# TDS: A Behavioral Transformation System based on Canonical Dataflow Representation

M. Ciesielski, Q.Ren, D.Gomez-Prado
University of Massachusetts, Amherst, USA

J. Guillot, E. Boutillon
Lab-STICC, Université Europeéne de Bretagne - UBS, France

**Abstract**: TDS is an experimental software tool for behavioral transformations of designs specified on algorithmic and behavioral levels. It transforms the initial design specifications into a data flow graph (DFG) optimized for a particular objective (latency, resource utilization, etc.) prior to high-level synthesis. The behavioral transformations are based on graph-based canonical representation, Taylor Expansion Diagram (TED), and are followed by structural transformations of the resulting DFG network. The system is intended for data-flow and computation-intensive designs used in digital signal processing applications.

## TDS System

The design, initially specified in C, C++ or behavioral HDL, is translated into a hybrid network composed of islands of functional blocks, represented using canonical, graph-based data structure (Taylor Expansion Diagram, TED), and structural operators. TEDs are constructed from polynomial expressions describing functionality of the arithmetic components of the design. Each TED is then transformed into a structural data flow graph (DFG) through a series of decomposition operations, including TED variable ordering, linearization, factorization, common subexpression elimination (CSE), and TED decomposition. An important feature of this process is that it minimizes the number of arithmetic operations, in particular multiplications, hence contributing to area minimization of final hardware implementation. This task is accomplished by properly ordering TED variables. The DFG obtained as a result of TED decomposition is then combined with the remaining "structural" operators (such as comparators, etc.), which cannot be represented functionally as TEDs. The entire global DFG network is further restructured to minimize the design latency, subject to the imposed resource constraints.

## TED Decomposition

TED decomposition is based on identifying common subexpressions and replacing them by new variables (e.g., $S_1 = c+d$, $S_2 = a+b$, in Fig. 2). Then, a cut-based decomposition is performed by identifying *split edges*, which decompose the graph disjunctively and introduce *ADD* operations; and *cut nodes* (dominators), which decompose the graph conjunctively and introduce *MULT* operations. The result is a Normal Factored Form (NFF), which is unique for a TED with fixed variable order. Different variable orderings will result in different factored forms and hence different DFGs.

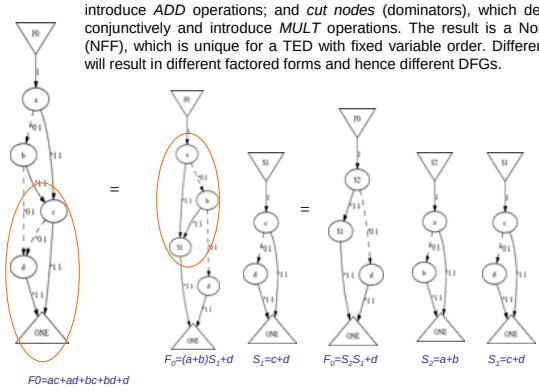



**Fig. 2 TED decomposition leading to a unique Normal Factored Form: $F_0=(a+b)(c+d)+d$**

## Replacing Constant Multipliers by Shifters

In addition to minimizing the number of multiplications during TED decomposition, the system replaces constant multiplications by shift operations, as illustrated in Fig. 4 for function *F=7a+6b*. First, constants coefficients are represented in canonical sign digit (CSD) form, and constant 2 is replaced by variable *L* (left shift). The modified TED is then decomposed by factoring out *L*, resulting in the DFG shown in the right part of the figure.

