Titel:

A Next-Generation Superscalar Uniprocessor Architecture

Autor:

Dr. sc. techn. Wolfgang Matthes

Franz-Mehring-Straße 22

9006 Chemnitz

Abstract

A uniprocessor superscalar architecture is proposed which comprises four universal operation units arranged according to a tree-shaped dataflow graph. The control principles are based on VLIW, microprogramming, and dataflow concepts. Each of the operation units is an ensemble of high-performance processing resources comparable to state-of-the-art processors (e. g. i860). The whole processor may be implemented with 10 to 50 million transistors, thus being a suitable implementation target for IC technologies of the 90's.

1. The Bases of Experience

Present superscalar architectures had been developed on the basis of comprehensive analytical work, especially relying on instruction level statistics. Such a measurement-oriented approach (MH86) may lead to considerabely good machines, but it has the obvious drawbacks of a disappointingly low rate of usable inherent parallelism and of reflecting current programming habits, leaving opportunities for further innovation untouched. Hence our approach is not to study programs, but to study underlying mathematical structures of important application problems. To obtain initial data we have simply browsed some collections of formulas. Figure 1 shows two frequently needed mathematical operations together with the corresponding dataflow graphs.

2. The Proposed Structure

A more elaborate bookkeeping of the resources needed will show that four universal operation units may be used

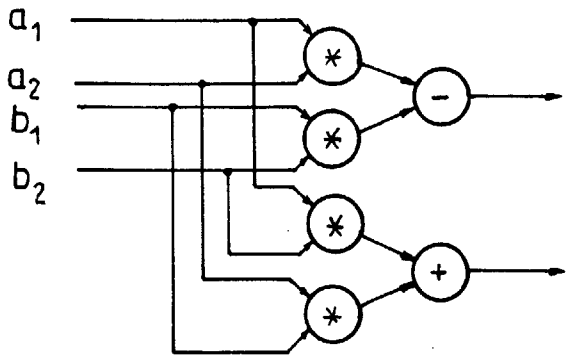
efficiently (MA91). Calculations whose dataflow graph comprises more than four nodes are to be executed in more processing steps. We realize only two essential interconnection structures: (1) none (i. e. independently operating units) and (2) tree-shaped structures.
The basic arrangement is shown in the Figure 2. The units form a tree structure with four operand data paths from memory and one result path to memory. The fourth unit is connected with a stack-organized accumulating memory which is used as the runtime data stack. Independent operation of the four units requires some bypass provisions. This cost/performance tradeoff will cause effecitvity losses if vectorized or unrolled code is to be executed. Figure 3 shows an extension of the proposed structure which avoids this drawback. Each of the operation units has a multipurpose memory (MPM) which can be used as an accumulator, a stack, a collection of vector registers, and a control storage. It has two independent ports for read and write accesses, respectively. Its capacity should be at least 8 kBytes, organized as 1024 buckets of 128 bits (if used as a vector register, it could hold two vectors of 1024 64-bit- elements). The whole structure is connected to the memory subsystem via four read- only and four read/write ports (the latter are used to provide the paths of the tree-shaped structure as well). This scheme allows to load and store the MPMs at maximum speed. Each of the operation units can execute even triadic operations (e. g. SAXPY) with two of the operands delivered from memory and one from the MPM. Results will be stored in the MPMs. They can be moved to memory at maximum speed after the operations have been completed.

3. The Internal Structure of an Operation Unit

Each of the operation units can process numerical and nonnumerical data, respectively. The structure of a processing kernel is shown in Figure 4. Some of the

Multiplication of complex numbers

$$(a_1, b_1) * (a_2, b_2) = (a_1 a_2 - b_1 b_2,\ a_1 b_2 + a_2 b_1)$$



Addition/subtraction of rational numbers

$$\frac{a}{p_1} \pm \frac{b}{p_2} = \frac{a p_1 \pm b p_2}{p_1 p_2}$$

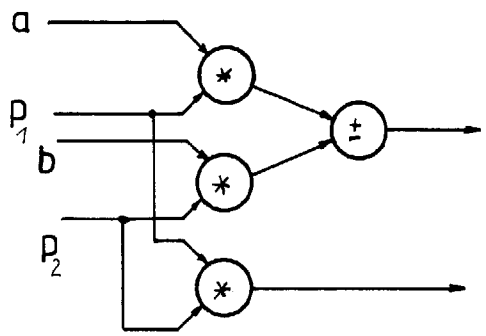


Figure 1: Dataflow graph examples.

