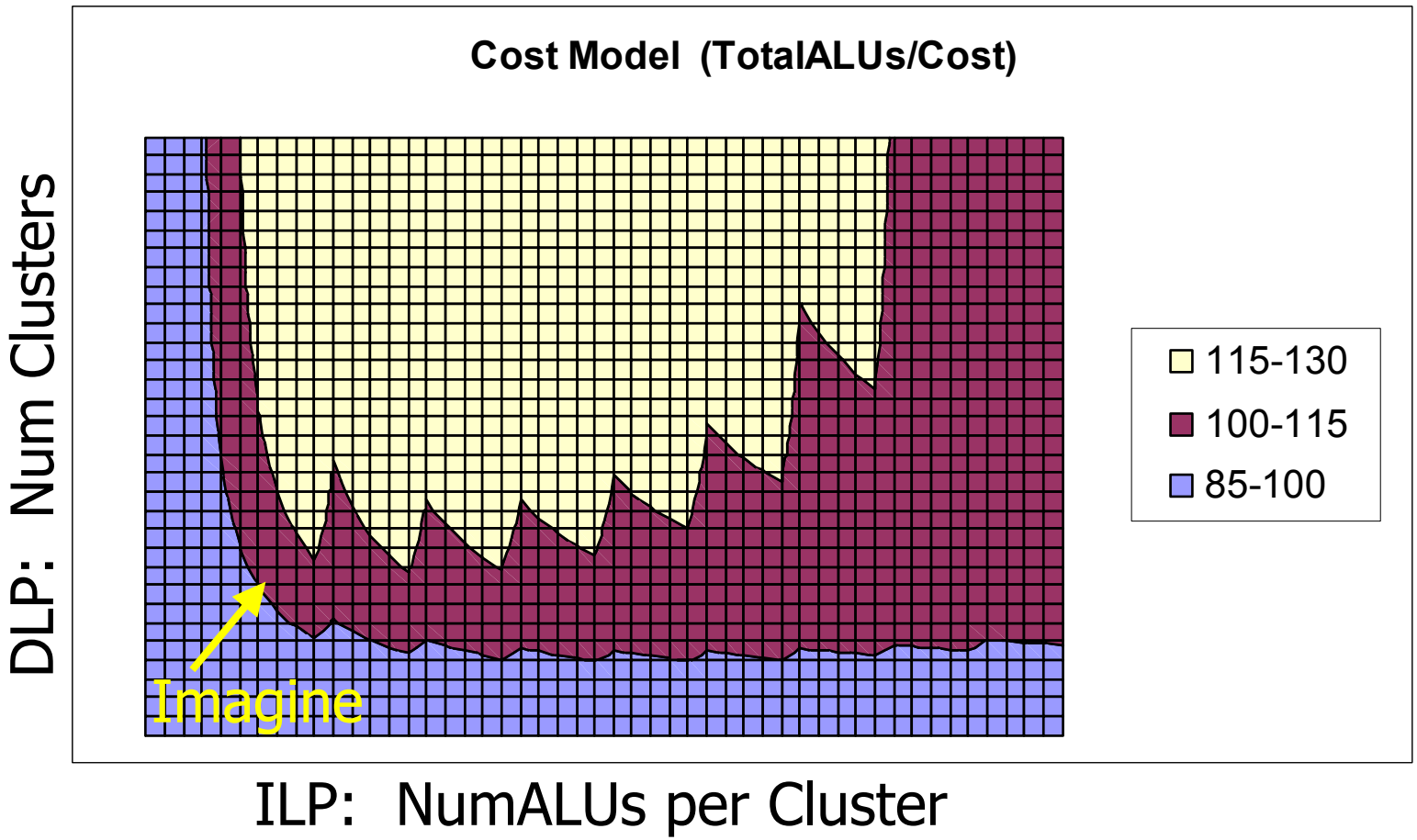
Basic Formula

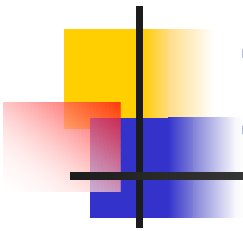
- Cost = NumThreads (NumClusters * ClusterArea + ClusterCommSwitchArea + microControllerArea)
  - ClusterArea = f(A_F, InternalClusterSwitchSize)
  - 1 ScratchPad and CommUnit per 5 other ALUs

