# Internal Representation of Embedded Hardware- / Software - Systems

Th.Benner, J.Henkel, R.Ernst
Technische Universität Braunschweig
Institut für Datenverarbeitungsanlagen
W–3300 Braunschweig

The *Extended Syntax Graph (ESG)* is an approach to combine some advantages of data flow graphs (DFGs) with those of syntax graphs. An embedded system described by $C^x$, a superset of ANSI-C can be entirely represented by the ESG. In the same way, an efficient dataflow analysis becomes possible. The ESG constitutes the central data structure of *COSYMA (COSYnthesis for eMbedded Architectures)*, our experimental tool for automated hardware-software co-design.

**Key Words and Phrases:** software oriented approach, syntax graph, data flow graph

## COSYMA

COSYMA [ErHe92] is a software oriented approach for automated hardware-software co-design. The designer describes the embedded system with $C^x$, a superset of ANSI-C defined by us, which includes timing constraints and parallel processes as significant extensions. The design starts with a software solution except for our defined components and primitive peripherals. Only when timing constraints are violated, code segments are moved to a hardware implementation by an iterative procedure. The result of the partitioning process consists of software for a processor core, one or more coprocessors and peripherial components. These hardware components are synthesized with the synthesis-tool OLYMPUS ([MiMa90]) based on a description in HardwareC which is generated by COSYMA. For the software parts, COSYMA generates ANSI-C code which is then compiled.
Data flow graphs often used for the high-level synthesis are inappropriate for the internal representation of dynamic data structures and parallel processes. In contrast, syntax graphs, as e.g. used by [Mar90], offer such a possibility. Furthermore, syntax graphs can easily be transformed into HardwareC and ANSI-C.
For the partitioning process and estimation, COSYMA requires information about the data dependencies. Using regular syntax graphs this would be a time consuming task. Therefore, we extended the usual syntax graph description by an implicit symbol table and local data flow graphs.

## The Extended SyntaxGraph ESG

The ESG is a directed acyclic graph describing a sequence of declarations, definitions and statements. Each identifier occurring in the graph is accompanied by a pointer to its definition. Conversely, each definition is extended by pointers to all instances. Then, the costs of communication between the processor core and its coprocessors can quickly and easily be determined with the assistance of this implicit symbol table. Only in cases where a variable is used by software as well as by hardware the variable itself is transferred to a coprocessor. More precisely, the implicit symbol table provides an upper bound on the communication costs, because variables might be re-assigned. As an illustration fig. 1 shows a code segment which computes the factorial of a number. The dashed lines correspond to pointers to definitions and instantiations. Each definition starts with a definition node (*def*), each expression starts with an instruction node (*inst*). The loop in this example must be completed within five microseconds. This restriction is represented by the connection between the label nodes *lab1* and *lab2*.

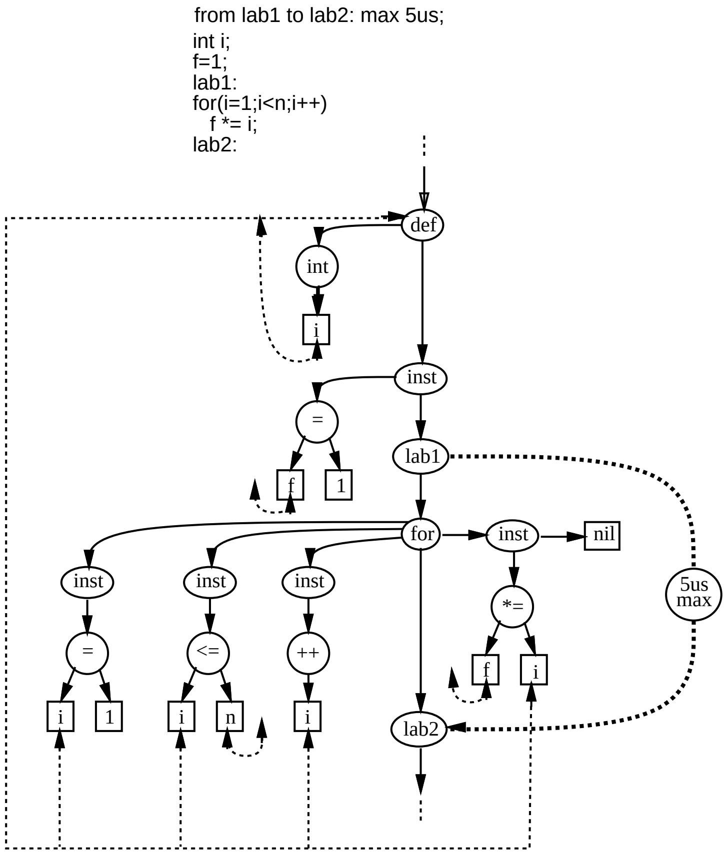


Figure 1: extended syntax graph

The implicit symbol table contains no informations about data dependencies between operations which COSYMA requires to perform a local scheduling on the graph in order to estimate hardware execution times. Therefore, we overlayed the syntax graph by a second graph consisting of cross-linked blocks, the so called *Basic Scheduling Blocks (BSBs).* These BSBs contain local DFGs on which the local scheduling is executed. Both graphs share the same operator nodes thereby enabling a transition from the syntax graph and the DFG and vice versa. Fig.2 illustrates the relationships between both representations. Dashed lines in the syntax graph (Fig. 2a) correspond to pointers to definitions. The operator node indices in both graphs match identical operator nodes.