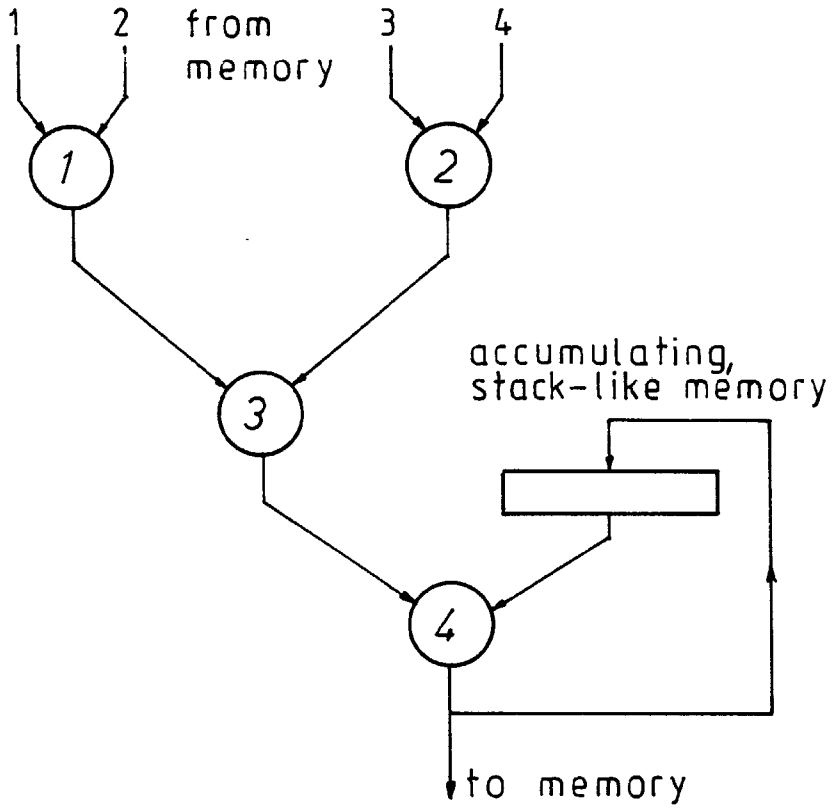
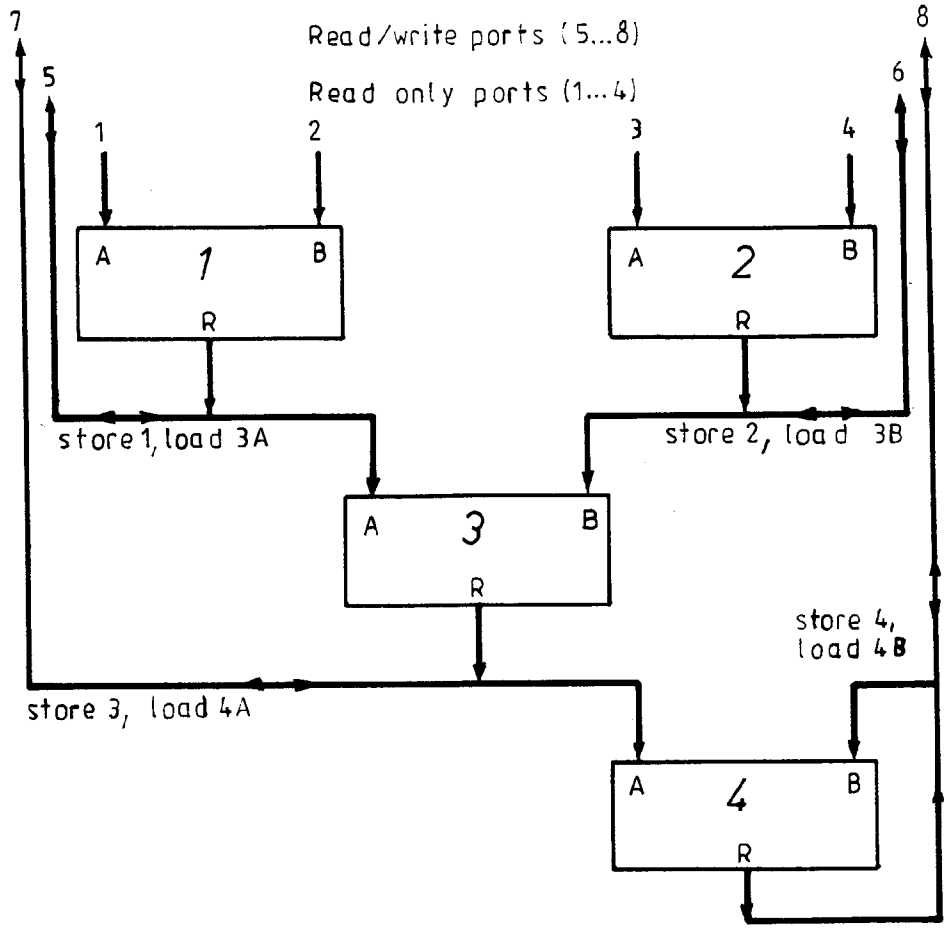


