ADVANTAGES OF LOSP/PUN FOR EDA SOFTWARE COMPANIES

William Bricken

June 2001

This memo explores what we might include in a pitch to a substantive EDA software company such as Cadence or Synopsis.

As usual, this is a collection of small pieces, often pedagogical in tone, intended as an excess to be pruned, with many unfilled holes.

I construct short summaries after we filter this stuff.

Summary

Losp/Pun improves upon state-of-the-art logic synthesis. It will need to be rewritten and integrated into a CAE workbench. It also has many incipient capabilities which could extend its utility.

CONTENTS *ANNOTATED*

WHAT LOSP/PUN OFFERS

"Research in logic synthesis is a very satisfying enterprise because improvements in algorithms can immediately translate into smaller or faster circuits, and smaller and faster circuits have substantial commercial impact." [Devadas, *Logic Synthesis*, p.xiv]

"The need of practical synthesis and optimization algorithms for multiple-level circuits has made this topic one of utmost importance in CAD." [DeMicheli, *Synthesis and Optimization of Digital Circuits*, p.344]

Our advantage is simple: we have a better way of working with logic. This advantage applies to many aspects of chip development, and is focused in the sub-field of logic synthesis. Fortunately, CAD software is fragmented over dozens of products (even within Cadence and Synopsis); Losp/Pun can improve several different components of the EDA design flow by providing transformational tools which are strictly more powerful, more efficient, and more coherent. A dominating factor in the semiconductor industry is time to market. Here Losp/Pun provides a tremendous advantage.

The essential point for BTC is that Boundary Logic Technology (BLT:-) can substantively improve the back-end performance of most logic synthesis tools, however the BLT component is about 5% of a marketable EDA software product. Thus we must either license BLT software or spend considerable resources rebuilding (or buying) CAD interaction tools and infrastructure.

BLT will be quite difficult to integrate into existing EDA backends and algorithms, since the supporting data structures are very different. Most likely, BLT would be used as a stand-alone process (perhaps post-process) aiding design. It could be used as a market differentiator for the right CAD company. When deeply integrated into a CAE tool, Losp/Pun would provide a completely new approach to designing circuits, an automated exploration (a *bike*) rather than a meticulous and arduous construction (a *hike*).

Logic Synthesis

Logic synthesis is a technical, highly mathematical field. Since design software is fragmented across the design flow, design changes from logic synthesis are often lost as the chip moves from mathematics to physicality. As well, logic synthesis depends on simulation models which cannot express the physical complexity encountered in complex submicron devices.

Our unique capability is *Boolean minimization*, being able to algorithmically identify designs which meet specified criteria. Minimization is a substantively more difficult problem that verification, which is certifying that one design is equivalent to another design.

The data structure underneath almost all advanced logic synthesis tools is the Binary Decision Diagram (BDDs and variants Ordered OBBDs, Free FBDDs, Zero-suppressed ZOBBDs, Multi-terminal MOBDDs, Edge-valued EVBDDs, and many others). Historically, the first Boolean CAD tools, based on flattened logic networks, exploded exponentially in size as inputs increased. Since the early 1990s, BDDs have gained dominance as a better and more efficient tool. They too have a problem with exponential complexity, but it is managed heuristically by BDD variants. They also have a fatal flaw: BDDs do not map directly onto circuit structures, stripping a designer of design visualization and enforcing blind algorithmic verification. Worse, BDDs are not algebraic, making manipulation and transformation awkward at best.

Losp/Pun is fully Boolean, using a representation which can be directly read as a circuit (i.e. distinction nets) and directly manipulated by Boolean algebra. This alone is a substantial improvement, in addition Losp/Pun uses an iconic Boolean algebra which is technically more efficient than other methods. We thus offer both new capabilities and new efficiencies in achieving those capabilities.

Integration of an iconic algebra into a BDD-based system creates deep problems. At best we would be able to exchange BL algorithms in total for specific BDD techniques. This would not achieve an integrated design tool, but would provide local improvements. As a stand-alone tool, Losp/Pun would likely outperform an entire suite of commercial logic synthesis tools, again creating both transformation strengths and integration weaknesses.

Non-synthesis Software Tools

Logic synthesis tools constitute a small fraction of the code in an EDA system. The bulk of a synthesis tool, like any software tool, is at the interface. This includes

- *Schematic Capture*: translating circuit diagrams into part and configuration information

- *Hardware Description Language* (HDL): a programming language for hardware design, incorporating logic parallelism and timing.

- *Timing Specification and Display*: models of temporal circuit activity, including potential timing conflicts.

