Technology library modeling for information-driven circuit synthesis

Technology Library Modelling for Information-driven Circuit Synthesis

Lech Jóźwiak and Szymon Biegański
Eindhoven University of Technology
L.Jozwiak@tue.nl

Abstract

Due to weaknesses in circuit synthesis methods used in today's CAD tools, the opportunities created by modern microelectronic technology cannot effectively be exploited. This paper considers major issues and requirements of circuit synthesis for the nano CMOS technologies, and discusses our new information-driven circuit synthesis technology that satisfies these requirements. It focuses on an adequate technology library modelling for information-driven circuit synthesis. The new circuit synthesis technology considerably differs from all other known synthesis methods and overcomes their main weaknesses. The experimental results demonstrate that it is able to produce very fast, compact and low-power circuits.

1. Introduction

This paper addresses the problem of an adequate synthesis of digital circuits for the modern nano CMOS circuit implementation technologies. It briefly considers some major issues and requirements of circuit synthesis for the nano CMOS technologies, and discusses our new information-driven circuit synthesis technology that satisfies these requirements. It focuses on the issue of an adequate technology library modelling for the purpose of information-driven circuit synthesis.

Introduction of the nano CMOS technologies created new opportunities, as well as unusual complexity, and particularly: extremely high device and interconnect densities, extremely small devices' dimensions, and huge length of interconnects. Due to this complexity, interconnect scalability problems, power supply reduction and very high operating frequencies, many previously ignorable phenomena have now a great impact on the circuit correctness and other quality aspects. This results in many new difficult to solve issues, including: power and energy crisis, increased leakage power, interconnect scalability problems and dominating influence of interconnects on major physical circuit characteristics (e.g. area, speed, …), etc. [8]. Unfortunately, the available circuit synthesis methods and tools do not well address the needs of circuit synthesis for the modern technologies, due to: not accounting for the recently changed importance relationships among various circuit characteristics, not explicitly account for timing and power and using some proxy attributes for area that often do not well correlate with the actual area, not applying the now necessary multi-objective circuit optimization and trade-off exploitation, and being not effective for many classes of circuits due to making many prior assumptions excluding many possible circuit structures [1][2][5].

In result, the proxy synthesis targets of the available logic synthesis methods and tools very much differ from the actual synthesis targets of circuits implemented in modern technologies. In consequence, a substantial post synthesis technology mapping effort is required. Unfortunately, the technology mapping can not guarantee proper final results, because the initial circuit synthesis is performed without close relation to the actual synthesis target.

From the above it should be clear that, for the modern circuit implementation technologies, ***a new much more adequate circuit synthesis technology is needed that will enable the following***:

- *consideration of all possible circuit implementation structures during the synthesis*;
- *direct synthesis into specific technology targets*;
- *synthesis of robust more regular circuits with minimized interconnects*;
- *explicitly accounting for the actual area, timing and power related information*;
- *performing the total multi-objective optimization of the circuit's quality and effective trade-off exploitation among the different objectives*.

According to our knowledge such a circuit synthesis technology did not exist till now: none of the commercial circuit synthesis tools or published research tools has the above features. Therefore, we developed a new information-driven circuit synthesis technology that satisfies the above requirements. The

978-0-7695-3277-6/08 $25.00 © 2008 IEEE
DOI 10.1109/DSD.2008.13

main advantages of our new circuit synthesis technology are the following:

- *generality and high flexibility*: accounting for all possible circuit realization structures and trade-offs among the circuit area, power consumption and speed;
- *direct synthesis into the technology primitives of a given circuit implementation technology* (e.g. LUTs or gates);
- *very effective and efficient processing of incompletely specified functions*;
- *minimization of the number and length of interconnections*;
- *simplicity and regularity of the circuit structures synthesized*;
- *enhanced route-ability, low usage of resources, high-speed and low power consumption* resulting from the circuits compactness, regularity, and minimized interconnects;
- *efficient direct collaboration with physical synthesis*, due to the natural ability to directly account for the timing and/or power related information from placement and/or routing.

This all contributes to the ***superior result quality*** comparing to the traditional circuit synthesis technologies.

2. Information-driven circuit synthesis

Our **information-driven circuit synthesis** is based two theories:

- ***the theory of general decomposition of discrete relation networks*** [2][3], and
- ***the theory of information relationships and measures*** [4].

The information-driven circuit synthesis approach relies on the analysis of the information flow structure and relationships in the function to be implemented, as well as, in the circuit under construction, and usage of the results of this analysis to control the circuit construction. Information flows in the circuit are appropriately ordered, combined, compressed and kept as local as possible. In this way both interconnections and active elements are minimized. **The information-driven approach uses**:

- **general decomposition generator** that is able to generate ***all correct circuit structures for a given function*** (no structures are excluded a priori);
- **information relationships and measures** to control the generator in order to ***efficiently construct only the most promising circuit structures***;
- **timing, power and area related information** (e.g. physical gate characteristics, signal arrival and required times, signal activity etc.) to control the ***satisfaction of the optimization constraints and objectives***, and enable the ***multi-objective optimization*** and ***trade-off exploitation***.

