DEVELOPMENT NOTES FOR DELAY OPTIMIZATION

William Bricken
February 2004

CHANGING DESIGNERS

This is what will make a designer change methodology:

1. A solution that meets costs (size is a leading indicator) and speed specification, and

2. A solution that that meets cost with improved speed or a solution that meets speed with improved costs.

COMPARISON TYPES

Here's a comparison spreadsheet:

1. Cases when Synopsys meets both specs:

-- we can be smaller at the same speed, given that the speed is not at a limit (ie Synopsys' best)

-- we can be faster for the same area for those designs that our logic reduction is significantly better (eg CORDIC)

-- we do not know about relative performance in the middle ground

2. Cases where Synopsys fails to meet area:

-- we can meet area that is within 80% of Synopsys best, given some flexibility in speed (ie slower than the speed that Synopsys generates in failing the spec). I'm using "30%" as a nominal range of flexibility, but obviously the area/speed trade-off has a wider range.

3. Cases where Synopsys fails to meet speed:

-- we can meet speed that is within 80% of Synopsys best, given some flexibility in area (ie more area than Synopsys generates in failing the spec)

4. Cases where Synopsys fails to meet both speed and area:

-- for some designs we can meet both, particularly designs for which we have superior competitive logic reduction

In general, I believe we can provide a designer with more accurate and more flexible fine-tuning for design spec trade-offs, and we can reduce the number of iterations required to get to the desired configuration.

Synopsys requires a target speed and area, and then generates a result. If either the speed or that area result is insufficient, the designer iterates. ILOC is somewhat the same, except that we have a diversity of speed/area design data points. They vary widely depending upon ILOC optimization mode (for eg, see the data at top of page), and can be varied significantly by fine tuning ILOC parameters.

[These statements characterize what I am currently working on, not the eventual performance of a funded ILOC development (!).]

ISSUES WITH MAINTAINING DELAY PERFORMANCE

1) Abstractly you can flatten a design to have exceptional speed, BUT that requires specialized hardware (PLAs for eg), and it is impractical for most designs above say 20K gates. We're proving that for generic ASIC designs, we can improve speed beyond Synopsys best speed. More improvement requires more area. We are showing capability if the design is allowed to grow, and the competition cannot meet speed specs.

2) Pragmatically, designs with XOR networks (such as DMC54, comparators, and arithmetic) have asymptotic speed improvement. You cannot feasibly get below a lower delay limit, given any area growth. Roughly, ILOC can get 10% lower speed than Synopsys best speed with about 20-30% more area. For XOR designs, ILOC then requires a lot more area (up to 50% more area) for a small improvement, say 5%. Beyond that, no improvement is possible.

3) As any tight limit is approached, processing effort increases significantly, indeed exponentially. So we also need to address (later please ;-) how much time a particular result requires. I believe Synopsys has "effort" parameters, with the understanding that very-high-effort may return a result next year. ILOC is fully parallel in design, so that it can reach exceptional speeds with some up-front software and hardware effort. The native ILOC processing is also relatively fast (I'd guess we can reach 2-3x the competition), cause the internal boundary logic transformations are very efficient.

TIME-AREA TRADEOFF

I'm trying to find a measure for the relative desirability of the time/area trade-offs. At least with the ILOC algorithms, it turns out to be very important where you put the optimization bias. We now have some tools to do fine-grain tuning of ILOC performance trade-offs.

I'm seeing curves like this: [Caution: this ILOC data is not at all complete; for example, none of the low-delay areas have been area-optimized off the critical path -- they do not represent our capabilities. The relevant data is the slopes not the absolute values.]

	ps	u^2		
DMC54	DELAY	AREA	-slope	ps/u^2 (delta-DELAY/delta-AREA)
	10720	40283		
	9320	41382	1400/1099	1.27
	8507	43270	813/1888	.431
	8025	44873	482/1597	.302
	7185	47258	840/2385	.352
	6768	50516	417/3258	.128
	6179	55601	589/5085	.116
	...			
	4884	113036		
	4493	123116	391/10080	.039
	4127	129651	366/ 6535	.056
	4004	133186	123/ 3535	.035

The middle ellipsis is swapping over from an area minimization regime to a delay minimization regime.

The proportionate gains are not linear, but rather step-wise. For this design, the cost of better delay begins to accelerate at about 150% of Synopsys best delay performance, so that the area cost of moving past Synopsys' best is very high:

	Synopsys best	u^2 cost to gain 1 ps
we cannot do this -->	70%	1000s perhaps
	80%	40
ILOC improvement	90%	20
	100%	10
	110%	8
	120%	6
	130%	3
	140%	2

How much area is a designer willing to trade for a particular delay gain, and vice versa? Although "30% is a lot", it seems to be quite dependent on how tight the specs are relative to what is possible. Would twice as much area be OK if that were the only way timing could be met? -- OK pragmatically, the spec is a compromise and both area and delay are somewhat flexible. I have seen cases with FPGAs when the choice was between one or two chips, ie the design was 110% of a single chip capacity. In such cases, delay is very quickly compromised to bring area down to one chip!