Figure 2: The proposed structure of four operation units.



Internal details of an operation unit

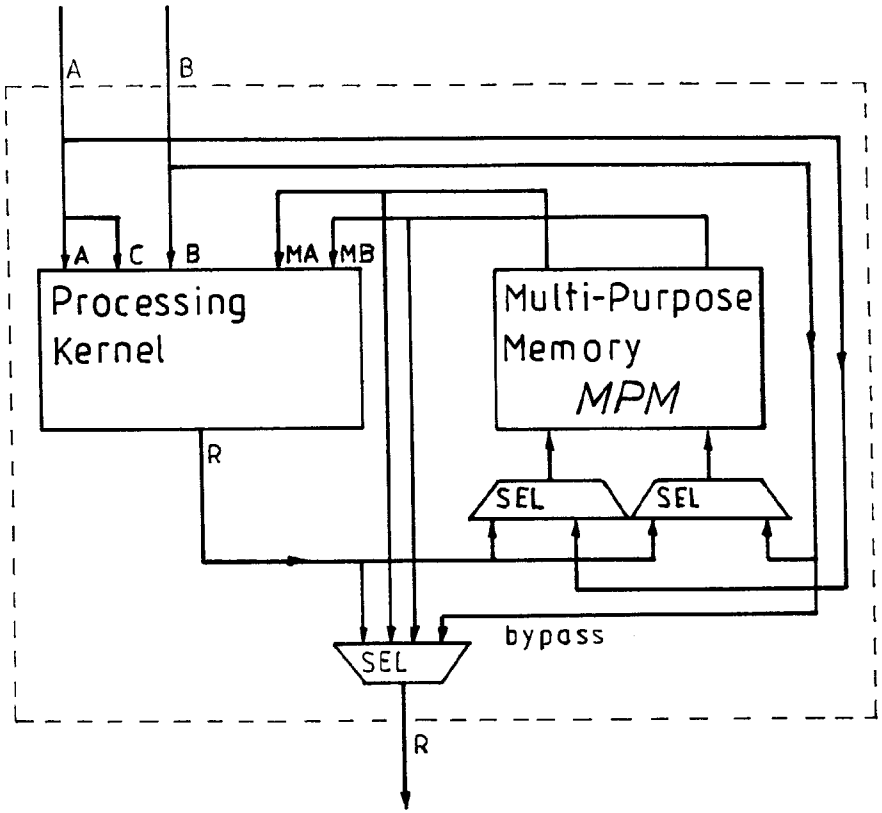


Figure 3: The proposed structure extended.