The circuit synthesis is not divided into the technology independent logic synthesis and technology mapping, but is directly performed into the primitives of a given implementation technology (e.g. gates of a given technology library). The technology library description is one of the input data to our synthesis tool. From this description our tool automatically extracts all the functional and physical information required for the multi-objective circuit synthesis. Information relationships and measures make it possible to control the circuit convergence, compactness and interconnections. Our approach minimizes both the number and length of interconnections and explicitly accounts for area, timing and power consumption. Since in parallel to information relationships and measures any sort of additional information can be accounted for (as e.g. related to the signal timing or activity), the timing and power driven synthesis, as well as very flexible and precise delay, power and area tradeoffs are possible. In consequence, the circuits synthesized are small, ultra-fast and low-power at the same time. ***This all together fulfils the requirements of an adequate circuit synthesis for the modern nano CMOS technologies*** as formulated in Section 1.

The apparatus of information relationships and measures facilitates the analysis and quantitative measurement of the information flows and their relationships. Some main ideas of this apparatus are briefly introduced below. Let us consider a finite set of elements *S*, called symbols. Information about symbols pertains to the ability to distinguish certain symbols from other symbols. Table 1 shows the truth table of a multi-output Boolean function. Each row of the truth table (function's product term) is represented by a unique symbol from *S*. Through its two values 0 and 1, variable x_1 induces two compatibility classes on the symbols (terms): B^0={0,2,3,4} and B^1={1,2,3,5}. x_1 has value 0 (1) for each symbol in class B^0 (B^1) (don't care '-' means: 0 and 1). Variable x_1 is not able to distinguish between symbols 0, 2, 3, and 4, because they belong to the same compatibility class. x_1 is able to distinguish between 4 and 5, because they are not placed together in any compatibility class. In this way information is modelled with set systems [3] [4]. Elementary *information* describes the ability to distinguish a certain single symbol s_i from another single symbol s_j ($s_i, s_j \in S$ and $s_i \neq s_j$). Any set of such atomic portions of information can be represented by an *information set* IS defined on $S \times S$ as follows [4]: IS = {{s_i, s_j} | s_i is distinguished from s_j in the

Table 1. Example 3-input 2-output Boolean function *f*

S	$x_1\ x_2\ x_3$	$f_1\ f_2$
0	0 0 0	0 0
1	1 1 1	0 0
2	- 0 1	0 1
3	- 1 0	0 1
4	0 1 1	1 1
5	1 0 0	1 1

information modelled}. For instance, information given by set system $\pi_{x1}=\{\overline{0,2,3,4};\overline{1,2,3,5}\}$ induced by x_1, can be represented by information set $\text{IS}(\pi_{x1})=\{0|1\ 0|5\ 1|4\ 4|5\}$. Information relationships between variables or set systems representing various information streams can be analyzed by considering relationships between their corresponding information sets. In particular, the relationship and relationship measure expressing information similarity are defined in [4] as follows:

- *common information* CI (i.e. information that is present in both π_1 and π_2): $\text{CI}(\pi_1,\pi_2) = \text{IS}(\pi_1) \cap \text{IS}(\pi_2)$
- *information similarity (affinity) measure* ISIM: $\text{ISIM}(\pi_1, \pi_2) = |\text{CI}(\pi_1, \pi_2)|$

(e. g. $\text{CI}(\pi_{f1},\pi_{x1}) = \{0|5\ 1|4\}$, $\text{ISIM}((\pi_{f1},\pi_{x1}) = 2)$.

In real applications, we use some more complex normalized and weighted measures obtained through associating an appropriate importance weight $w(s_i|s_j)$ with each elementary information and combining selected simple measures.

In a single step of general functional decomposition, function ***f*** being decomposed is split into two sub-functions (see Fig. 1): ***predecessor sub-function g*** and ***successor sub-function h***. The input support of *f* is divided into two subsets: ***bound-set* U**, being the *g*'s input support, and ***free-set* V**, being a partial input support of *h*. Outputs of *g* constitute the remaining part of the *h*'s support. This single decomposition step is recursively applied to both predecessor and successor functions until each sub-function in the network constructed this way can be directly mapped onto gates in a given technology library.

