Dataflow Architecture

Kerem Irgan

Outline

- Sequential execution model
- Motivation
- Dataflow principles
- Dataflow graphs
- Dataflow languages
- Types of dataflow machines
- Drawbacks in dataflow computing
- Influences of dataflow in conventional architectures.

Sequential execution model

- Instructions of a program are executed in an implied **(sequential) order**.
  - In early days instructions were processed **one at a time.**
- Single separate storage structure
  - both instructions and data
- Programming languages also designed with sequential execution semantics.
- Control-flow architecture
- von Neuman architrecture
  - **bottleneck**; the limited bandwidth between the CPU and memory compared to the amount of memory.
  - **Parallelism** is the solution.

Motivation

- To exploit maximum parallelism inherent in a program.
- Minimize ordering constraints.
  - No ordering constraints other than **data and control dependences.**
- Overcome the performance limitations of the traditional sequential execution model.

Dataflow principles

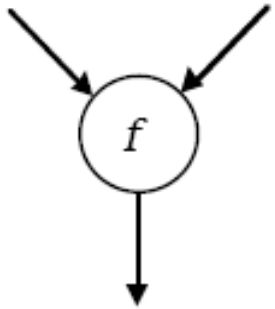
- **Asynchronous**
  - No program counter.
  - The execution of an instruction is based on the availability of its operands.
    - The synchronization of parallel activities is implicit in the dataflow model.
- No constraints on sequencing except the **data dependencies** in the program.
- Instruction fetch and execution are **data-driven**.
- A dataflow program is represented as a **directed graph**
  - nodes (or actors) represent instructions and arcs represent data dependencies between the nodes.

Dataflow principles

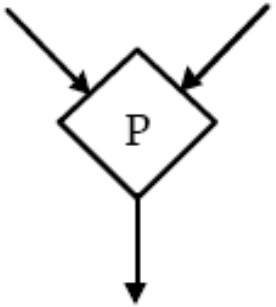
- "variables" and "memory updating" are non-existent.
  - Instead, objects (data structures or scalar values) are consumed by **an actor** (instruction) yielding a result object which is passed to the next actor(s).
- **Tokens**
  - The operands are conveyed from one node to another in data packets called tokens.
- Instead of a register file, they have a token store
  - Associates tokens to the appropriate instruction.
- **Fine-grain parallelism**.

Dataflow graphs

- The machine level language is represented by dataflow graphs.
- A data value is produced by an **Operator** as a result of some operation, f.
- A True or False control value is generated by a **Decider (a predicate)** depending on its input tokens.



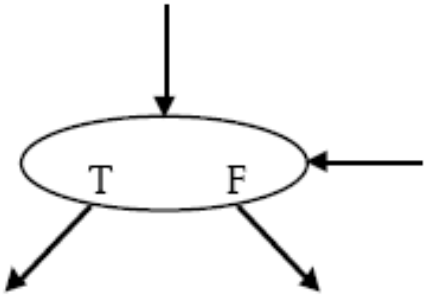
Operator



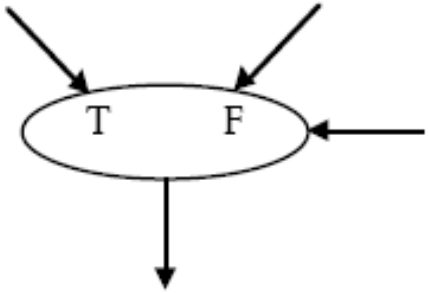
Predicate

Dataflow graphs

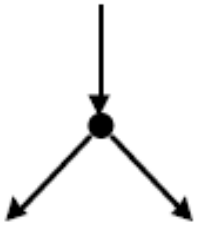
- Data values are directed by means of either a **Switch** or a **Merge** actor.



Switch



Merge



Copy

- a **Copy** is an identity operator which duplicates input tokens.