The three points we have for Synopsys DMC54 are:

DELAY	AREA	-slope		
5718	59173			
5260	64700	458/5527	.083	ps/u^2
4612	94931	648/30231	.021	

That is, 50 u^2 to gain 1 ps is OK to Synopsys. This chart shows:

1) The beginning delay for low area is very good, indicating to me that area is not really a concern. Synopsys biases heavily toward good delay. We can reduce this design to 37K u^2 at slower speeds, most probably not a relevant improvement.

2) The curve is steep; lowering delay is quite expensive. I don't know if Synopsys limits its delay reduction to have reasonable area when one asks for "minimal delay regardless of area". Our claims to lower delay more than Synopsys depend on Synopsys actually giving its lowest possible delay.

3) Comparing slopes, we are also getting a better ratio of ps/u^2 as area increases and delay decreases.

We (hopefully) know that Synopsys cannot improve upon the 4600ps delay, that it can improve upon the 64K area by going to about 7000ps (ie 50% more delay), and that it cannot improve upon about 45K area at any speed. Which parts of these ranges are of competitive interest to BTC? I'd currently assume that exploring above 5000ps for smaller area is not of interest.

Synopsys area of 95K for 4.6ns is a non-comparable standard at 4.5ns. I do not find it sensible to ask for 95K at say 4.5ns. The question of "faster with less area" is only comparative within speeds that Synopsys can achieve; beyond those speeds, we have crossed a performance boundary, and the question becomes "how much do you want what the competition cannot provide?"

I've concentrated on delay regardless of area assuming that getting the best timing was significantly more important than keeping the area low. We have previously shown that regardless of delay, we can improve area by around 15-

20% on average. Getting both best area and best delay will certainly require a full code rewrite, since that would require making the best transformation decision under all circumstances for every transformation -- a big job.

DELAY-VS-AREA GRAPH

All of this centers around one idea: The achievable delay and area performances, considered together, cover a portion of a delay-vs-area graph. "Best delay for a constant area" is a line, while "best area for a constant delay" is a different line on that graph. There may be no point in the space representing best delay and best area concurrently.

A graph example is attached at the end of the memo.

In the graph, optimal area performance follows the lower portion (the part labeled BEST) of the possible results space. As delay gets lower, it becomes impossible to achieve low area. Similarly as area gets lower, low delay cannot be achieved. The point is that it may be impossible to write an algorithm that follows the BEST portion of the possible results space. On first iteration, results will fall somewhere in the middle, successive iterations might move the results more toward BEST, but very rarely will an algorithm move concurrently toward both best delay and best area. The net result is that you generally wind up with "pretty good area" for a given delay, or "pretty good delay" for a given area. And you have no way to improve one without degrading the other.

What spaces on the graph does our performance need to fall within? We could, for example, show Synopsys performance and ours on the same graph, with ours consistently showing either better area or better delay (but not necessarily both).

This graph shows that both systems do fairly well, but there are areas for which ILOC has a very large advantage, and areas for which Synopsys is better. Quality of performance depends on the specification, and "improving delay while holding area constant" is not a choice. Even knowing what the trade-off will be is not a choice, rather it is an experiment conducted over successive iterations.

Complexity

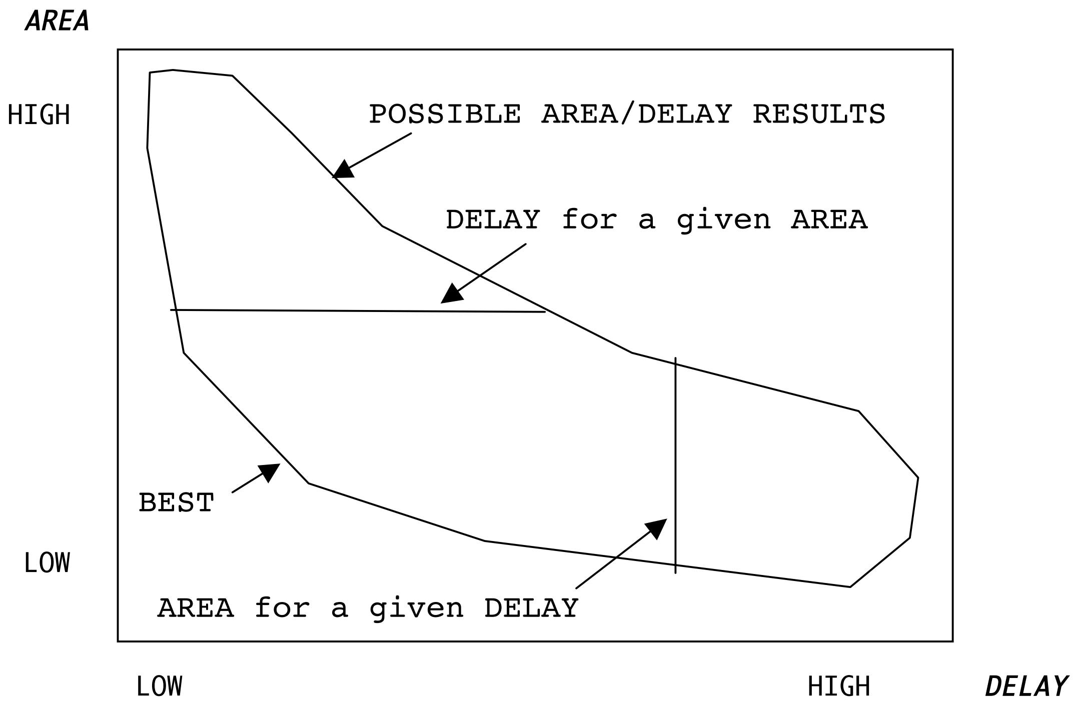
The actual situation is far more complex, since each of a dozen *algorithms* has a different performance curve for each of a dozen goal parameter settings, and performance trajectories too cover a space rather than stay on a simple curve. And the entire arrangement is different for every different functional design!