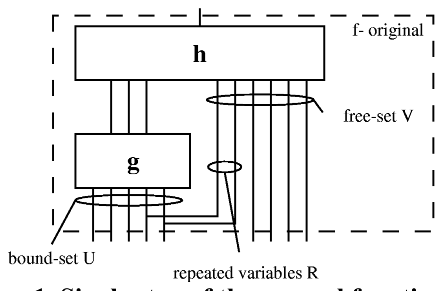

Figure 1. Single step of the general functional decomposition

Our circuit synthesis constructs the circuit level by level from its primary inputs to primary outputs (bottom-up) through repeating the single decomposition step (Fig. 2). The bottom-up synthesis enables continuous precise control of timing, area and power consumption during the synthesis. The continuous availability of precise information on the timing, area and power consumption of the already synthesized bottom part of the circuit, as well as, on the corresponding characteristics of support signals for its upper (to synthesize) part enable very well informed synthesis decisions and very precise trade-off exploitation. At each level, the input support (primary inputs and/or intermediate variables) of the not yet synthesized part of a function being decomposed has to provide all information necessary to compute the function's output values. However, information necessary for computing the function's values is distributed across its support variables. These variables also contain some redundant information. To implement the function, the decomposition network has to eliminate the redundant information, and preserve and restructure the required information, to finally represent the required information at the output as demanded by the function. Consequently, each sub-function *g* should eliminate some redundant information and combine the required information from its inputs, transfer the required information to its output and represent it in an appropriate manner. The *bound-set U* determines *what information is delivered* to a certain sub-function g. The *g*'s *output set system* π_g determines *what information is transferred* by *g*. *U* and π_g together define the multi-valued function of *g*. In order to implement this function in binary hardware, it has to be transformed into a set of binary functions. The g's binary functions determine *how the transferred information is represented* at the *g*'s binary outputs [5].

The **sub-function construction procedure** is composed of the following steps:

1. Construct a limited set of the most promising bound-sets *U* and corresponding output set systems π_g.
2. Order the input supports *U* from the set constructed

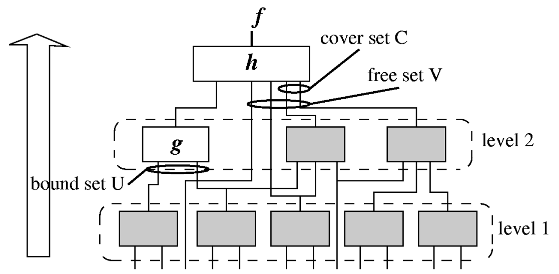

Figure 2. Bottom-up functional decomposition

in step 1 according to their quality.
3. Consider a limited set of the supports U in the order of their quality, and for each support U construct a set of the corresponding binary gate implementations of the multi-valued sub-function g.
4. From the set of implementations constructed in step 3, select the implementation that maximizes the signal and/or information convergence of the sub-function g and optimizes a given area/delay/power trade-off.
5. Construct a new function h by expressing f in new variables.

The first four steps are guided by analysis of the information relationships, gate characteristics, area, timing and power related information, and optimization constraints and objectives. They were explained in our previous publications [4][5]. The last step is straightforward.

4. Requirements of library modelling for information-driven circuit synthesis

In the traditional two-step circuit synthesis process that involves the technology independent logic synthesis and technology mapping, the initial synthesis is often performed in wrong direction and the design freedom is used on random instead of being carefully exploited for the actual network optimization. This results in inferior synthesis results. To eliminate this problem, we proposed to replace the two-step synthesis with a single-step direct circuit synthesis (direct mapping) into the gates of a given technology library, when directly accounting for the actual implementation costs.

The single-step circuit synthesis process requires availability of adequately complete and accurate information on the logical and physical features of the technology gates from the very beginning of the circuit synthesis process, as well as an effective and efficient usage of this information throughout the whole process. The corresponding data structures must enable an accurate modelling of this information, effective search for gates during the sub-function construction process, and efficient application of the selected gates in the network under construction. In particular, the single-step synthesis requires:

- an adequate characterization of gates' logic features and physical features related to area, timing and power dissipation, and
- a methodology to efficiently provide correspondence between the representations of multi-valued sub-functions in the decomposition and the functional representations used in the characterization of physical gates from a given technology library.

To guarantee the generality of application of our information-driven circuit synthesis approach to every gate-based technology and every library, the actual circuit synthesis methods, algorithms and heuristics have to be independent of any particular technology or library – the technology or library specific features have only to be used as data for the circuit synthesis methods, algorithms and heuristics.

The gates' functional representation of the library model has to ensure the following:

- adequate modelling of each gate's function from the information viewpoint, to allow for an efficient computation of the information relationships and measures necessary for the information-driven synthesis;
- efficient realization (compatibility) check between the required and available functions,
- easy translation between the function's term and minterm representations.

