розв'язків, що одержуються ORTM алгоритмом, можуть бути використані довільні ітераційні підходи. Ефективний новий алгоритм на основі обмінів кластерів запропонований в [5].

1. Базилевич Р.П. Декомпозионные и топологические методы автоматизованого проектирования электронних устройств. Львів, 1981. 2. Ober U., Glesner M. Multiway netlist partitioning onto FPGA-based board architectures. Proc. European Design Automation Conference with EURO-VHDL (EURO-DAC'95), 1995. Pp.150-155. 3. Kuznar R., Brglez F., Kozminski K. Cost minimization of partitioning into multiple devices. IEEE/ACM 30th Design Automation Conference, 1993.5. Nan-Chi Chou, Lung-Rtien Liu, Chung-Kuan Cheng, Wei-Jin Dai, Rodney Lindelof. Local Ration Cut and Set Covering Partitioning for Huge Logic Emulation System. IEEE Trans. On CAD of Integrated Circuit and System, vol.14, No.9, Sept.1995, pp:1085-1092. 6. Базилевич Р.П., Рибак О.Г. Оптимізація пакування програмованих логічних матриць на основі оптимального згортання схем. В цьому віснику.

УДК 621.326

Березко Л.О., Ірисов О.Є.
ДУ "Львівська політехніка", кафедра ЕОМ

ЗАСОБИ ПІДВИЩЕННЯ ЕФЕКТИВНОСТІ ОБРОБКИ НЕЧІТКИХ ДАНИХ

Засоби обробки нечіткої, неповної та невизначеної інформації дають відчутні переваги в ситуаціях, коли вимагається отримати достатньо прийнятні результати, грунтуючись на нечітких вихідних даних. Впровадження цих засобів вимагає наявності в комп'ютерних системах відповідного апаратного та програмного забезпечення. Пропонується один з можливих варіантів реалізації підтримки нечіткої арифметики на RISC-процесорі.

Вступ. Бінарні операції з нечіткими даними можна реалізувати, застосовуючи інтервальну арифметику на рівневих множинах (α-зрізах) або min/max згортку [1,2,3]. Перший метод порівняно з другим дозволяє сильно скоротити кількість необхідних операцій та зменшити об'єм пам'яті для збереження операндів оскільки, при виконанні обчислень кількість відліків функції належності залишається постійною. Кількість α-зрізів, що описують нечітку величину, вибирається, виходячи з необхідної точності подання та допустимого часу обробки даних.

Хоча перший метод набагато швидший, ніж другий, він все одно забезпечує значно повільнішу обробку даних, ніж традиційна машинна арифметика. Для підвищення швидкості виконання нечітких операцій існують такі можливості: розробити сопроцесор (додатковий модуль інтервальної арифметики); додати до існуючого процесора невеликий набір інструкцій, що забезпечує інтервальну арифметику (в цьому випадку ставиться мета - кожна інтервальна операція повинна виконуватись з використанням не більше двох звичайних операцій) [4].

Модифікація RISC-процесора для підтримки нечіткої арифметики. Удосконалюємо RISC-процесор з простою архітектурою load-store, запропонованим [5], при побудові якого дотримувались основних принципів побудови RISC-процесорів.

Як відомо найбільш розповсюдженими операціями при роботі в нечіткому середовищі є такі: додавання, множення, віднімання та ділення інтервалів; інфінум та супремум; обчислення середини та ширини інтервалу; обчислення перетину інтервалів; заперечення інтервалу та інші, що досить легко реалізуються в одній машинній інструкції [1,3].

Оператори інтервальної арифметики визначаються таким чином [3]:

$$A+B=[a_1, a_2]+ [\cdot b_1, b_2]= \cdot[a_{1+}b_1, a_{2+}b_2]$$
$$A\text{-}B=[a_1, a_2] - [\cdot b_1, b_2]= \cdot[a_1 - b_1, a2\text{-}b2] \quad (1)$$
$$A\times B=[a_1, a_2] \times [\cdot b_1, b_2]= [\min (a_1\cdot b_1, a_1\cdot b_2, a_2\cdot b_1 ,a_2\cdot b_2),$$
$$\max (a_1\cdot b_1, a_1\cdot b_2, a_2\cdot b_1 ,a_2\cdot b_2)]$$
$$A/B=[a_1, a_2] / [\cdot b_1, b_2]= [\min (a_1/b_1, a_1/b_2, a_2/b_1 ,a_2/b_2),$$
$$\max (a_1/b_1, a_1/b_2, a_2/b_1 ,a_2/b_2)]$$

де a_1, a_2, b_1, b_2 – межі інтервалів відповідних операндів.