Graph of Cost Model



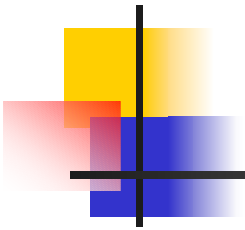
Graph of Total ALUs / Cost





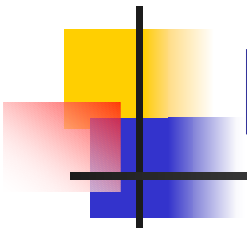
InternalClusterSwitchSize

- From *Register Organization for Media Processing*
  - SwitchSize = $(p_eN + 2N)(p_eN + N) * w^2 * b^2$
  - N = Number of functional units
  - p_e = ¼ Number of external ports per ALU
  - b = 32 Data width
  - w = wire pitch (typically 0.5 – 2um)
  - SwitchSize = $N^2/100$
  - Switch can overlap part of the ALUarea



MicroControllerArea

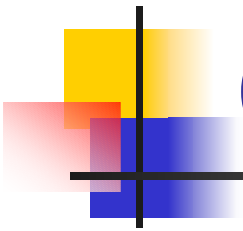
- MicroControllerArea = MemoryArea + DecoderAreaPerFuntionalUnit * NumFunctionalUnitsPerCluster
- How the MicroController Scales
  - DLP -- MicroControllerArea is constant
  - ILP -- As instructions get wider we need fewer of them. Memory area remains constant but the decoder area grows.
  - TLP – Each thread requires 1 MicroController



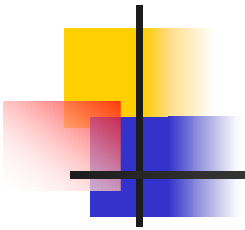
How Performance Scales

- ILP: our cluster configurations
- Kernel and overall performance
- ILP expansion and uc size
- Kernel classification
- We extrapolated to scale DLP, TLP

Our Cluster Configurations

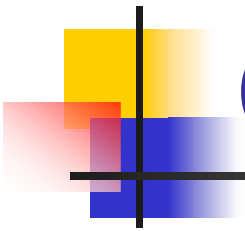


- Wimp: 1 ADD, 1 MUL, 1 DIV … 78
- Tin: 3 ADD, 1 MUL, 2 DIV … 100
- Gold: 3 ADD, 2 MUL, 1 DIV … 100
- Straw: 6 ADD, 4 MUL, 2 DIV … 181
- Stud: 12 ADD, 8 MUL, 4 DIV … 352



Kernel Performance

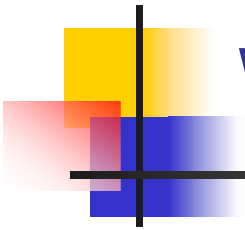
- Speedup of straw/gold = 1.75x, stud/gold = 2.88x
- Kernel performance/area: straw and gold fare the best



Overall Performance

- Include the StreamC overhead and things are very different…
- Stud and straw speedup now 1.2-1.3x
- Wimp is the performance/area champ
- StreamC pipelining would probably help

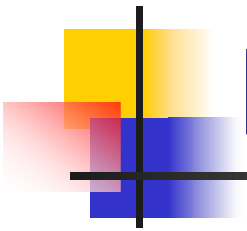
What is the cost of ILP?



- Is it uc size?

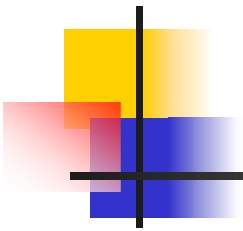
  Actually, if kernel is well-matched to HW and we don’t unroll excessively, NO.

- But the control part of the microcontroller (25% of area) does grow.