While the execution-time efficiency is crucial, because the above operations are performed multiple times during the synthesis process, the memory efficiency is not critical, due to the small sizes of the input sets of typical gates (up to 4 or 5 inputs). Also the input supports of sub-functions in general decomposition that are not further decomposed, but directly mapped into gates are not higher than 6, and usually lower or equal to 4. The small input sizes allow for an efficient representation and processing of Boolean functions using *minterms*. For simplicity of different function manipulations (such as: comparison, etc.) a canonical minterm representation must be developed and used. Such a representation has to satisfy the following requirements:

- a function must be completely specified;
- every input combination has to represent a minterm;
- the input combinations (minterms) have to be ordered using an arbitrary ordering function, e.g. sorted in the ascending (descending) numerical order, when each input minterm is considered as a binary number, and the correspondingly ordered output values have to be stored in the form of a vector of binary values;
- a (decimal or hexadecimal) integer translation of the basic binary representation may also be used, and a human readable form, if convenient.

We use a canonical compact minterm representation being a simple implementation of the function's signature. Any Boolean function, for which the order of its minterms is fixed (e.g. ascending numerical order), is unambiguously defined through the corresponding vector of its output values, referred to as the function *signature* or *label*. We implemented the signature as a bit vector whose successive values correspond to the function's values. For a Boolean function of n inputs, the length of a bit vector representing its signature is equal to the number of all unique input minterms, i.e. $k = 2^n$. For a modern 32-bit architecture computer, it gives the size of 5 inputs to fill its arithmetic logic unit completely. For wider supports, the signature processing has to be performed in parts. Thus, for modern processors, the performance of the signature processing algorithms is high and constant for functions of up to 5 inputs, and it drops exponentially with every additional function's input. Consequently, the compact minterm representation can not be used for large functions. To overcome this problem, the Boolean functions are required to be efficiently translated back and forth between the general-term and minterm representations.

Since each physical gate implements a corresponding completely specified function, the output set system of a physical gate is actually a bi-partition. One block of the output set system corresponds to the ON set of the compact minterm representation and the other to the OFF set. The translation between the signature, set system, and information set representation of each gate's function is thus quite straightforward.

5. Library pre-characterisation

We developed a library modelling method and corresponding tool that automatically constructs technology library models satisfying the above requirements. To guarantee the generality of application of our information-driven circuit synthesis approach to every gate-based technology and every library, the technology or library specific information is used as data for the circuit synthesis methods, algorithms and heuristics, and is this way exploited during the synthesis process to optimize the circuit under construction. Not a single feature related to a specific technology library is hard-coded into any of the algorithms or heuristics of the actual circuit synthesis method. To insulate the actual synthesis process from a specific technology library and to facilitate the actual synthesis, the library is pre-characterized during the initialisation phase of the synthesis tool. The initialization phase has to be performed once for each different library, before the actual synthesis can start.

The original target technology library description given in the form of an ASCII file is parsed and converted into an internal library model, suitable for an efficient library search with our multi-valued sub-function construction procedures. The library pre-characterisation process prepares and fills the data structures containing all and only the information about each gate of a given technology library that is necessary for an effective circuit synthesis performed through the information-driven general decomposition. To facilitate the library search and function matching during the multi-valued sub-function construction, a ***homogeneous gate instance library*** is constructed, where each physical gate is represented by its all actually distinct application instances, expressed using the same uniform efficient data structures throughout the whole gate instance library. This homogenous instance library forms a homogenous search space for the sub-function construction algorithms, what simplifies the algorithms and enhances their efficiency.

Each gate has at most $n!$ different applications, expressed with its different ***representatives***, i.e. non-equivalent, distinct functions realized by the gate for its all possible input permutations. Each gate representative is characterized by its *input permutation*, and corresponding *Boolean function* it implements for this input permutation.

During the matching phase of the sub-function construction process, a (not) completely specified Boolean sub-function required to be realized is matched against the list of candidate gate representatives in the homogenous gate instance library. To significantly enlarge the number of possible direct applications of each physical gate, we use the definition of Boolean matching for circuit synthesis extended as follows. ***Two single-output combinational functions f(x) and g(x) (with the same number of input variables) match when they are NPN-equivalent***. Here, the Boolean functions that are equivalent under negation of inputs form an *N-equivalence class*, under permutation of inputs a *P-equivalence class*, and under negation of inputs, permutation of inputs, or negation of outputs, *an NPN-equivalence class.* Thus, a set of Boolean functions is considered to be *NPN-equivalent* if and only if there is a variant of the NPN transformations, so that after applying them these functions become functionally equivalent. The extended Boolean matching implies resolving if two Boolean functions are the same under negation of inputs, permutation of inputs, and/or negation of outputs. Its goal is to explore in the network under construction the modifications related to the De Morgan's laws and simple polarisation transformations on the gates' inputs and/or outputs. The NPN transformations used determine the way particular selected gate is connected in the network under construction, i.e.:

- which inputs must be inverted using the input inverters and which must be directly connected,

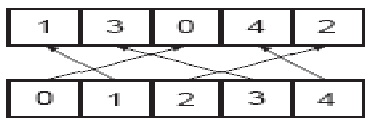

Figure 3. Permutation representation

- which external inputs must be connected to which gate inputs, and
- whether the output requires an output inverter.

