Мултиформен, мултиформатен цифров компаратор

Димитър С. Тянев, Димитър Г. Генов,
Веселин В. Стефанов, Мартин Д. Георгиев

Резюме: Аналитично е обоснована възможността за синтез на мултиформен и мултиформатен цифров компаратор, основаващ се на алгоритъма за без знаково сравнение на двоични числа. Синтезираният компаратор представлява нова оригинална логическа схема, която е способна да сравнява цели и дробни двоични числа със знак, както и двоично-десетични такива, представени в различни инверсни машинни кодове и кодирани в различни двоично-десетични кодове. Същата схема е в състояние да сравнява и числа, представени във форма с плаваща запетая според стандарта IEEE-754. Схемата е с висока степен на универсалност. Тя с успех може да замени в цифровите процесори традиционния алгоритъм за операция сравнение, реализирана чрез изваждане. Като случайна величина е изследвана латентността на компаратора. Определен е закона за разпределението й и неговите параметри.

Ключови думи: компаратор, цифров компаратор, признаци *LT, EQ, GT*, двоични числа, двоично-десетични числа, код 8421, 8421^{+3}, 2421, числа с плаваща запетая, стандарт IEEE-754, латентност, критерий на Пирсон, геометрично разпределение

Multi-form, Multi-format Digital Comparator

Dimitar S. Tyanev, Dimitar G. Genov,
Veselin V. Stefanov, Martin D. Georgiev

Abstract: There is analytical grounded possibility for synthesis of multi-form and multi-format digital comparator based on the algorithm for unsigned comparison of binary numbers. The synthesized comparator is a new original logic scheme capable of comparing integer and fractal signed binary numbers, as well as binary-coded decimal numbers presented in different inverse codes and various binary-decimal codes. Same scheme can compare also numbers presented in a floating point format according to IEEE-754 standard. The scheme has high degree of universality. It can replace successfully traditional algorithm for comparison used in digital processors which is implemented by subtraction. The comparator latency has been explored as a random quantity. The distribution law and its parameters have been defined as well.

Key words: comparator; digital comparator; LT, EQ, GT signs; binary numbers; binary-decimal numbers; 8421, 8421^{+1}, 2421 codes; floating point numbers; IEEE-754 standard; latency; Pierson criteria; geometric distribution.

Въведение

Операция сравнение *Compare(x,y),* на две числа x и y е често срещана в алгоритмите. За изпълнение на тази операция в цифровите процесори се включват съответните машинни команди [1], [2], [6]. Операнди x и y на тези команди са числа със знак и могат да бъдат представени в различни форми. Установяването на отношението между двете числа

$$x * y, \qquad * \in \{<, =, >\} \tag{1}$$

цели да формира и присвои съответните истинни стойности на признаците *LT (Less Than), EQ (equal)* и *GT (Greater Than)* според следните правила:

$$if \ \ (x < y) \ \ then \ \ LT = 1, \ EQ = 0, \ GT = 0. \tag{2}$$

$$if \ \ (x = y) \ \ then \ \ LT = 0, \ EQ = 1, \ GT = 0. \tag{3}$$

$$if \ \ (x > y) \ \ then \ \ LT = 0, \ EQ = 0, \ GT = 1. \tag{4}$$

Операция сравнение се изпълнява в АЛУ на цифровия процесор чрез привеждане на отношението (1) към еквивалентното му, след представяне на операндите в инверсен машинен код (обратен или допълнителен)

$$x * y \quad \rightarrow \quad (x - y) * 0 \ . \tag{5}$$

Като се отчете още, че числата могат да бъдат представени в различни бройни системи, в различни машинни кодове, а така също и в различни форми и формати, с голяма увереност може да се твърди, че такова изпълнение на операция сравнение е възможно най-бавното.

В същото време съществуват хардуерни структури, които съдържат в себе си като логически възли комбинационни компаратори, както и съществуват интегрални схеми, съдържащи единствено цифрови компаратори, позволяващи каскадно удължаване на разрядността на числата [3], [4], [5]. В тези случаи комбинационният компаратор е синтезиран въз основа на алгоритъма за без знаково сравнение на две двоични комбинации, при което основното му достойнство е високото бързодействие.

Описаното тук изследване е посветено на един принципно нов подход за изграждане на многофункционални цифрови компаратори, способни на високоскоростно изпълнение на операция сравнение, чрез хардуерна реализация на операция (1). Многофункционалността се изразява в заложената мултиформеност и мултиформатност, позволяваща сравнението на:

1. Цели и дробни двоични числа без знак, както и такива със знак, представени в прав, в обратен или в допълнителен кодове, както и в техните модификации;

2. Цели и дробни двоично-десетични числа без знак, както и такива със знак, кодирани в код 8421, 8421^{+3}, 4221, 2421, представени в прав, в обратен или в допълнителен кодове, както и в техните модификации;

3. Числа, представени във форма с плаваща запетая според стандарта IEEE-754.

Възможностите на такава логическа схема я определят като универсална или още като инвариантна към фóрмата и формата за представяне на сравняваните числа.