Kernel classification

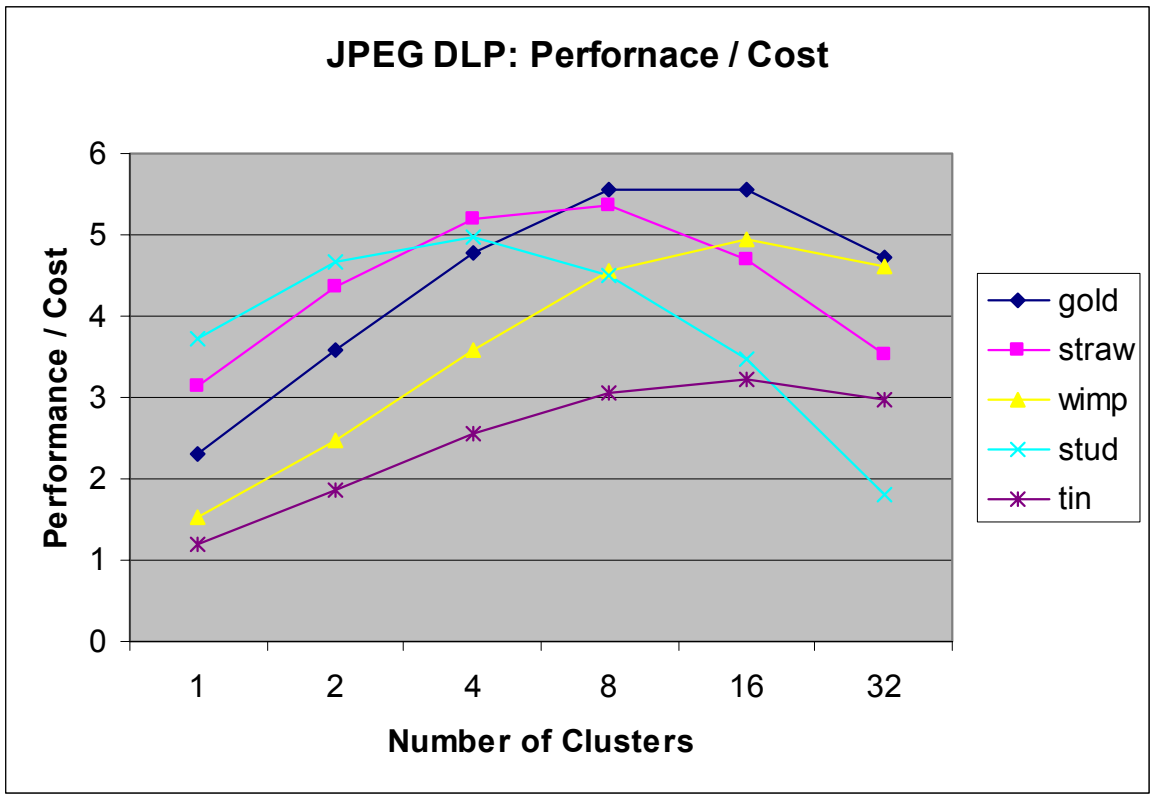
- In our code, we saw:
  - Wimpy kernels
  - FU-limited kernels
  - Zero-communication kernels
- In provided code, we saw:
  - Comm-limited kernels
  - Bigger kernels



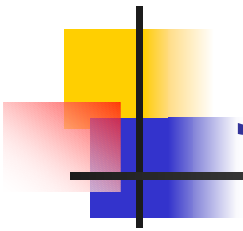
How Performance/Cost Scales

- Estimate for performance based on:
  - Scheduled kernels (uCode file)
  - Number of cycles per kernel invocation (num_loop_instr * num_iter) + num_non_loop_instr
  - Number of kernel invocations
- JPEG: extracted DLP and TLP performance
- Performance/Cost = K / (cycles * cost)