Each different *design* has a different delay-vs-area performance graph. I know of no accurate way to combine performance over all different types of designs, mainly cause the space of design architectures is exceedingly large and diverse. Worse, even the same functional design can have very different performance graphs, depending on cell-library, hardware architecture, and optimization preferences other than delay and area (egs: power, noise).

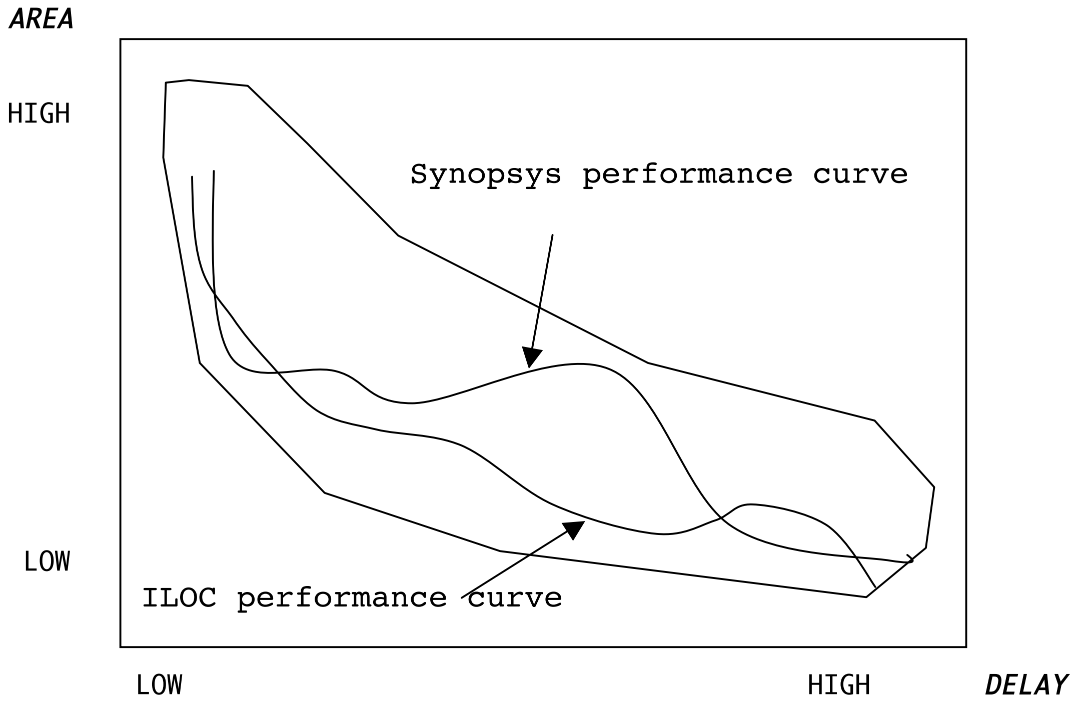
The other designs in our comparison suite demonstrate this: Relative to ILOC performance, Synopsys freezes on TOO_LARGE, does really poorly on CORDIC, and is about the same for TABLE5. These differences are due almost entirely to the type of functional structure of each design. That is to say, algorithms that perform well on look-up tables do poorly on arithmetic; those that do well on arithmetic are also good for ALUs but poor for glue logic, etc. Both Synopsys and ILOC use different algorithms for different types of functionality, so we are not even comparing algorithm performance -- we are in fact comparing good judgment in tool composition.

These observations sum to: *logic synthesis is complex*.

AREA vs DELAY PERFORMANCE



COMPARATIVE AREA vs DELAY


AREA
HIGH
LOW
Synopsys performance curve
ILOC performance curve
LOW
HIGH
DELAY