Application of Superscalar Design Techniques to CISC Instruction Sets

- Implementation is complicated by:
  - Variable-length instructions
  - Multiple instruction formats
  - Traditional "condition code" indicators

- For well-established architectures,
  - Simple recompilation may be an unacceptable solution

- Superscalar hardware architecture must provide an acceptable performance improvement on existing object code
The Superscalar Microarchitecture of the MC68060

- Independent and decoupled four-stage pipelines
  - Instruction Fetch Pipeline (IFP)
    - IAG instruction address generation
    - IC instruction fetch cycle
    - IED instruction early decode
    - IB instruction buffer

- Operand Execution Pipelines (OEPs)
  - DS decode instructions + select components of operand address
  - AG operand address generation
  - OC operand fetch cycle
  - EX instruction execution
  - OEPs consist of two pipelines operating in lockstep
    - Primary OEP (pOEP) & secondary OEP (sOEP)

---

MC68060 Block Diagram

---

Hot Chips VI
August, 1994
Conceptual Model of the Operand Execution Pipeline

- Each execution pipeline consists of two "compute engines"
  - Operand address generation
  - Instruction execution
- Each compute engine follows the RISC model
  - Register file with dual read ports + ALU datapath

Resources = f(Base, Index, Address_result, A, B, Execute_result)

Conceptual Model of the OEPs (cont.)

- This model is the foundation of the 68060 microarchitecture
- Six key elements which support the model:
  - "Early decode" function within the IFP
  - Loading of fetched instruction stream into the FIFO Ibuffer
  - Superscalar dispatch algorithm located in the DS stage of the OEP
  - Result forwarding within the n-tuple of issued instructions
  - "Dynamic execution relocation" between the two compute engines
  - Optimized branch execution including 256-entry Branch Cache
"Early Decode" within the IFP

- Table lookup scheme
  - 16-bit opcode produces a decode longword
  - Identifies all the resources required by the instruction
    {memory, immediate, register} operands
  - Allows efficient evaluation of the dispatch algorithm

- Additional information to control the DS stage of the OEP
  - Instruction length
  - Operand type and address generation

- Fixed-length decode
  - Translates various instruction formats
  - Single definition which is used by the OEPs

Loading Fetched Instruction Stream

- Given a variable-length ISA and the ability to issue multiple instructions, how is the next set of instructions loaded into the execution pipeline?
  The 68060 solution is the use of a FIFO instruction buffer

- Packaged as machine instructions
  - Use instruction length from early decode
  - Each buffer entry is 1 instruction

- FIFO buffer implemented with three read ports
  - If current pOEP instruction is location \( i \) of FIFO,
    then FIFO reads at locations \( (i+1), (i+2), (i+3) \)
  - Allows the \( \{(i+1),(i+2)\} \) or \( \{(i+2),(i+3)\} \) pair to be sent to OEPs
Superscalar Dispatch Algorithm

- Evaluates a series of tests (DS stage)
  - Using early decode information, potential conflicts analyzed

- Data hazards between pOEP and sOEP
  - Inputs to sOEP must not be outputs from pOEP
  - If hazard is detected, then the sOEP instruction cannot be issued

- If all tests successful, both instructions issued to the AG stage

- If any test fails, only the pOEP instruction is issued

Result Forwarding within the Dispatched Group

- In 2-operand ISAs
  - Simple assignment instructions used extensively
  - Example:
    - `mov.l (d,A0),D0` # move memory-to-register
    - `add.l D1,D0` # register-to-register operation
    - Data hazard involving D0

- Avoid the detection of the data hazard
  - Rename the operand {destination = source} for simple assignments

- 2 heavily-used constructs benefit:
  - `mov.l <ea>,Rx`  
    - `<op> R`x  # Rx can be either source and/or destination
  - `mov.l <ea>,Rx`  
    - `<op>, D`x, <mem>
Dynamic Execution Relocation

- For most opcodes,
  - DS/AG engine used for address generation
  - OC/EX engine used for instruction execution

- Certain opcodes “executed” by DS/AG engine
  - Register renaming techniques
  - Results available with no pipeline stalls

- '060 adds third category
  - Opcodes which can be executed by either engine
  - Relocation from OC/EX to DS/AG
    - Allows more effective use of register renaming hardware
    - Minimize pipeline stalls
  - If no pending register writes to source operands at issue time,
    - Relocate the execution from OC/EX to DS/AG

Optimized Branch Execution

- 256-entry Branch Cache Unit within the IFP
  - Hardware table associating branch PC and target address
  - Organized as 4-way set-associative (4 x 64)
  - 4-state prediction model

- Instruction folding
  - Executes taken branches predicted correctly in 0 cycles

- Pipeline optimized for correct prediction
  - If mis-prediction, then
    - IFP & OEPs are aborted
    - New instruction stream established

- Execution Strategy for Non-Predicted Branches
  - Assumed branch resolution = f(branch direction)
    - If forward branch, then assumed to be not-taken
    - If backward, then the branch is assumed to be taken
MC68060 Summary

- Measured Performance
  - 1.2 CPI measured on range of desktop and embedded applications
  - 45-55% of instructions issued as pairs/triplets (existing 680X0 code)
  - 50-65% of instructions issued as pairs/triplets (targeted 68060 code)
  - \(68060-50 = 2.5 \times 3.5 \times 68040-25; \ 68060-66 = 2.1 \times 2.9 \times 68040-40\)

- Features
  - 2.5 million transistors, 0.5u TLM 3.3v process
  - Interfaces to 5v logic
  - Supports full, 1/2, and 1/4 speed buses
  - Worst case power dissipation of 3.9w at 50MHz
  - Sampling now; production ramp 4Q94

- Maintains hardware & software compatibility with the largest installed base of 32-bit embedded control microprocessors in the world