JPEG: DLP Results



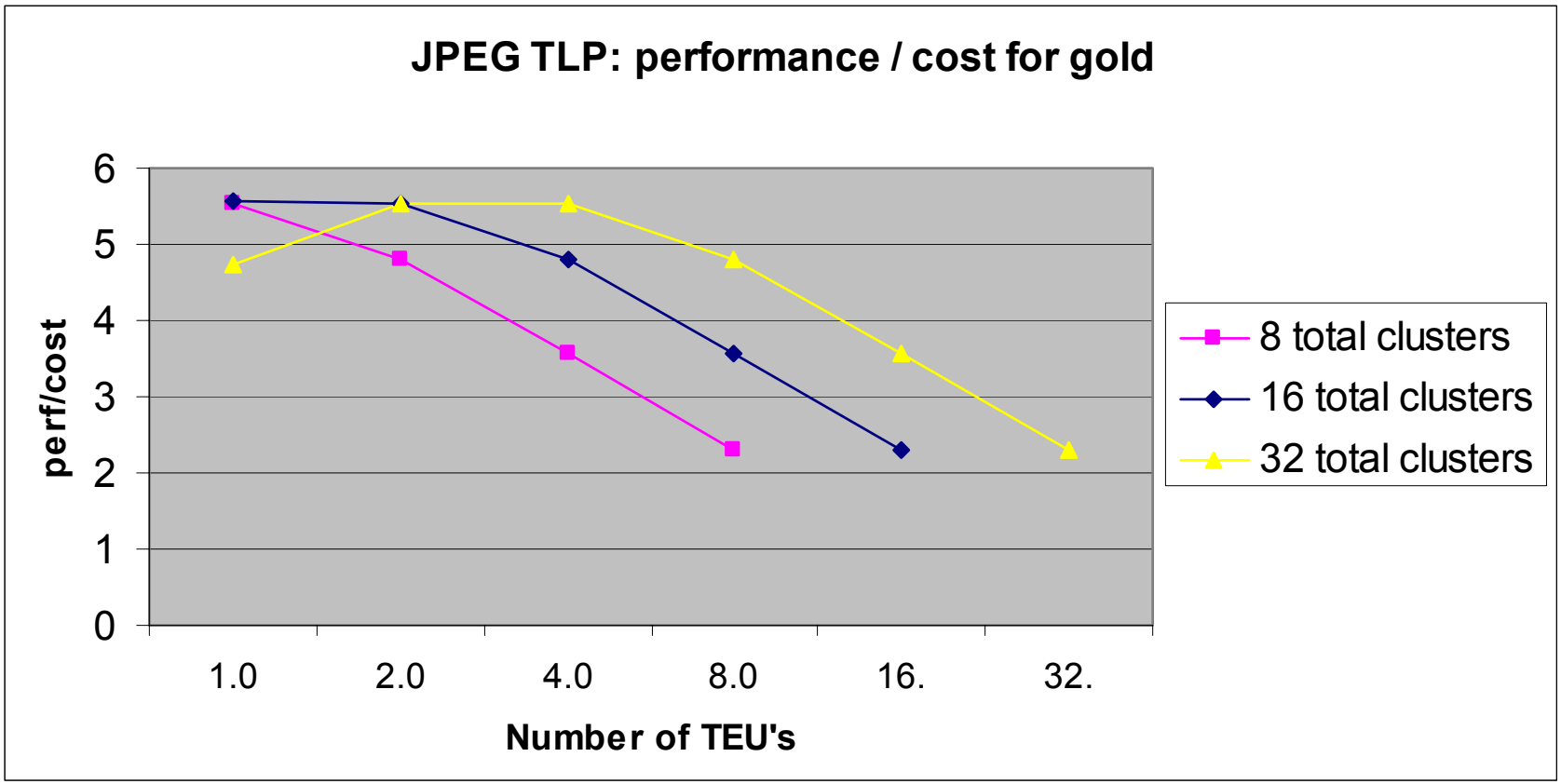
	A	M	D	C
Wimp8	1	1	1	70
Tin8	3	1	2	100
Gold8	3	2	1	100
Straw8	6	4	2	181
Stud8	12	8	4	352