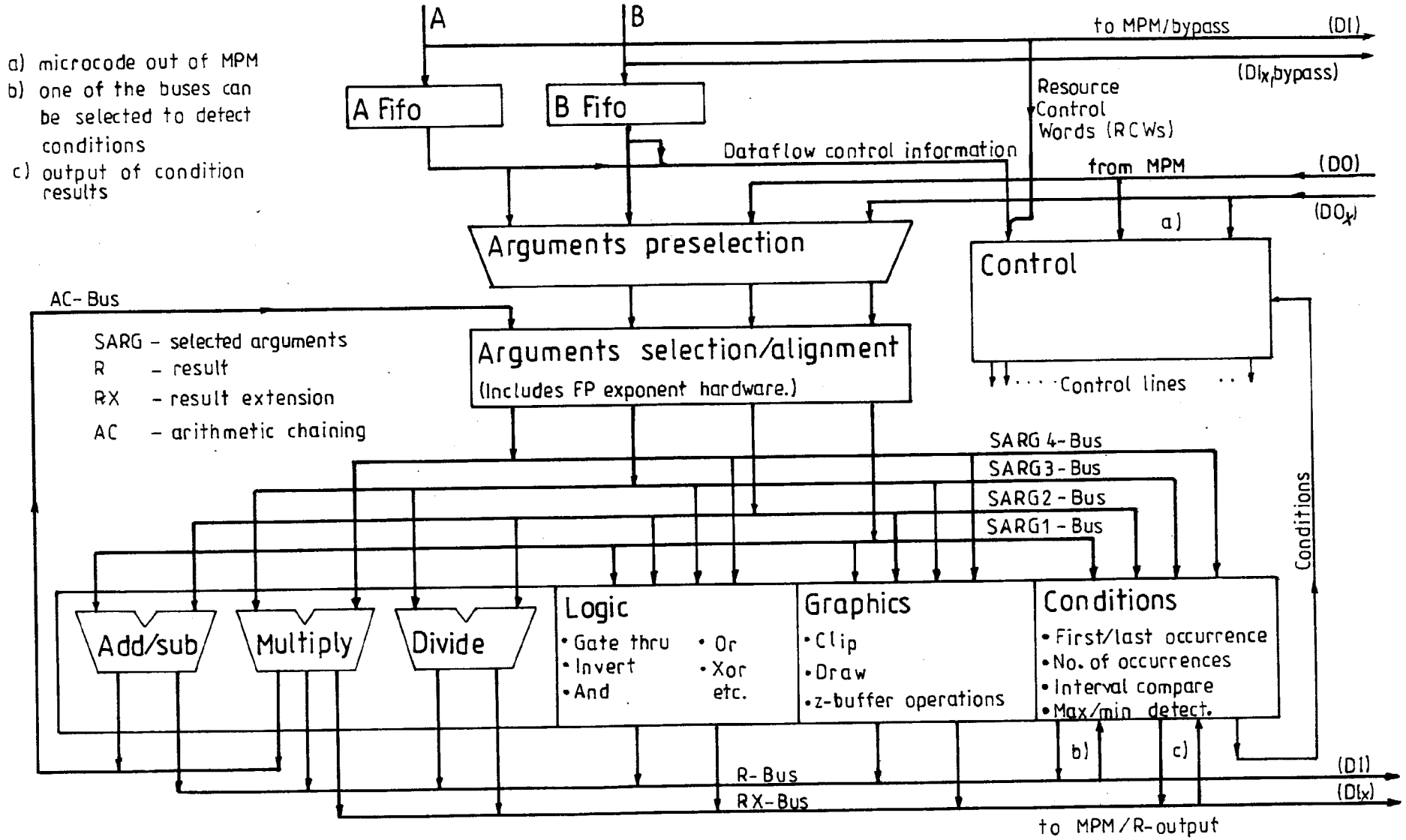


Figure 4: Internal structure of a processing kernel.