Dataflow languages

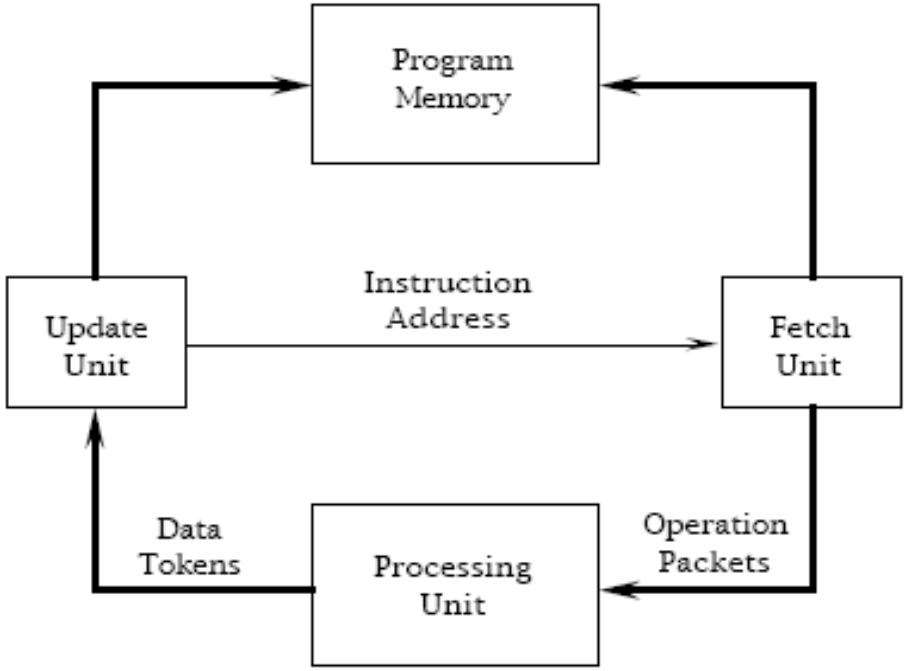
- There is a special need to provide a high-level language for dataflow computers
  - The dataflow graph is not an appropriate programming medium.
- Imperative class of languages
  - Mapping existing conventional languages to directed graphs using dataflow analysis often used in optimizing compilers.
    - Their sequential nature makes them inefficient
- Dataflow or applicative languages
  - Developing high-level dataflow languages which can express parallelism in a program more naturally and to facilitate close interaction between algorithm constructs and hardware structures.
  - Value Algorithmic Language (VAL)
  - Irvine Dataflow language (Id)
  - Stream and Iteration in a Single-Assignment Language (SISAL)

Types of dataflow machines

- Static dataflow machines
  - Not allow multiple instances of the same routines to be executed simultaneously
  - Conventional memory.
- Dynamic dataflow machines
  - Allow multiple instances
  - Tags
    - To distinguish between different instances of a node
    - A tag is associated with each token that identifies the context in which a particular token was generated.
    - An actor is
    - considered executable when its input arcs contain a set of tokens with identical tags.
  - Content-addressable memory (CAM)

Static dataflow model

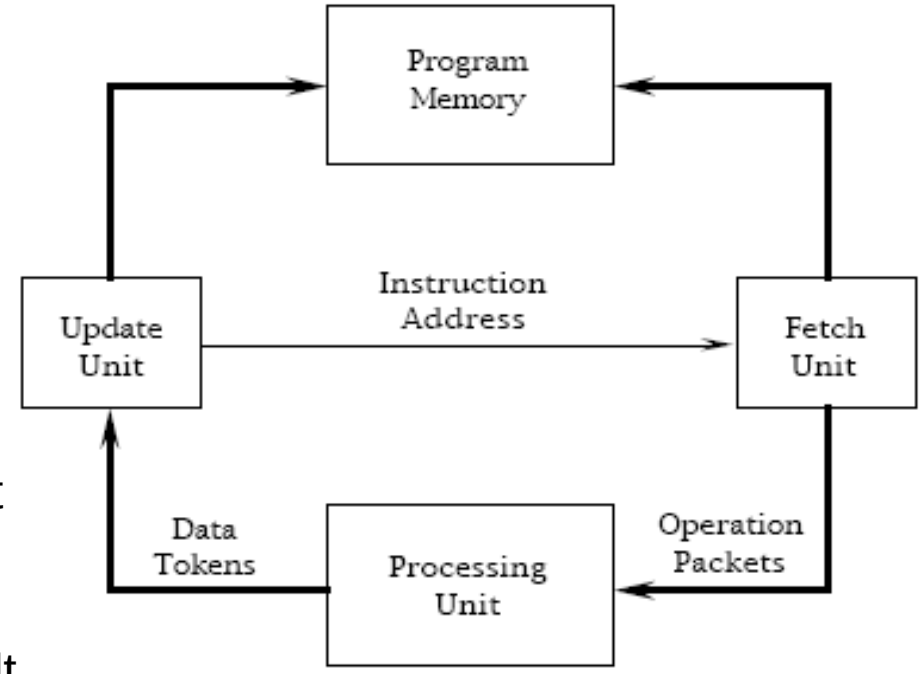
- Proposed by Dennis and his research group at MIT
- Program Memory contains
  - instruction templates
    - represents the nodes in a dataflow graph.
    - Contains an operation code, slots for the operands, and destination addresses,
      - presence bits (PBs).
        - » To determine the availability of the operands