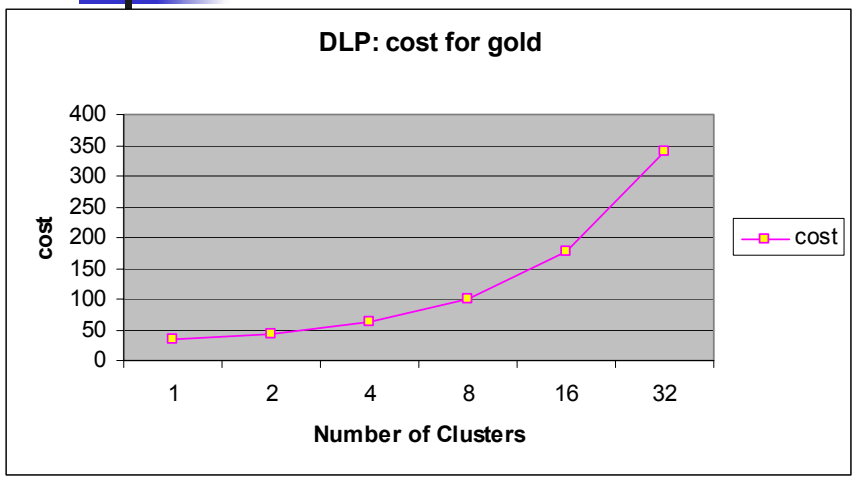
JPEG: TLP Results (1)

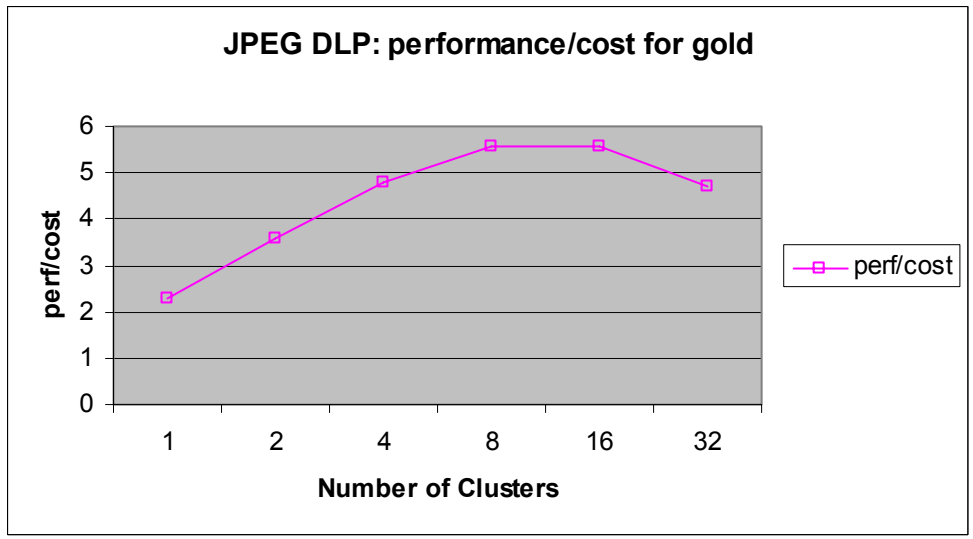
- Ran entire JPEG encode process on different 8x8 portions of the image
- Kept total number of clusters constant while adding TEU's
- Not useful to pipeline our implementation, since DCT dominates

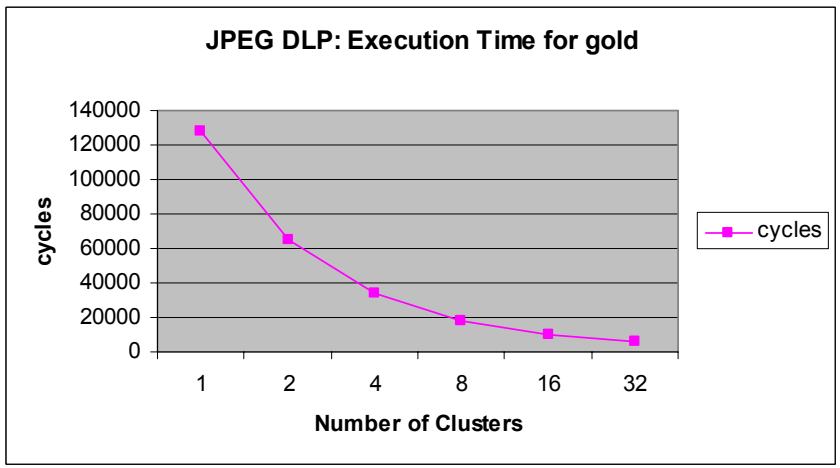
JPEG: TLP Results(2)

