N88-23234

# ADAPTING HIGH-LEVEL LANGUAGE PROGRAMS FOR PARALLEL PROCESSING USING DATA FLOW

Hilda M. Standley
Department of Computer Science and Engineering
Toledo, Ohio

## ABSTRACT

EASY-FLOW, a very high-level data flow language, is introduced for the purpose of adapting programs written in a conventional high-level language to a parallel environment. The level of parallelism provided is of the large-grained variety in which parallel activities take place between subprograms or processes. A program written in EASY-FLOW is a set of subprogram calls as units, structured by iteration, branching, and distribution constructs. A data flow graph may be deduced from an EASY-FLOW program. All permissible schedulings of executions within the graph are dictated by the data dependencies between units.

NASA Lewis Research Center Grant # NAG 3-699.
Technical Monitor: L. J. Kiraly

## SOFTWARE TO FACILITATE PARALLELISM

Parallel software technology continues to lag behind parallel hardware technology. Synchronization and communication problems have been solved at the hardware level to the degree to enable several multiprocessor systems to be offered commercially. Software is required to provide for the efficient utilization of these multiprocessor systems. Three areas must be addressed in a parallel software solution: (1) the determination of potential parallelism, (2) partitioning the programs into processes or tasks, each of which may be assigned to a single processor, and (3) scheduling the program partitions to execute in a cooperative fashion.

(1) DETERMINE POTENTIAL PARALLELISM

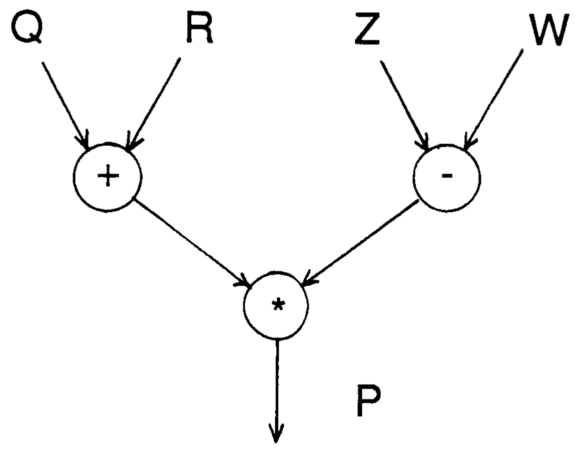
(2) PARTITION A PROGRAM

(3) SCHEDULE PROGRAM PARTITIONS

## DATA FLOW

The data flow schema of parallel computation (Agerwala, 1982) offers at the same time a model of software and hardware. Data values flow between nodes representing operations in a data flow graph. Data flowing into operations serve as operands. Input data values are consumed by an operation and result values output and directed to other operations for which they serve as operands. Execution is completely data driven. The presence of all data values required as operands triggers the execution of an operation. A single "operation" may be low-level (for example, an addition) or high-level (for example, the execution of a subprogram) (Babb, 1984).

$$P := (Q + R) * (Z - W)$$



## LANGUAGE DESIGN PROJECT GOALS

The language design project, resulting in EASY-FLOW (Standley, 1987), has three goals: to develop a language (1) to expose potential parallelism both implicitly and explicitly, at the large-grained level or below (referred to as "variable resolution"), (2) to provide for the continued use of the magnitude of software in existence with only minor modifications, and (3) to require very little retraining of conventional language programmers.

- TO EXPOSE PARALLELISM
- TO USE CURRENT SOFTWARE WITH ONLY MINOR MODIFICATIONS
- TO NECESSITATE VERY LITTLE RETRAINING OF CONVENTIONAL LANGUAGE PROGRAMMERS

## STRUCTURING UNITS WITHIN AN EASY-FLOW PROGRAM

An EASY-FLOW program is specified as a hierarchy of units. Each unit consists of a substructure of units, a reference to an external unit, or it is atomic. An atomic unit is a call to a subprogram expressed in a conventional, high-level language such as FORTRAN or C.

Constructs in the EASY-FLOW language are used to specify the subunits (if any) in a unit and the relationships between them. Lists of "input values" and "output values" associated with each unit may be used to determine the data dependencies between units and can consequently be used in establishing the proper scheduling of unit executions.

EASY-FLOW offers the minimal set of language constructs required for the flow of control: sequencing (SUBPROGRAM call), branching (IF-THEN-ELSE), and looping (ITER for iteration). One additional construct, DISTRIBUTE, provides an explicit notation for parallelism.

- SUBPROGRAM CALL
- IF-THEN-ELSE
- ITER
- DISTRIBUTE

## THE EASY-FLOW PROGRAM

The language constructs provide a framework within which one or more units may be placed. Multiple units appearing within a structure are termed a "unit set." Each unit is enclosed within an input list and an output list pair, stating the names of the data values required as "operands" and produced as results, respectively. A data flow graph may be constructed by the EASY-FLOW language processor from the data dependencies determined by these input/output pairs.

```
MAIN PROGRAM:
DECLARATIONS:
UNIT
ENDUNIT
UNIT
    IF
    THEN
    ELSE
ENDUNIT
UNIT
ENDUNIT
```

## EXAMPLE PROGRAM, MAXTWO

The example program, MAXTWO, calculates the maximum of the values of the two functions, f and g, at a point X. Four names for data values are declared having type "real." The program consists of one unit, called MAIN, with one input value, X, and one output value, RESULT. The body of the MAIN unit is a unit set consisting of three units, each having a call to a subprogram as its body, for calculating functions f(X), g(X), and max(f(X),g(X)). These three units may appear in any order. Three subprograms F, G, and MAX (assumed to be written in a conventional language) must be supplied in order to complete the program.

```
MAXTWO:
    declare: real X,FX,GX,RESULT
    unit MAIN:
        input: X
            unit CALCF:
                into: X => X
                subprogram F(X,FX)
                outof:FX => FX
            endunit CALCF
            unit CALCG:
                into: X => X
                subprogram G(X,GX)
                outof:GX => GX
            endunit CALCG
            unit FINDMAX:
                into: FX => FX
                     GX => GX
                subprogram MAX(FX,GX,
                               RESULT)
                outof:RESULT=>RESULT
            endunit FINDMAX
        output: RESULT
    endunit MAIN
```

## THE EASY-FLOW LANGUAGE PROCESSOR

The EASY-FLOW language processing system constructs a data flow graph from the program and partitions and assigns code for execution based upon the nature of the target machine. If the target machine is a uniprocessor, a topological sort on the nodes in the data flow graph determines an appropriate, although not necessarily unique, unit execution sequence. For a multiprocessor system, the partitioning and assignment may be directed by the data flow graph. The language processing system must also provide for the "sanitizing" of the traditional language subprograms, removing references to global variables, for example.

## EASY-FLOW LANGUAGE PROCESSOR

- CONSTRUCTS DATA-FLOW GRAPH
- PARTITIONS AND ALLOCATES UNIT EXECUTION BASED UPON TARGET ARCHITECTURE
- "SANITIZES" TRADITIONAL LANGUAGE SUBPROGRAMS

## BIBLIOGRAPHY

Agerwala, T., and Arvind, 1982, "Data Flow Systems--Guest Editors' Introduction," computer Vol. 15, No. 2, pp. 10-13.

Babb II, R., 1984, "Parallel Processing with Large-Grained Data Flow Techniques," Computer, Vol. 17, No. 7, pp. 55-61.

Standley, H., 1987, "A Very High Level Language for Large-Grained Data Flow," 1987 ACM Computer Science Conference Proceedings, St. Louis, Mo.