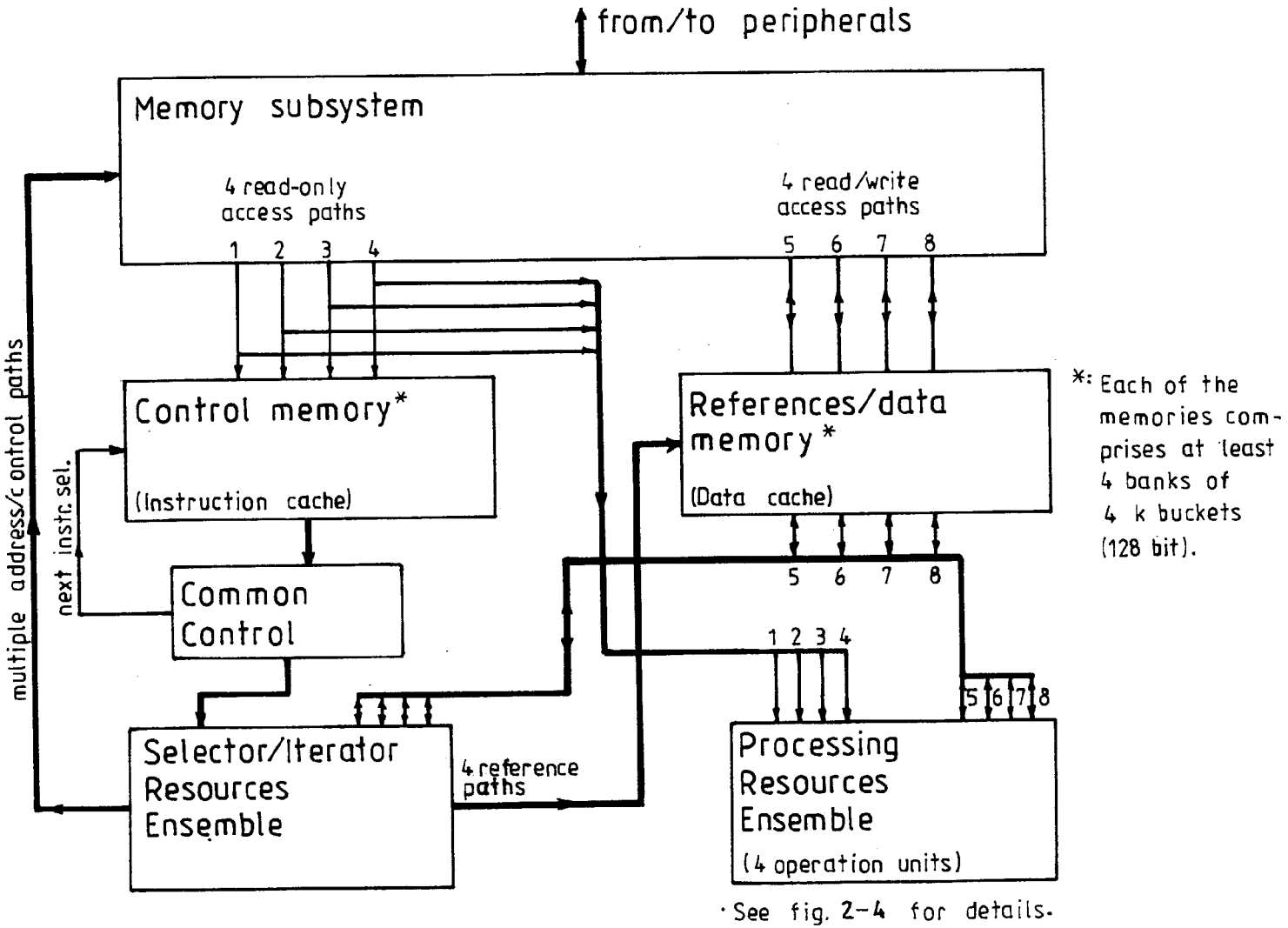


Figure 5: Processor structure.