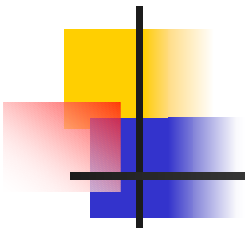


Cost vs. Number of Clusters





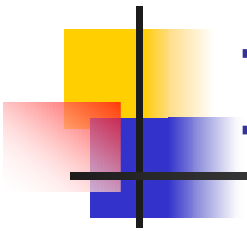




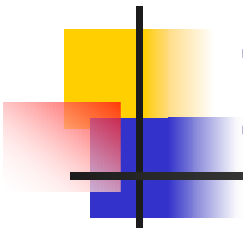
Recommendations

- Automate *.md file generation
- Have compiler convert DIV to MULT
- Continue kernel classification
- Explore TLP with different execution units
- Collect data on more applications and coding styles
- Scaling model for the SRF

ILP perf/area on JPEG



Wimp	1.14	.82
Tin	.86	.54
Gold	1.00	1.00
Straw	.72	.97
Stud	.38	.81
	(overall)	(kernel code)



ILP and code size

This kernel from mpeg is typical:

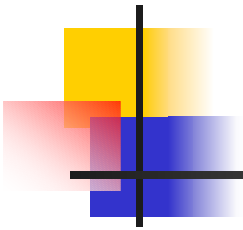
Name	*Cycles (p(1))*	*uc size*
Wimp	126	58125
Tin	52	37284
Gold	43	34790
Straw	22	33840
Stud	16	47418

Extracting cost/perf

config	
gold8	8
cost	100

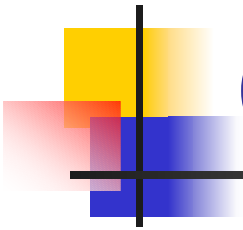
kernel	cycles/loop	nonloop cycles	unroll	loop count	# invoke	cycles/invoke	total cycles
gen_idx_str	8	5	4	2	4	21	84
dct	60	36	2	4	256	276	70656
quantize	7	5	1	8	4	61	244
rle1	19	6	1	8	4	158	632
rle2	14	5	1	8	4	117	468
							72084

	TEU0 - gold8		TEU1 - none		TEU2 - none		TEU3 - none				
kernel	**invocate**	**cycles**	**invocate**	**cycles**	**invocate**	**cycles**	**invocate**	**cycles**			
gen_idx	1	21	0	0	0	0	0	0			
dct	64	17664	0	0	0	0	0	0			
quantize	1	61	0	0	0	0	0	0			
rle1	1	158	0	0	0	0	0	0			
rle2	1	117	0	0	0	0	0	0	**cycles**	**cost**	**perf/cost**
		18021		0		0		0	18021	100	5.549081627



Estimating Wire Pitch

- 0.18um process
  - Metal pitch ranges from 0.64 – 1.6 um/wire
  - Assume some room for power & ground
  - Also some room to shield for noise
  - 2 um/wire seems reasonable
- Accounting for wire pitch
  - SwitchSize = $2.81N^2 * 2^2 * 32^2 / 10^6 = 0.012N^2$
  - This formula can be simplified to $N^2/100$



ClusterArea

- ALUarea and Switch can overlap
  - Introduced an overlap factor O_F
  - O_F indicates how wire limited the ALU area is and how much of the switch can overlap this area
  - O_F was set to 0.75
- ClusterArea
  - ClusterArea = $MAX(A_F, O_F A_{iCSw}) + (1-O_F)\, A_{iCSw}$
  - Note that A_F is larger than $O_F A_{iCSw}$ when $N_F < 100$