- *Component Models and Design Rules*: Abstract logical devices and assembly constraints.

- *Library Tools*: Management of libraries of components.

These can be considered primarily to aid specification. Once roughed out, design changes are ruled by logic synthesis tools.

After synthesis, another set of physical design tools (PCB design) moves the abstract model into physical reality, addressing concerns such as

- mechanical constraints
- critical signal paths and their floorplans
- heat dissipation and thermal effects
- radio-frequency emissions
- transmission line (wiring) distortion and other effects
- physical testability
- manufacturing constraints and faults.

Specification, synthesis, and physical design software is not integrated, and at times not even coherent. Currently Losp/Pun is limited to synthesis; in the immediate future BTC should avoid building new interface and physical design tools. We are thus quite dependent on a partnership with existing EDA tool companies.

A dominant design concern is chip complexity, which is approaching ten million transistors packed into the size of a postage stamp. Modular design is mandatory, so are scalable tools. Testing physical chips for physical defects is critical, especially since smaller sizes imply higher defect rates. As complexity increases linearly, test vectors increase at least quadratically. Currently, massive dedicated computers provide test vector verification in overnight batches. Increasing testing time from hours to days is not feasible. Currently there is little Automated Test Pattern Generation (ATPG) software in Losp/Pun; this would be simple to add, thus increasing the range of our applicability.

The current versions of Losp/Pun do not scale, since they were developed as research tools, heavily embedded with both correctness tests and design options. This is the primary reason for a code rewrite.

More About Logic Synthesis

Within logic synthesis, there are dozens of mathematical transformations, tools and techniques. An important initial distinction is between combinational logic and sequential timed logic. Most circuitry consists of logic blocks separated by register banks. Logic controls the computed functionality; registers control the timing and reuse of logic components.

Fundamentally, timing analysis is outside of the scope of Losp. Pun supports registers, timing models, and register reconfiguration, but at this time only weakly. Since timing is considered to be 90% of the difficulty of design, this may appear to be a limitation. In a traditional design flow, it is.

Within the Losp/Pun model, however, timing issues too are simplified. Thus Pun is particularly strong at producing timed and pipelined circuits, since it has a very simple, single type (the distinction, or enclosure) internal model. The weakness is an incompatibility with current timing design approaches, which results in a disconnection with established circuit library components.

Interestingly, timing tools break a design into simple components: registers, and the combinational logic between them. Timing is largely the iterative application of combinational logic synthesis.

Synthesis of combinational logic is the strongest aspect of Losp/Pun. Logic minimization is multifaceted, it is a search through a design space with dozens of parameters to consider. Pun provides automated design control of several common parameters, including

- functional behavior (usually held invariant)
- count of logic gates and transistors
- number, length, and layout of wires
- signal transversal time (critical path)
- pipelining and logic chunking
- fanin and fanout
- power consumption [Pun has weak tools currently]
- technology mapping (choosing library components)

The primary tool for integrating these diverse objectives has been designer expertise. Most synthesis tools provide shallow control of design decisions and verification of functional invariance. None that I know of provide a parameterized space which is explored algorithmically. In fact, designers tends to believe this is not even possible, since their judgments are subtle. Losp/Pun can provide within minutes a circuit which is close to ideal, but not the exact best (a *satisficing solution*). With an interface, it could be used to explore potential designs rapidly, and to provide basis designs which can be possibly improved by designer expertise.

The following technical areas identify the research frontier of CAD algorithms for combinational logic:

- equivalence checking, logic verification, tautology proving
- redundancy removal and minimization to specified criteria
- Boolean factoring and constraint reasoning
- network decomposition and transformation
- test vector generation, BDD generation and technology mapping.

Losp excels in the first three, Pun is very good at the last two. A catch is that designers are not accustomed to having or using tools which have the capability of addressing the above frontiers.

CIRCUIT DESIGN AND MANUFACTURING

VLSI circuits are perhaps the most complex artifact made by man. Their design and manufacture requires an extensive range of skills and processes. Thus, market leaders in Electronic Design Automation (EDA) tools have dozens of products, each applying to a different aspect of the design task.

A central distinction is between CAD tools, which are used for the physical design of circuit boards, and CAE tools, used for conceptual and mathematical chip functionality design.

Chip Architectures

Chip architectures differentiate types of computation.

- *Application Specific Integrated Circuits* (ASICs) are the most customized, specific hardware to do a specific task. They are justified only for large volume, single purpose applications.

- *Digital Signal Processors* (DSPs) are specialized for pattern recognition tasks.

- *Field Programmable Gate Arrays* (FPGAs, also CLBs, PLDs, CPLDs) are reconfigurable, used for prototyping and in circumstances where hardware functionality must be frequently updated.