Таблиця 1

Множення інтервалів A та B

Інтервал		Результат	Знакові розряди	
A= $[a_1,a_2]$	B= $[b_1,b_2]$	A x B	S_1,S_2	S_3,S_4
> 0	> 0	$[a_1\cdot b_1, a_2\cdot b_2]$	00	00
> 0	< 0	$[a_2\cdot b_1, a_1\cdot b_2]$	00	11
< 0	> 0	$[a_1\cdot b_2, a_2\cdot b_1]$	11	00
< 0	< 0	$[a_2\cdot b_2, a_1\cdot b_1]$	11	11
перет. 0	> 0	$[a_1\cdot b_2, a_2\cdot b_2]$	10	00
перет. 0	< 0	$[a_2\cdot b_1, a_1\cdot b_1]$	10	11
> 0	перет. 0	$[a_2\cdot b_1, a_2\cdot b_2]$	00	10
< 0	перет. 0	$[a_1\cdot b_2, a_1\cdot b_1]$	11	10
перет. 0	перет. 0	$[\min (a_1\cdot b_2, a_2\cdot b_1),$ $\max (a_1\cdot b_1, a_2\cdot b_2)]$	10	10

Таблиця 2

Ділення інтервалів A та B

Інтервал		Результат	Знакові розряди	
A= $[a_1,a_2]$	B= $[b_1,b_2]$	A / B	S_1,S_2	S_3,S_4
> 0	> 0	[a/b, a/b]	00	00
> 0	< 0	[a/b, a/b]	00	11
< 0	> 0	[a/b, a/b]	11	00
< 0	< 0	[a/b, a/b]	11	11
перет. 0	> 0	[a/b, a/b]	10	00
перет. 0	< 0	[a/b, a/b]	10	11
> 0	перет. 0	ділення на 0	00	10
< 0	перет. 0	ділення на 0	11	10
перет. 0	перет. 0	ділення на 0	10	10

Інтервальні операції подібні до звичайних, але дають верхню та нижню межі величини, що опрацьовується, а не одне число. В деяких випадках (+,-) інтервальна операція зводиться до двох окремих звичайних операцій, в інших (X,1) – необхідно вирішити, які

конкретні величини треба прийняти за результат виконання операції. Особливості множення та ділення ітервалів наведені в таблицях 1 та 2, де S_1-S_4 – знакові розряди чисел, що зберігаються у форматі з рухомою комою.

Алгоритми виконання машинних інструкцій для операцій (+,-) будуються за виразами (1), а для операцій (x,/) – за таблицями 1,2.

На рис.1 показано модифіковану структуру конвеєра RISC-процесора для забезпечення найшвидшого виконання машинних інструкцій інтервальних операцій (+, -, x, /).

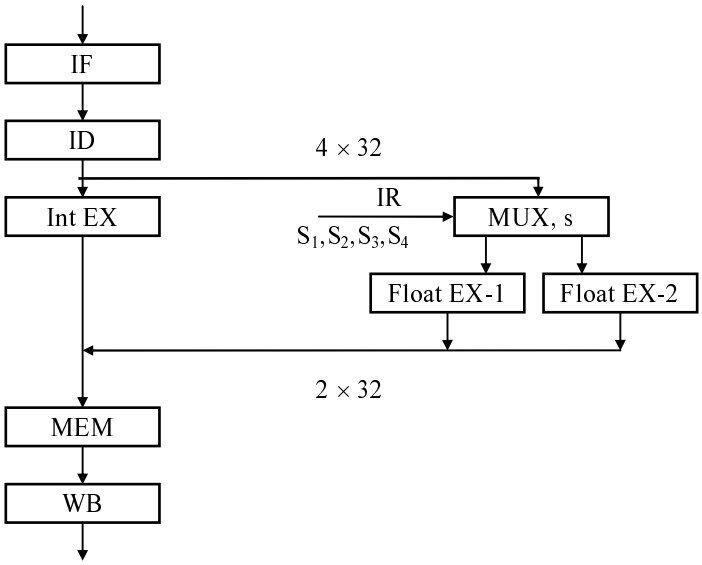

Рис.1. Модифікована структура конвеєра процесора

На рис.1 прийняті такі позначення: IF, ID, EX, MEM, WB – фази конвеєра; IF- вибірка команди; ID – декодування команди; EX – виконання команди; MEM – звертання до пам'яті даних; MUX s – блок мультиплексорів, що керується вмістимим регістра інструкцій IR, та керуючим словом S_1,S_2,S_3,S_4, що визначається за табл. 1,2. Блок int EX виконує цілочисельні операції (включаючи множення та ділення) та визначення ефективної адреси для звертання до пам'яті та переходів. Блоки Float EX-1 та Float EX-2 виконують додавання , віднімання, множення та ділення чисел з рухомою комою звичайної або подвійної точності подання.