Opcode	
PB	Operand 1
PB	Operand 2
Destination 1	
Destination 2	

Static dataflow model

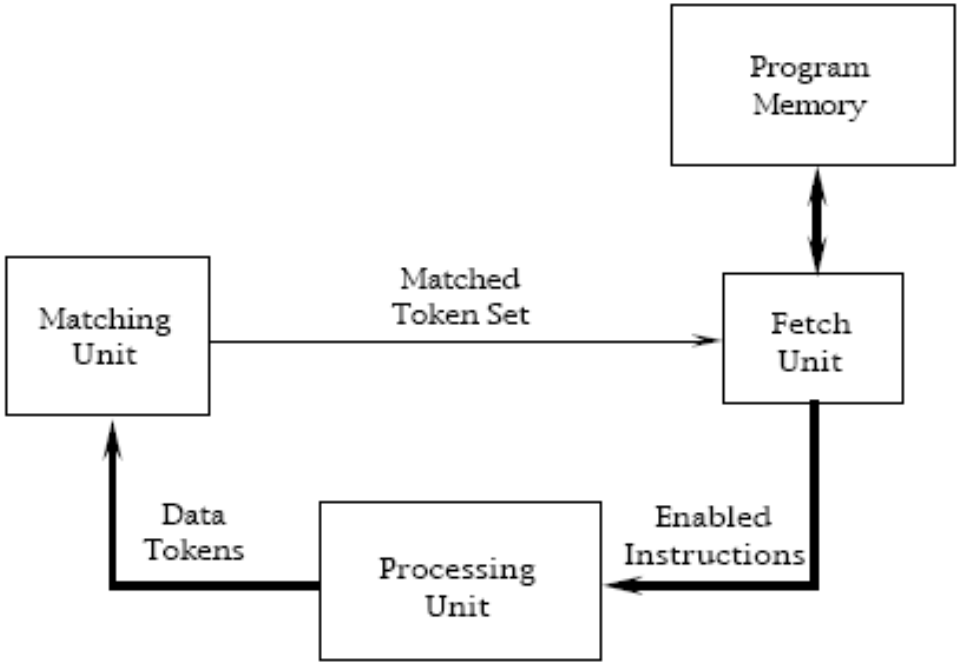
- The Update Unit
  - responsible for detecting the executability of instructions.
  - sends the address of the enabled instruction to the Fetch Unit.
- The Fetch Unit
  - fetches and sends a complete operation packet containing the corresponding opcode, data, and destination list to the Processing Unit
  - clears the presence bits.
- The Processing Unit
  - performs the operation, forms a result packets, and sends them to the Update Unit.
- The Update Unit
  - stores each result in the appropriate operand slot and checks the presence bits to determine whether the activity is enabled.