During the circuit synthesis, P transformation is realized through an appropriate permutation of the actual gate inputs and connection of the corresponding input signals to the inputs of a particular gate, to implement its corresponding application in the network under construction. Thus, the implementation of this transformation does not result in any substantial cost change of the resulting network. N transformation changes the polarisation of the input and/or output signals. The change of polarisation is simply realised through the insertion of an inverter. The costs of N transformation are thus low and defined by the inverter costs in a given technology. An inverter may add a small extra area, delay or power consumption. However, many of the input inverters can be later eliminated, when connected an inverted gate input to an inverted output of a gate already present in the circuit under construction or during the circuit post processing. Only the inverters directly placed by the primary inputs (the first circuit level) cannot be later simply removed, and therefore, it is not desired to use gates with inverted inputs at the first circuit level. The post-processing removes the inverters placed inside the network, when a gate that drives a given input inverter of the next gate has its complementary equivalent in the representative library. In this case, the output inverter of the driving gate connected to the input inverter of the driven gate result in double inversion, i.e. in a direct interconnection of the driving gate output with the driven gate input. To have adequate information for the inverter elimination, all pairs of mutual complementary gates present in the library are detected, and with each such gate information is associated denoting the complementarity, and pointing to the complementary gate. The complementarity feature is exploited in the post-processing, as well as in the main decomposition process. Since the complementary gates are equivalent in relation to the information processing, a better of them from the physical viewpoint is selected and used.

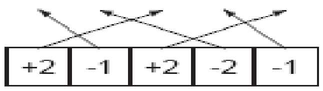

Figure 4. Permutation from Fig. 2 represented in *shift notation*

Moreover, during the uniform instance library creation, all possible gate instance replicas (i.e. instances that realize the same function) found in the library are compared, and only the non-dominated of them (e.g. smaller footprint area or faster) are kept. We also introduced special "don't care" (DC) inputs to represent the extra inputs of a gate that actually has fewer inputs, in order to use the gate for the sub-functions with wider input supports. The DC inputs are denoted in the corresponding formula as Boolean alternative of a positive and inverted input.

Exploration of the gate input permutations during the matching phase requires an efficient data structure for representation of input permutations. The aim of the permutation data structure is to efficiently represent all possible applications of a particular gate for its different fixed input orders. A pair: (input permutation, the gate's Boolean function corresponding to the input permutation) defines a particular gate application. The permutation data structure acts as a "patch panel" or input "switch board", and allows us to denote each possible connection of the *m* external signals to *n* gate inputs, where *m* and *n* are not necessarily equal. For example, the permutation of the basic ascending input order: 0, 1, 2, 3, 4 into the order: 1, 3, 0, 4, 2 is graphically represented in Fig. 3. Observe that each permutation is unambiguously defined through the vector of position shifts of each input variable index in relation to the index position in the basic ascending input order. For example, the input permutation from Fig. 3 is given in the shift notation in Fig. 4. To represent the lack of permutation, one must perform no shifts in the basic permutation. In this case, the vector of shifts consists of zeros in all its positions. To further simplify the permutation data structure, we proposed to denote the shift values in modulo (n) notation, where n is the number of gate inputs. The example from Fig 3 and 4 is given in modulo(*n*) notation in Fig. 5. Due to the simplicity of the permutation data structure and small size of the typically considered input supports, the operations on this data structure are very efficient. When a certain gate representative is being used during the decomposition process to construct a circuit implementing a given sub-function, all the internal inputs of a physical gate have to be connected according to the input permutation of the gate representative to the particular corresponding external inputs. This is performed in a straightforward way through transforming the permutation vector into the corresponding form of the actual input switch board.

Due to symmetries, the number of representatives corresponding to a particular gate can often be substantially reduced. For the symmetric gates (i.e. implementing the symmetric functions as AND, OR,

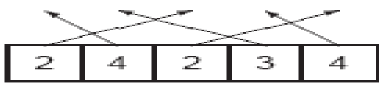

Figure 5. Permutation from Fig. 2 and 3 represented in *modulo(n) notation*

EXOR, etc.) the input permutation does not influence the logic function implemented by the gate. For the partially symmetric gates, the input permutations inside the symmetric input subsets do not influence the function. Thus, for a certain gate, we have only to consider all possible input permutations that result in actually different functions, and can avoid permutations among symmetric inputs.

An example of a Boolean function with an input symmetry, DC input and potential input inverter is the following:

F(A,B,C,D) = (A+A')*(B+C+D'),

where inputs B and C form a two-input symmetry group (BC), input A is a DC input, and input D is negated. We consider the simple input symmetries, as well as, the rotational and group symmetries. An example of a Boolean function with rotational symmetry is the following:

F(A,B,C) = (A+B)*(B+C)*(A+C),

where shifting all three input variables at once does not change the logic function. An example of a Boolean function with higher order group symmetry is the following:

F(A,B,C,D) = (A+B)*(C+D),

where exchanging pairs of input variables ((A,B) with (C,D)) does not change the logic function.