Зазвичай RISC-процесор працює з машинними інструкціями різних типів [5]. Нові інструкції, які треба додати для реалізації інтервальних операцій, не повинні суттєво ускладнювати процесор. Тому обираємо для нових інструкцій формати R-типу (рис.2).

Orcode	RS1	RS2	RD	Function

Рис.2. Формат інструкцій R-типу

Множини вихідних інструкцій процесора доповнюється таким чином: IADDF Fd, Fa, Fb – додавання, ISUBF Fd, Fa, Fb – віднімання, IMULF Fd, Fa, Fb – множення, IDIVF Fd, Fa, Fb – ділення двох інтервалів, F – стандарт представленнях даних, I – ознака інтервальної інструкції.

Кожний інтервал зберігається у регістровому файлі рухомої коми (FPRs) у вигляді вмісту пари 32-бітних регістрів. В інструкції R – типу (рис. 2) поле RS1 адресує пару

регістрів FPRs для першого операнда, RS2 – відповідно для другого операнда, RD – пару регістрів для результату виконання інтервальної інструкції.

При використанні операторів нечіткої арифметики в складі мов високого рівня кожний з них транслюється в такі послідовності машинних інструкцій:

1. Блок інструкцій завантаження нечітких операндів (у вигляді рівневих множин) з пам'яті до регістрового файла рухомої коми FPRs. Можуть бути використані такі інструкції: LF Fd, Adr або LD Dr, Adr (Load single/double precision floating point).

2. Блок інструкцій інтервальної арифметики (в межах регістрової структури процесора): IADDF, ISUBF, IMULF, IDIVF.

3. Блок інструкцій запису до пам'яті нечіткого результату (у вигляді рівневих множин) з регістрового файла рухомої коми FPRs, : SF Adr, Fs або SD Adr, Fs (Store single/double precision floating point).

Висновки. Оператори обробки нечітких даних добре векторизуються. Блокування інструкцій дозволяє зменшити ймовірність зривів конвеєра RISC-процесора через залежність даних або умовні переходи. Як правило, серійні процесори мають невикористані можливості для суперскалярного або LVIW виконання інструкцій, завдяки чому їх функції можна розширити з найменшими витратами.

1. Дюбуа Д., Прад А. Теория возможностей. Приложения к представлению знаний в информатике. М.,1990. 2. Борисов А.М., Крумбер О.А., Федоров И.П. Принятие решений на основе нечетких моделей. Примери использования. Рига, 1990. 3. Scott Ferson, I Alin Cooper, Dwayne R.I. Moore, Robert C. Lee. Beyond Point Estimates. Risk Assessment Using Interval, Fuzzy and Probabilistic Arithmetic. Society for Risc Analisys, Washington, DC, December1997. 4. Nguen H.T., Koshelev M., Kreinovich V., Kosheleva O. Computational Complexity and Feasibility of Fuzzy Data Procesing: Why Fuzzy Numbers, Wich Fuzzy Numbers, Wich Operations with Fuzzy Numbers. Technical Report UTEP-CS-97-28, November 1997. 5. David A. Patterson, John I. Mennesy. Computer Architecture. Aquantitative Approach. San-Francisco, California, 1996.

УДК 621.382.323

Бєляєв С. М., Троценко В. В.
ДУ "Львівська політехніка", кафедра ЕОМ

АПАРАТНО-ОРІЄНТОВАНА КОНТЕКСТНА ТЕХНОЛОГІЯ СТИСКУ ІНТЕНСИВНИХ ВІДЕОПОТОКІВ В РЕАЛЬНОМУ ЧАСІ

Розглянуто програмну модель апаратно-орієнтованої технології стиску монохромних растрових відеопотоків модифікованого варіанта відомого [1] алгоритму контекстно-базованої технології стиску з метою підвищення коефіцієнта стиску і пристосування до швидкої апаратної реалізації (з інтенсивністю ~100 мегапікселів на секунду). Надано порівняння результатів стиску монохромних зображень з іншою контекстно-базованою технологією [2], яка претендує на проголошення стандартом JPEG-2000.