- *Random Access Memories* (RAMs, in Static SRAM and Dynamic DRAM varieties) are specialized for storing and retrieving digital data.

- *Printed Circuit Boards* (PCBs) combine chips, data busses, memories, input/output connections, and other components.

- *System on a Chip* (SoCs) reduces a PCB to an ASIC.

Losp/Pun is generally transparent to chip architecture; at this time the algorithms do not distinguish between types of netlist. By algorithmically making an initial distinction between netlist structures, the Losp algorithms can be targeted for efficiency but not improved in optimization performance.

Huge architectures, such as SoCs, must be partitioned. Since physical architectures provide a natural partition, algorithms must address both predefined and discovered hierarchical partitions (or abstractions). Losp is algebraic, internal variables can represent arbitrary configurations and subconfigurations, providing a direct mathematical tool for hierarchical analysis and pattern abstraction. One innovative potential is that Losp can identify common patterns across families of circuit types, building library elements that are hierarchical, generic and efficient.

Design Objectives

Design objectives vary widely. The mutually dependent factors listed below are annotated by Losp/Pun capabilities.

- *cost*
  controlled by yield and manufacturing overhead (not in Losp)
  Losp optimization provides a constant cost improvement
  time to market is greatly improved by Losp automation

- number of gates and transistors
  logic minimization is over a single gate type (generalized NOR)
  library mapping is particularly easy
  transistor designs and mappings supercede logic concepts
  Losp could provide excellent transistor level manipulation

- size of market
  market analysis not done

- general purpose, dedicated, or reconfigurable
  Losp/Pun flexibility is excellent for each
  Occlusion Array architecture very reconfigurable

- speed
  closely tied to physical prototypes
  Pun has good tools for critical path control and pipelining

- power efficiency, heat dissipation
  closely tied to transistor minimization
  no models currently in Pun

- clock rate and timing
  Pun abstracts timing, potentially new (untried) retiming tools
  Losp transformations fully integrated with registers
  single clock is assumed (needs generalization)

- ease of manufacturing (fault tolerance, high yield)
  appropriate for BL hardware designs
  Pun may provide an advantage, unknown for now

- ease of testing
  Pun flexibility contributes greatly
  good test vector generation tools
  excellent formal verification tools
  generates testable hardware architectures

- ease of design
  given a good interface, Losp/Pun is superb
  requires new, higher-level design skills
  also excellent for easing burden using old design skills

- chip-level integration, semiconductor technology, feature size
  unexplored

- IP protection
  largely unexplored, perhaps some excellent tools

Design Tasks

The main VLSI design tasks (the design flow) are

- *Specification*: Not a direct aspect of EDA tools, specification is the expression of desired functionality and manufacturing constraints in an approximate mathematical language which conveys management intent to engineers.

- *Design Entry*: Creating a circuit design from a specification.
  transcription of the formal specification into Pun is sufficient

- *Functional Verification*: Certifying that the entered design maintains fidelity with the specification.
  Losp is excellent
  Boolean minimizer is a new tool

- *Synthesis*: Rebuilding the verified design to meet performance goals.
  All automated and parameterized in Pun

- *Physical Design*: Rebuilding the synthesized design for physical embodiment.
  Nothing in Losp/Pun

- *Physical Verification*: Certifying that the physical design maintains fidelity with the functional and performance constraints.
  Nothing in Losp/Pun

- *Testing*: Certifying that the manufactured chip maintains fidelity to the specification.
  Excellent control of design for testability
  Excellent test vector generator and evaluator
  No physical models.

The software tools for circuit design include:

- *Schematic Entry*: Pun needs all of these.

- *Hardware Description Languages* (HDLs): Translators from VHDL, Verilog and others are not written, although 90% of each is easy.

- *HDL compilers, simulators, and synthesis tools*: None written. Pun could be seen as a compiler and simulator of BL circuits. There's a strong mapping between BL and conventional models

- *Simulators*: Within the BL model, excellent.

- *Programmable Logic* (PLDs): Hardware architectures based on Occlusion

- *Test benches*: No integrated interface, good software tools. No hardware tools.

- *Timing analyzers and verifiers*: None.

PRODUCT CATEGORIES

Synopsis partitions their product world into functions (design methodologies) and behaviors (design tasks). Methodologies organize product groups, tasks organize market segments. Losp/Pun contribution and advantage is measured in stars.