Each library gate is analyzed in terms of its input symmetries. The input symmetries found are exploited to reduce the size of the uniform instance library. Having information about the symmetric and DC inputs, we can limit the total number of distinct input permutations that have to be considered to describe all possible ways of use of a gate for circuit construction, when preserving one permutation of positive/inverted inputs (one representative) of a gate for each distinct Boolean function. For every possible input permutation that results in an actually different function, the truth table of the corresponding function is computed. Subsequently, the input combinations in the truth table are sorted as ascending binary numbers, and the corresponding signature, output set-system and information set representing the function are constructed. In our information-driven decomposition approach, the binary gate implementation of a given (multi-valued) sub-function requires covering of the set of the elementary information items of a given (multi-valued) sub-function by the information set computed by a particular gate or combination of several gates. The related comparisons of the required and delivered information sets are performed multiple times during the decomposition process. In consequence, the optimisation of the information set representation has a significant impact on the performance of the entire synthesis process.

For example, for the three-input gate that implements the totally asymmetric function: f(A,B,C)=(A+A')*(B+C'), all distinct input permutations must be considered, because for an asymmetric function there are no symmetric inputs subsets. For each distinct input permutation, the corresponding truth table is computed with input combinations as ascending binary numbers. For instance, the input permutation (BCA) results in the following truth table:

symbol	inputs			output
	B	C	A	
0	0	0	0	1
1	0	0	1	1
2	0	1	0	0
3	0	1	1	0
4	1	0	0	1
5	1	0	1	1
6	1	1	0	1
7	1	1	1	1

The output set-system (bi-partition) corresponding to this truth table: $\pi_f = \{\overline{2,3}; \overline{0,1,4,5,6,7}\}$, together with its corresponding information set IS(π_f), the input order (permutation) (BCA) and cell reference are stored in the list of the gates instances. During the decomposition process, to find a gate that implements a required sub-function given through its information set IS, such a set-system π_f (information set IS(π_f)) must be found in the list of the gate instances that its information IS(π_f) covers IS, i.e. IS$\subseteq$IS(π_f).

The representative set size reduction due to input symmetries depends on a particular library. The more symmetric or partially symmetric gates a given library contains, the greater its corresponding representative set size reduction. Typical libraries contain a lot of completely or partially symmetric gates and this represents a significant reduction potential that can be as high as several times. Therefore, it makes much sense to adequately explore and exploit this potential.

The gate physical feature modelling is much more straightforward than an adequate logic feature modelling, and consists of a direct extraction from the original library description in ASCII form of the values of the physical parameters that are required to be considered during the single step-circuit synthesis. Due to the bottom-up synthesis approach and availability of the information on the gates' physical features, a very precise control of the circuit physical features is possible during the information-driven circuit synthesis. At each step of the decomposition process, the consequences of a particular gate application for the area, delay and power consumption are analysed, and the results of this analysis are used to select the most promising binary gate realization of the multi-valued sub-function. Also, the area, delay and power related information is re-computed for the already synthesized part of a circuit under construction.

The **gate instance data structure** of the library model includes the following:

• **logic features**:

– ***logic function realized*** in several representations that are used to different purposes:

- the Boolean function in the formulae format,
- the Boolean function signature being a compact minterm representation/the function's set system representation,
- the function's information set;

– ***input support size***;

– ***input permutation vector***;

– ***bitmask representing the DC-inputs*** (if any);

– ***bitmask representing the inverted inputs*** and/or the ***inverted output*** (if any);

– ***cross references to***:

- the corresponding gate implementing the complementary Boolean function/output phase (if any),
- the corresponding gate implementing the same Boolean function, but without inverted
- inputs (if any);

– ***symmetric inputs*** of group, hierarchical and rotational symmetries (if any);

• **physical features**:

– ***gate foot-print area***;

– ***gate delay***, (separately) for every arc (input pin):

- intrinsic delay:
  · rise,
  · fall,
- output driver:
  · rise (pull-up) strength,
  · fall (pull-down) strength,

– ***power related parameters***.

In result of the above described gates' logic and physical feature modelling, a library model is constructed that can efficiently be used in the single-step information-driven circuit synthesis.

6. Virtual gates

In a typical modern gate library, all 2-input binary functions have their direct gate representation and can directly be mapped onto corresponding gates. Also, most of the 3-input functions have their direct gate representations, but only 10-15% of all possible 4-input functions and a very small fraction of functions with more inputs can directly be mapped onto corresponding library gates. Although the introduction of the gate input and output inverters and related NPN-equivalent matching greatly increases the direct mapping ability, a majority of functions of, for instance, up to 4-inputs is anyway not covered. Thus, a much more sophisticated sub-function construction procedure is required for the synthesis of gate-based circuits than for LUT-based FPGA circuits, where each k-input binary function can be directly mapped onto a k-input LUT, slice or CLB. Moreover, the implementation costs of different k-input functions in an FPGA technology are the same while in a gate library based technology are in general different, due to different area, delay and power characteristics of different gates.