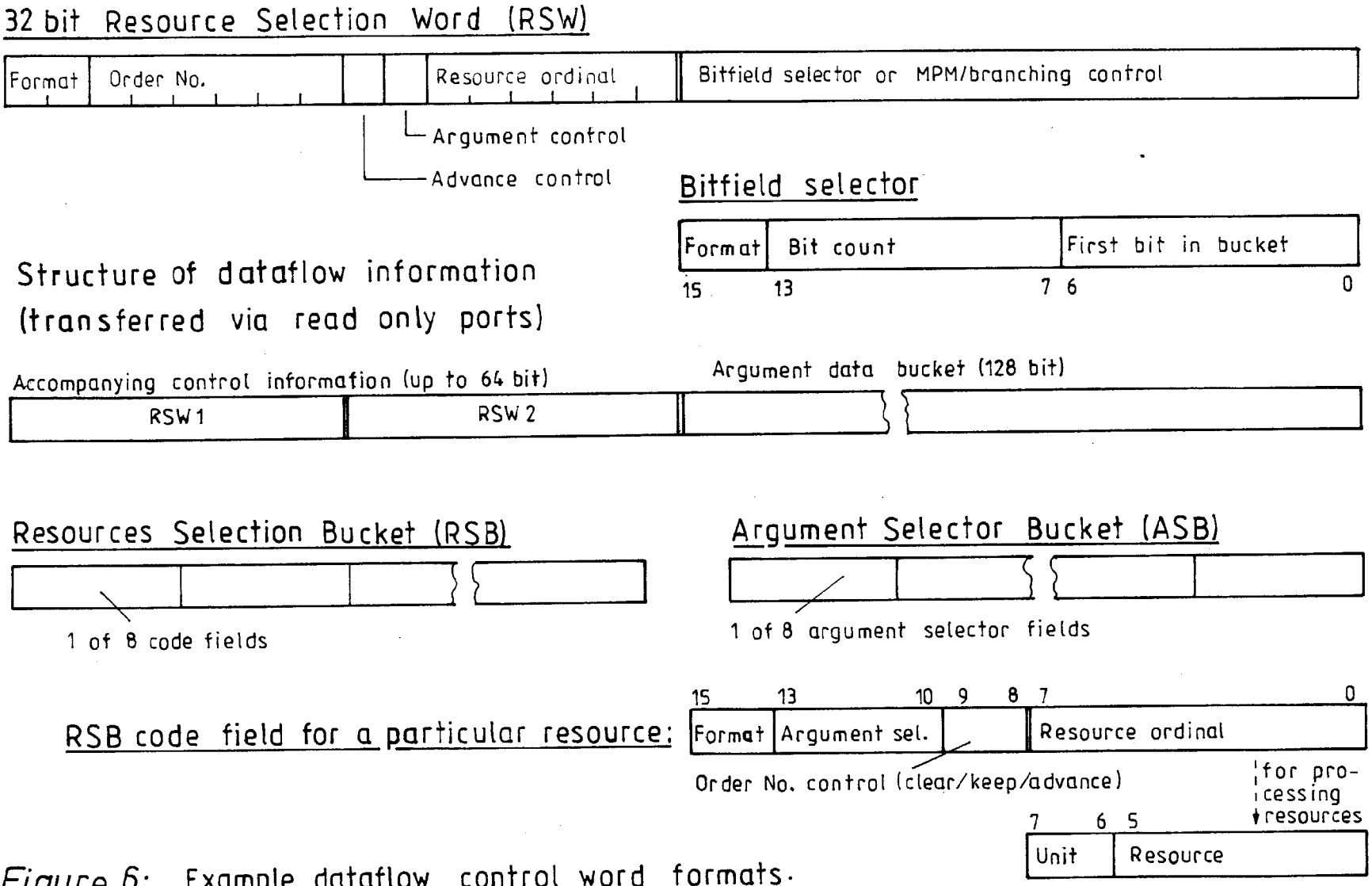


Figure 6: Example dataflow control word formats.