Synopsis Design Methodologies	***Losp/Pun Advantage***
design reuse	*
FPGA synthesis	**
functional verification	***
internet design environment	
logic synthesis	***
physical synthesis	
static verification	*
system level design	
test automation	**

Losp/Pun advantage includes potential tools (marked with ~) as well as implemented tools.

Cadence differs in focus from Synopsis, concentrating more on PCB than functional tools.

Synopsis Design Tasks

automated test pattern generation	**~
behavioral synthesis	*
datapath synthesis	
design for test	**
design optimization	***
design planning	
designWare	
DSP	*~
dynamic gate-level power analysis	*~
dynamic timing analyzer	
formal verification	***
functional verification	***
FPGA solutions	*
hardware IP characterization	
hw/sw co-design and co-verification	
IP modeling	
library development	**
links to layout	**~
logic synthesis	***
mixed HDL simulation	
model directory	
nanometer IC design	
physical verification	
power management and diagnostics	*~
power solutions	
RC extraction	
reliability analysis	**~
simulation	**~
simulation models	*~
static timing analysis	**~
test	***~
testbench automation	
timing verification	
top-level routing	

Cadence Functional Partitions

schematic capture	
A/D mixed signal verification	
digital verification	***
high-speed PCB design	
PCB layout	
PCB routing	
IC packaging	
FPGA design	*~
component information management	

LOSP/PUN FUNCTIONAL CAPABILITIES

The Losp/Pun EDA tools are a collection of hundreds of boundary logic algorithms. The Losp suite performs optimization of boundary logic forms. The Pun suite applies Losp to the optimization of circuitry.

Losp/Pun can efficiently generate functionally invariant circuits across a wide diversity of specified configurations and chip architectures. It's primary function is circuit reconfiguration to meet specified parameters. Since these tools are based in a formal system, the correctness of circuit transformations can be verified.

Distinction networks provide an extremely flexible model which underlies all other compositional models (transistors, gates, functional blocks, architectures). The fine grain-size makes it inappropriate for PCB modeling.

Below, significant competitive advantage is starred. Some items are duplicates across lists.

General Boolean Capabilities

- evaluate logic expressions in given binding environment
- minimize logic expressions algebraically ***
- identify tautologies **
- determine satisfiability and generate counterexamples **
- perform partial function evaluation
- minimize variable references ***
- identify abstract symmetries **
- identify sequential and parallel components of computation ***
- standardize logic representations
- verify equivalence of structurally different forms
- Boolean factoring **
- Boolean pattern abstraction *
- Boolean constraint reasoning *
- case analysis

Boolean Tools Applied to Circuit Optimization

- network decomposition and transformation **
- redundancy removal and minimization ***
- don't care analysis and minimization ***
- critical path modification **
- specification of maximum fanin and fanout
- structure abstraction **
- structure sharing *
- test vector generation and BDD generation *
- technology and library mapping *
- asynchronous, object-oriented graph-reduction model **

Pun Circuit Reconfiguration Capabilities

Once a locally minimal logical form of a circuit is generated by Losp reduction, the topological form can be manipulated to achieve specified design constraints.

- deepening (distribution) **
- structure sharing (coalesce) *
- trading-off connectivity for depth (factorization) *
- matching to structural templates (technology mapping) **
- retiming (register relocation)

Controllable Design Parameters

- low-redundancy (literals, forms, distribution of forms) **
- CNF/DNF, SOP/POS
- Implicate Normal Form *
- generalized nor
- four-input look-up tables *
- specific gate sets:
  {nor,not},{and,or,not},{and,or,not,mux,xor}
- specified fanin and fanout
- structure sharing (increased fanout, decreased area) *
- maximal structure sharing (disassemble and reconstruct) **
- depth/wiring tradeoffs (critical path reduction) **
- low-level abstraction (xor and if-then-else/mux units)
- high-level structural abstraction
  (specific to functionality) *
- specific structural partitioning
  (designated library modules) *
- parens form (no partitioning)
- occlusion arrays ***
- partial function evaluation (some bound inputs) *

EDA Specific Capabilities

• parse and return EDIF circuit descriptions	
• optimize combinational and sequential multilevel circuits	***
• detect false paths and minimize reconvergent fanout	*
• remove don’t care conditions	**
• reduce delay along critical path	*
• reduce gate count or area	**
• exchange timing for wiring	*
• optimize structure sharing in multilevel circuits	*
• specify fanin and fanout limits	
• identify equivalent circuits and pin-point differences	
• generate circuits to meet multiple design objectives	***
• identify patterns and hierarchical decompositions	**
• generate test vectors	**
• identify abstract patterns, construct library elements	*
• technology mapping for xor, mux, n-LUT, others	*
• simulate functionality with bit-streams	***