To facilitate an accurate assessment of the area, timing and power dissipation costs during the decomposition process, we decided to create an ***extended gate library*** consisting of the actual physical gates of a given technology library and additional virtual gates. The ***virtual gates*** represent optimal complete decompositions into physical gates of the single-output Boolean functions of up to n-inputs that are not directly implemented with the actual physical library gates. This way, the extended gate library includes implementations of all single-output Boolean functions of up to n inputs. It is practical to limit such an extended library to gates of maximally n inputs, considered as a feasible or practical input support size for the synthesis algorithms executed on a particular computer. For the contemporary personal computers the practical input support size seems to be $n <= 8$, while the computation time for n=4 is substantially lower than for the higher values of n. The practical input support size will slowly (logarithmically) increase with the increase of the computation strengths of computers.

It is possible (but not necessary) to build all the virtual gates and the maximal functionally complete extended library of gates of up to n-inputs in advance, before starting to use our synthesis tool to the actual circuit synthesis. However, the extended library can also be build stepwise, on-the-fly, and according to the actual needs during the decomposition process of functions that require for their realization particular binary sub-functions of up to n inputs that are not yet implemented with the physical gates or earlier constructed virtual gates. Both on-the-fly or in advance, our synthesis tool constructs a corresponding optimal circuit for every (required) single-output Boolean function that is not yet included in the extended library, and includes it as a virtual gate into the extended library to enable a fast direct function mapping on the newly created gate and facilitate the accurate computation of the area, delay and power consumption costs. In the case of the on-the-fly computation, the newly constructed virtual gate is also directly considered for its use in the circuit under construction. The internal structure of a virtual gate being an optimal multi-level and multi-gate realisation of a particular Boolean function, together with the models of gates involved in the virtual gate, give a quite complete and accurate information for construction of a complete virtual gate's model, involving the logic and physical parameters as specified in the previous section. In particular, the area, delay and power characteristics of each virtual gate can be quite accurately computed. The virtual gates can also be pre-synthesized to get more precise values of their physical characteristic from their post-layout analysis, if necessary.

The library model containing all the actually distinct instances of gates from a given technology library and extended with the virtual gates is used as a ***homogenous Boolean function realization library*** during the information-driven general decomposition process. In this library all the (required) single-output Boolean functions of up to n inputs have their corresponding physical or virtual gate realizations that are adequately characterized from the logical and physical viewpoint. The Boolean function realization library is implemented using the earlier described efficient data structures, and enables an effective and efficient multi-valued sub-function construction in the information-driven circuit synthesis, as well as multi-objective circuit optimisation and effective trade-off exploitation among the area, delay and power consumption.

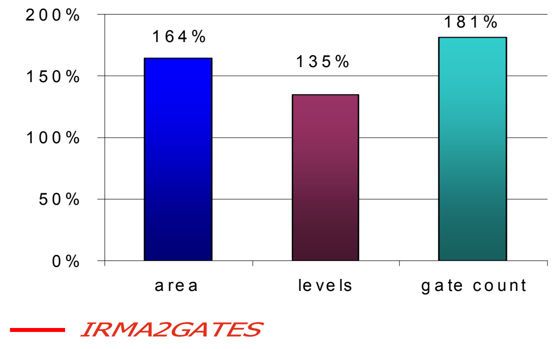

Figure 6. Synthesis result comparison from *IRMA2GATES* to *SIS 1.3* on MCNC benchmarks for the STDcell library [6]

7. Experimental results

To experimentally verify our proposed library modelling process and related library parsing and modelling tool, we modelled among others the MCNC, STDcell and AMS c35b3 libraries, and used the models to synthesize several thousands various circuits. The results of these experiments demonstrated that the proposed library modelling process, as well as library parsing and modelling tool work correctly, and are adequate for the single-step information-driven synthesis. The information-driven circuit synthesis process using the library models is a separate issue and its more precise discussion will be a subject of a separate paper. Nevertheless, the complete circuit construction method briefly discussed in the previous sections has been implemented in our information-driven circuit synthesis tool *IRMA2GATES*. Since this paper is not devoted to the experimental analysis of the circuit synthesis algorithms implemented in *IRMA2GATES*, but to the issue of an adequate technology library modelling for the purpose of single-step information-driven synthesis, only a brief impression of the synthesis result quality delivered by our method is presented below.

