Urszula Stańczyk
Krzysztof A. Cyran
Silesian University of Technology, Institute of Informatics

On logic function representation type choice for implementation of image processing

It is a well known and established fact that so-called finite window operators [1] can be expressed by logic functions, whose variables correspond to points from the neighbourhood defined by the processing window. Pixel values of binary images are then directly interpreted as variables values of a logic function, allowing for evaluation of the output.

Image processing can be implemented by hardware or software, each type characterised by some general and some specific parameters. Among general factors there should be named computational complexity of processing, time of obtaining the resulting image, and data storage for software implementations or structure size for hardware implementations.

However, for all implementations the purely graphic representations can be disregarded as too limited. This leaves truth tables, standard forms, logic expressions, characteristic numbers, decision diagrams, partitions and blankets and sets of cubes. Yet this list is redundant as numerical standard forms correspond to parts of a truth table and in literal form they are logic expressions. Also for assumed ordering of variables the bits of the binary characteristic number can be read from rows of a truth table, with LSB from the first row of all 0s and MSB from the last row of all 1s. The opposite action would create the truth table from a characteristic number, which for many variables becomes prohibitively big. That leaves the short list of truth tables, logic expressions, DDs, partitions and blankets and sets of cubes as reasonable choices in general, still not all of them, however, are efficient for all functions and types of implementation.

For hardware realisations there are usually used such switching elements as gates, or medium and large scale integrated circuits such as commutators, programmable logic devices or memory blocks. These elements can apply two kinds of representation: truth tables for realisations with commutators or memory blocks; or logic expressions for gates or PLDs [2].

The complexity of implementation with commutators depends on relation between the number of function variables and the number of address inputs to an element. If the number of variables to drive the commutator exceeds the number of address inputs then the realisation becomes complex resulting in a cascade connection of multiplexers; connections with demultiplexers or using gates to produce auxiliary sub-functions. With many variables support the multi-level structures are unavoidable and there comes with it lengthening of the processing time.

Memory blocks are currently relatively cheap elements. Variables of the function work as address to the memory and for binary function the memory block requires only one bit to store the functional value and a single output where this value is set.

For gates and PLD-based realisations usually the best form of logic expression contains the fewest literals and operators, which may be either minimal SoP or PoS form obtained through two-level minimisation process, parenthesised factored form that is a result of multi-level transformations or functionally decomposed function. Again multi-level structures can be used, this time, however, they are caused by multi-level structure of the implemented expression.

For software implementation all entries from the short list of representations have some pros and cons. A truth table is canonical but its disadvantage lies in its size. Obviously, it is possible to store just on-set or off-set, which cuts the number of values significantly. However, assuming certain weights for variables and some order with functional values it is then necessary to store all the combinations of variables to which each functional value corresponds. In the truth table to find the functional value for the given combination of inputs, the table data structure is accessed by simple index based on the variable combination. When only part of the truth table is stored the table has to be either linearly searched for possible match with input combination or there are needed some other mechanisms simplifying the search. Still the average access time will be longer. On the other hand, the size of the truth table can be so prohibitive that it may require storing in extended memory or to some type of disk drive, the access to which takes much longer. Cutting this size can enable to use conventional memory and work faster even with linear searching.

Another problem to be considered is the fact that software data types are rather thought of in units of bytes, words, double words and not single bits. Thus either there is allowed some waste of space or with compact storage there is needed the procedure of accessing single bits within byte, which, although simple to implement, requires yet additional processor time.

Sets of cubes are a more compact form of a truth table as they can gather into one row data contained in several rows in the truth table. It happens, however, at the expense of simplicity of access. Like in all reduced representation types to establish the functional value for any given input combination there is needed some kind of search through cubes.

The usefulness of decision diagrams as a data structure defining a logic function depends on a complexity of this function. Like all dynamic data structures, apart from fields containing the data necessary for function definition, they require additional fields for storage of pointers to its constituent elements. Although in many cases very convenient, in others DDs suffer from so-called memory explo-

sion problem, that is exponential dependency on the number of input variables or strong dependency on variable ordering.

Partitions and blankets are often used for the intended manipulation of a logic function. Application of this form of representation just to find functional values reminds both a truth table and sets of cubes, because the output blanket specifies for which symbolic labels corresponding to rows of combinations of input variables the function assumes value one and for which there is zero. Labels can directly correspond to binary numbers described in the row if there are no don't cares in the input parts of all rows and then the blanket just indicates on-set and off-set of the function.

Instead of keeping the values of a function, its logical expression may be used. Processing with this form, however, strongly depends on the complexity of the obtained formula. The least complicated evaluation happens for minimised SoP or PoS forms, but still with many terms it may take significant time to arrive at the output value.

Evaluating parenthesised factored forms takes longer because of some several levels of nesting, but when they contain fewer literals than any of minimal forms, they can be of advantage.

Decomposed function can be treated in terms of truth tables of the output function and sub-functions, or through describing these functions logic expressions. By definition at least two-level, but not necessarily that much nested as a factorised form, can be simplified by using multiple-valued notation. It allows for producing the functional value with lowered realisation costs through construction of some number of simpler sub-functions [3].

It is quite clear that the choice of logic function representation type for intended implementation is not straightforward. Furthermore, given task specification may at the logic synthesis stage pre-determine some type even when it is not the one needed for realisation. Fortunately, once there is obtained at least one form of formal representation, it is always possible to change it into another.

References

[1] H. J. A. M. Heijmans, *Mathematical Morphology: a modern approach in image processing based on algebra and geometry*, SIAM Review 37 (1995), 1–36.

[2] T. Łuba, *Synteza układów logicznych*, Oficyna Wydawnicza Politechniki Warszawskiej, Warszawa 2005.

[3] U. Stańczyk, K. Cyran, B. Pochopień, *Theory of logic circuits* vol. 1 — *Fundamental issues*, Wydawnictwo Politechniki Śląskiej, Gliwice 2007.