The ESG accomplishes the requirements of our experimental system. It allows the representation of the entire $C^x$ language and facilitates the compilation to ANSI-C and HardwareC, thereby keeping the syntactical structure of the original description. Based on the implicit symbol table the communication cost between the processor core and the coprocessors can easily be determined. Local scheduling becomes possible by the definition of BSBs.

## References

[ErHe92] R.Ernst, J.Henkel, *Hardware-Software Co-Design of Embedded Controllers Based on Hardware-Extraction,* Int'l Workschop on Hardware-Software Co-Design, Estes Park, Colorado, 30.9-2.10.1992.

[MiMa90] G.D. Micheli, D.C. Ku, F. Mailhot et al., *The OLYMPUS Synthesis System for Digital Design,* Design & Test Magazine, S. 37-53, Okt. 1990.

[Mar90] P. Marwedel, *Ein Software–System zur Synthese von Rechnerstrukturen und zur Erzeugung von Mikrocode,* Forschungsbericht Nr. 356, Universität Dortmund, Juli 1990.

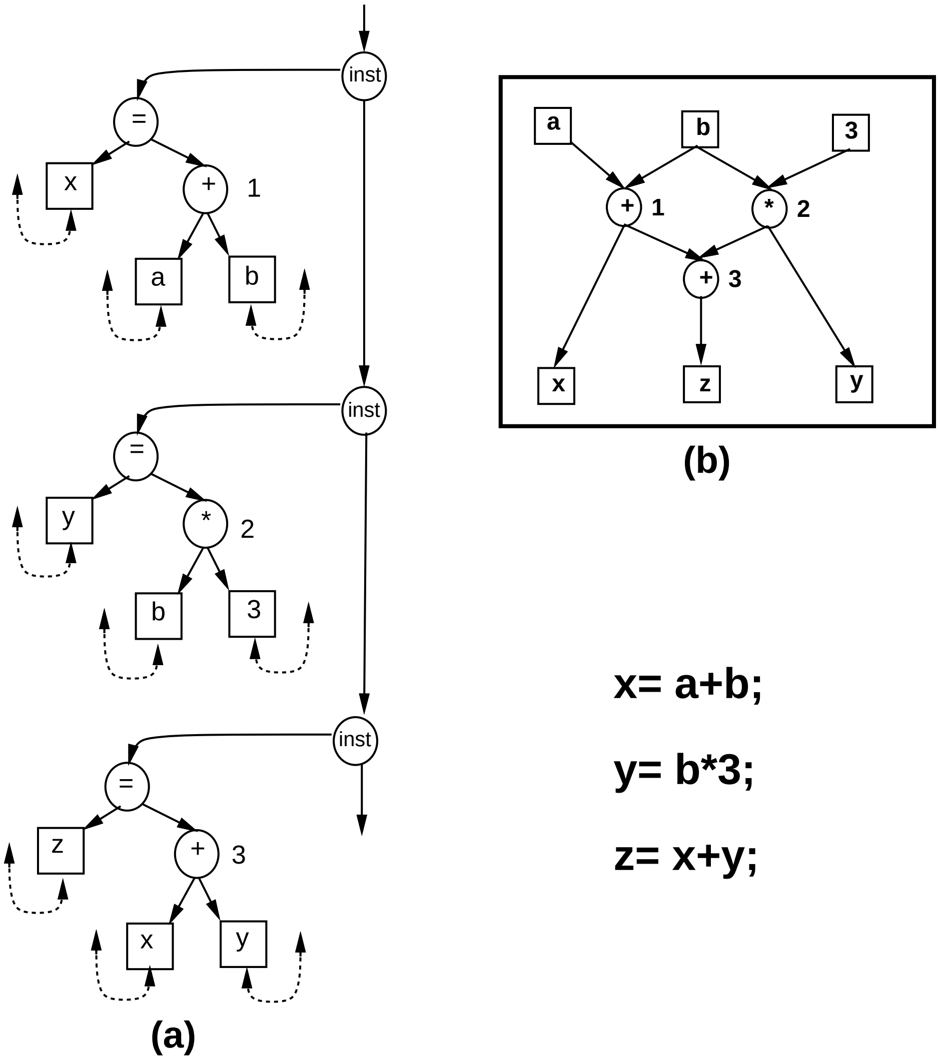


Figure 2: Syntax graph and respective BSB