In Fig. 6 the synthesis results from our *IRMA2GATES* are compared to the results from the well known UC Berkeley's tool *SIS 1.3* [6] regarding the area, gate-count and number of gate levels on the critical path (delay) for several MCNC benchmarks [7] and other popular functions. The results from *IRMA2GATES* are on average 64% better regarding area, 81% better regarding the number of gates, and 35% regarding the number of gate levels than from SIS. Moreover, we compared *IRMA2GATES* to *SIS* using more than 30 generated symmetric and quasi-symmetric Boolean functions (Fig. 7). We generated a set of 10-input completely specified symmetric functions, and then, mutated the basic functions, by replacing 20%, 50% and 70% of their 1 or 0 output entries with "don't cares" in each completely specified function. The circuits produced by *IRMA2GATES* are on average almost 3 times smaller and 25% faster than the circuits synthesized by *SIS*. These results demonstrate that IRMA2GATES is especially effective for the symmetric, quasi-symmetric and incompletely specified functions. More information on the benchmarks used and experimental results from our tool can be found in our other paper [5] devoted to different aspects of the method than this paper.

8. Conclusion

We developed a new effective, efficient and very flexible circuit synthesis technology adequate for the modern synthesis targets. The technology implements our original information-driven approach to circuit synthesis. It replaces the traditional dual-step process of technology independent logic synthesis and technology mapping with a single-step direct circuit synthesis (direct mapping) into the gates of a given technology library, when directly accounting for the actual implementation costs. The single-step circuit synthesis process requires availability of adequately complete and accurate information on the logical and physical features of the technology gates from the very beginning of the circuit synthesis process, as well as an effective and efficient usage of this information throughout the whole process. To satisfy these requirements, we developed a new library modelling method and implemented it in the form of a library parsing and modelling tool that automatically creates an adequate library model in the form of a homogenous Boolean function realization library, through constructing efficient data structures and filling them with the required information on the gates' logic and physical features. This library model enables an effective and efficient multi-valued sub-function construction in the information-driven decomposition process, as well as, the multi-objective circuit optimisation and effective trade-off exploitation among its area, delay and power consumption. The experimental results demonstrate a high quality of our single-step information-driven circuit synthesis approach. Our tools construct substantially smaller and faster circuits than other tools, and enable a multi-objective circuit optimization, trade-off exploitation and very flexible circuit structuring and re-structuring.

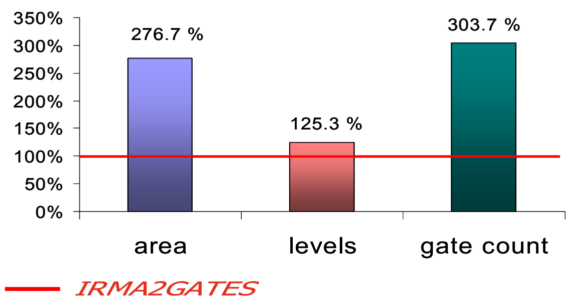

Figure 7. Synthesis result comparison from *IRMA2GATES* to *SIS 1.3* on generated symmetric and quasi-symmetric functions

9. References

[1] J. Cong and K. Minkovich: Optimality Study of Logic Synthesis for LUT-Based FPGAs, FPGA'06, February 22-24, 2006, Monterey, California, USA, ACM, pp. 33-40.

[2] L. Jóźwiak: General Decomposition and Its Use in Digital Circuit Synthesis, VLSI Design: An International Journal of Custom Chip Design Simulation and Testing, vol.3, No 3-4, 1995.

[3] L. Jóźwiak, A. Ślusarczyk: General Decomposition of Incompletely Specified Sequential Machines with Multi-State Behaviour Realisation, Journal of Systems Architecture, Vol. 50, December 2003, pp. 445-492.

[4] L. Jóźwiak: Information Relationships and Measures - An Analysis Apparatus for Efficient Information System Synthesis, *23rd EUROMICRO Conference*, Budapest, Hungary, September 1-4, 1997, IEEE Computer Society Press, pp. 13-23.

[5] L. Jóźwiak, S. Biegański, A. Chojnacki: Information-driven Circuit Synthesis with the Pre-characterized Gate Libraries, Journal of Systems Architecture, Elsevier Science, Amsterdam, The Netherlands, Vol. 51, No 6-7, June-July 2005, pp. 405-423.

[6] E. M. Sentovich, K. Singth, L. J., Lavagno, C. Moon, R. Murgai, A. Saldanha, H. Savoj, P. R. Stephan, R. K. Brayton, A. Sangiovanni-Vincentelli, SIS: A system for sequential circuit synthesis, Memorandum No. UCB/ERL M92/41, Electronic Research Laboratory, University of California, Berkeley. ftp://ic.eecs.berkeley.edu/pub/Sis/Sis-paper.ps.Z

[7] Collaborative Benchmarking Laboratory, Department of Computer Science at North Carolina State University, http://www.cbl.ncsu.edu/

[8] The 2005 International Technology Roadmap for Semiconductors, SIA, San Jose, CA, USA, 2005, http://www.itrs.net/Links/2005ITRS/Home2005.htm