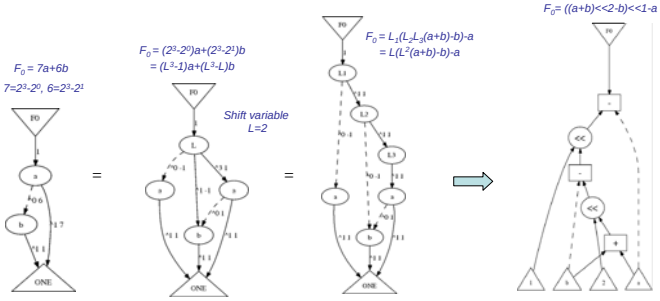



**Fig. 4 Transforming TED with constant multiplications into DFG with shifters**

## References

[1] J. Guillot et al. *Efficient Factorization of DSP Transforms using TEDs*, DATE'06
[2] M. Ciesielski et al. *Data-Flow Transformations using Taylor Expansion Diagrams*, DATE'07
[3] GAUT software: http://web.univ-ubs.fr/lester/www-gaut/
[4] TDS software: http://tango.ecs.umass.edu/TED/Doc/html/

The overall TDS system flow is shown in Fig. 1. The left part of the figure shows traditional high-level synthesis flow (using high-level synthesis tool GAUT), which extracts a data flow (DFG) from the initial specification. The flow on the right shows the TDS system, which transforms the initial DFG into an optimized DFG, which is then passed back to GAUT for high-level synthesis.

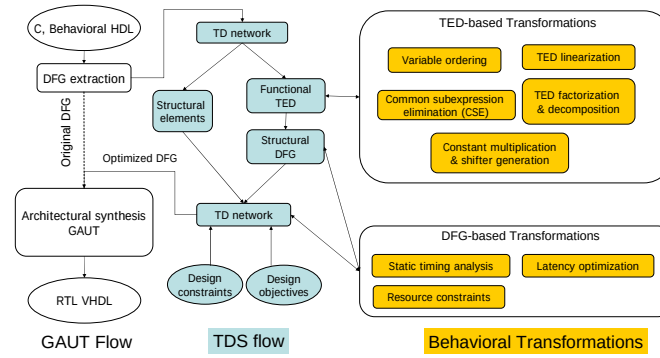



**Fig. 1 TDS system flow**

## Transforming TED into DFG

Once a TED is decomposed into a Normal Factored Form (NFF), a structural representation (DFG) is generated for that form by replacing additions by ADD operations and multiplications by MULT operations. Unlike NFF, a DFG is not unique, and a number of restructuring algorithms can be applied to minimize the expected latency, considering design constraints imposed on hardware resources.

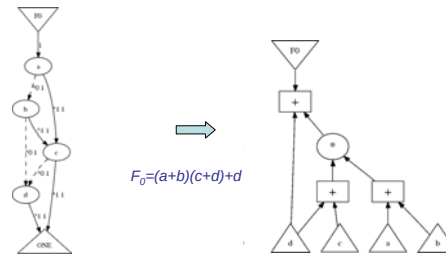
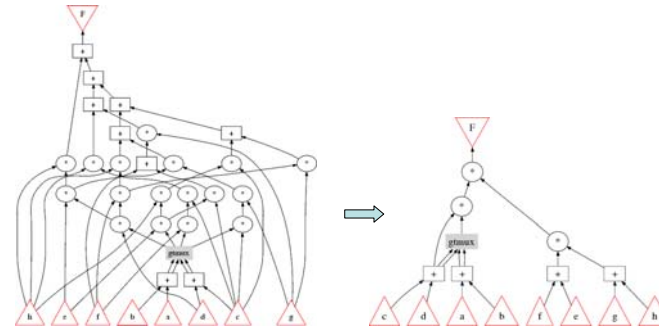



**Fig. 3 Transforming TED into DFG based on Normal Factored Form obtained by TED decomposition**

## TD Network Optimization

The diagram in Fig. 5(a) shows an initial DFG network for a simple design that involves arithmetic operators that can be written as functional expressions and represented as TEDs, and a structural element (*gtmux*, shown in gray), considered as a black box. The diagram in Fig. 5(b) shows the restructured DFG, with the number of multipliers + adders reduced from 17+9 to 3+4, and the latency reduced from 9 to 4. This DFG is then used as input to high-level synthesis to produce designs with smaller hardware area or smaller latency than from the original DFG derived directly from the initial design specification.




(a) (b)

**Fig. 5 TD Network optimization: (a) initial TD network, (b) optimized DFG**

*This work was supported in part by the National Science Foundation under grants CCF-0204146 and CCF- 0702506.*