Dynamic dataflow model

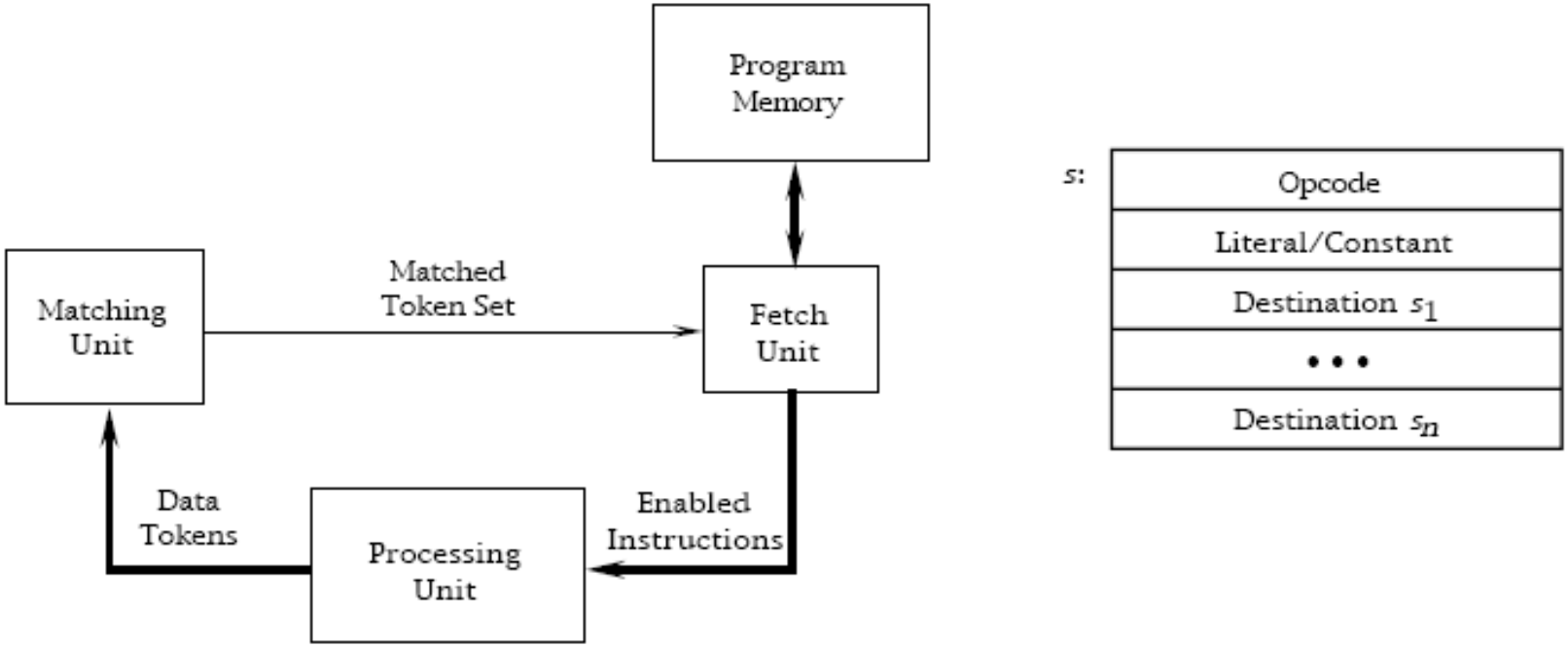
- Proposed by Arvind at MIT and by Gurd and Watson at the University of Manchester – Tagged-Token Dataflow Architecture

- The Matching Unit
  - a memory containing a pool of waiting tokens
  - Tokens are received by it.
  - brings together tokens with identical tags.
  - If a match exists, the matched token set is passed on to the Fetch Unit.
  - If no match is found, the token is stored in the Matching Unit to await a partner.



Dynamic dataflow model

- In the Fetch Unit
  - the tags of the token pair uniquely identify an instruction to be fetched from the Program Memory.
  - The instruction together with the token pair forms the enabled instruction and is sent to the Processing Unit.
- The Processing Unit
  - executes the enabled instructions and produces result tokens to be sent to the Matching Unit.



Drawbacks in dataflow computing

- Too fine-grained parallelism
  - Incurs more overhead in the execution of an instruction cycle
  - Poor performance in applications with low degree of parallelism.
- Inefficiency in representing and handling data structures (e.g., arrays of data).
  - Collections of many tokens
- Need for large token stores to handle all operations waiting for execution
- Means to schedule hundreds or thousands of operations that are ready to execute in a limited amount of available hardware.
- Difficulty of debugging
- The dataflow computing model has not been adopted in mainstream processor design.

Influences of dataflow in conventional architectures.

- Exploitation of fine- to medium- and large-grain parallelism.
  - conventional control-flow sequencing to be incorporated into the dataflow approach.
- Many innovations in ILP processors
- Dynamically scheduled superscalar processors
- Out-of-order execution
  - Execution window
    - Follows the sequential order.
    - Within the window, instructions are allowed to be completed in data dependency order.
  - CPUs dynamically tag the data dependencies of the code in the execution window
- CDC 6600 scoreboard
- Tomasulo`s algorithm
- Register renaming

References

- [1]Arvind and Nikhil, R. S., "Executing a Program on the MIT Tagged-Token Dataflow Architecture," Proc. Parallel Architectures and Languages, Europe, June 1987, pp. 1-29.
- [2]Readings in Computer Architecture, Mark D. Hill, Norman P. Jouppi and Gurindar S. Sohi, Eds., 1999, Morgan Koufmann publishers.
- [3]http://en.wikipedia.org/wiki/Dataflow_architecture
- [4] Lee, B. and Hurson, A. R. 1993. Issues in dataflow computing. Adv. in Comput. 37, 285--333.