Основни съображения относно представянето на числата

Какъвто и да е видът на представените числа, всеки от горните случаи се разпада на 4 подслучая, в зависимост от знаците им. Практически това е единственото общо нещо между изброените по-горе 3 случая. Налага се изводът, че независимо от всички останали характеристики, основавайки се на алгебрическия смисъл на числата, сравнението на знаците еднозначно определя стойностите на признаците (2), (3) и (4), когато те са различни както следва

$$\begin{aligned} &if \quad ((x > 0) \cap (y < 0)) \quad then \quad LT = 0,\ EQ = 0,\ GT = 1\ ; \\ &if \quad ((x < 0) \cap (y > 0)) \quad then \quad LT = 1,\ EQ = 0,\ GT = 0. \end{aligned} \tag{6}$$

Техническата реализация на сравнението на знаковите битове е облекчено от факта, че във всички форми, в които се представят числата, знаковият бит е най-левият (най-старшият) в разрядната мрежа. Веднага следва да се отбележи, че в резултат на избраната кодировка за знака на числата (0 за положителни и 1 за отрицателни), алгоритъмът на без знаковото сравнение, при сравнение на знаковите битове, ще определя признаците (6) на отношение (1) неправилно, т.е. инверсно, възприемайки знаковия бит по стойност, а не по смисъл.

В случаите, когато знаците на числата са еднакви, алгоритъмът на без знаковото сравнение ще определя признаците (2), (3) и (4) след сравнение на останалите битове на комбинациите, представящи числата. Това налага разглеждането на случаите по отделно, тъй като останалите битове са функция на машинния код.

Изказаните по-горе съображения обаче не са в сила за без знаковите числа, тъй като при тях старшият бит трябва да се възприема по стойност. Това налага предварително инициализиране на този случай, тъй като е изключение от го горните правила.

А) Двоични числа

Когато числата имат знак, за тяхното машинно представяне се използва обратен или допълнителен код, за получаването на които е необходим прав код. За *n*-битова разрядна мрежа с дясно фиксирана запетая [6] тези кодове се дефинират така

$$[x]_{OK} = \begin{cases} x, & if \ \ x \geq 0 ; \\ 2^n - 1 - |x|, & if \ \ x \leq 0 . \end{cases} \qquad [x]_{ДК} = \begin{cases} x, & if \ \ x \geq 0 ; \\ 2^n - |x|, & if \ \ x < 0 . \end{cases} \tag{7}$$

където чрез *n* е означена дължината на мултиформатната разрядна мрежа. Различните формати са възможни благодарение на правилото за знаково разширение на числата.

Аналогична зависимост дефинира инверсните кодове и за двоичните числа с ляво фиксирана запетая.

В първия случай на (7), когато и двете числа са положителни, отношението на двата инверсни кода $[x]_{ДК} * [y]_{ДК}$ (или $[x]_{OK} * [y]_{OK}$) е еквивалентно на отношение (1) и може да се изчисли чрез алгоритъма на без знаковото сравнение.

Във втория случай на (7), когато и двете цели числа са отрицателни, отношението на двата инверсни кода се изразява съответно чрез отношението на числата

$$\begin{aligned} &OK: \ (2^n - 1 - |x|) \ * \ (2^n - 1 - |y|) ; \\ &ДК: \ (2^n - |x|) \ * \ (2^n - |y|) , \end{aligned} \tag{8}$$

което очевидно също е еквивалентно на (1) и за двата кода, тъй като по-малкият модул ще генерира по-голяма по стойност разлика.

Налага се обобщението, че алгоритъмът на без знаковото сравнение може да се използва за сравняване както на числа без знак, така и на числа със знак, представени в инверсни машинни кодове, включително и модифицирани такива, които имат по 2 знакови бита. Това важи и за числата с ляво фиксирана запетая, както и за числата, структурирани с цяла и дробна част. С други думи положението на запетаята е без значение, тъй като тук се имат предвид само числа, представени в позиционни бройни системи.

Когато сравняваните числа са представени в прав или в обратен код, се налага предварително автоматично разпознаване и игнориране на знаковите нули (+0, -0), едно различие, което има място в тези два кода. Това лесно може да бъде осигурено с хардуерни средства. Специален коментар заслужава сравнението на числа представени в прав код, който за цели числа се дефинира както следва

$$[x]_{ПК} = \begin{cases} x, & if \ \ x \geq 0 ; \\ 2^{n-1} + |x|, & if \ \ x \leq 0 . \end{cases} \tag{9}$$

В първия случай на (9), когато и двете числа са положителни, отношението на двата кода $[x]_{ПК} * [y]_{ПК}$ е еквивалентно на отношение (1) и може да се изчисли чрез алгоритъма на без знаковото сравнение, тъй като то всъщност представлява отношението $|x| * |y|$.

Във втория случай на (9), когато и двете цели числа са отрицателни, отношението на двата кода се изразява съответно чрез отношението на числата

$$(2^{n-1} + |x|) \ * \ (2^{n-1} + |y|) \ . \tag{10}$$

което очевидно се решава чрез отношението $|x| * |y|$. В резултат на това признаците се определят вярно, само когато модулите са равни. При различни модули в този случай, стойностите на признаците *LT* и *GT*, се определят неправилно – техните правилни стойности са инверсни. Изключенията, което генерират числата със знак обаче могат да бъдат дешифрирани и всички възможни случаи да бъдат обобщени от следната логическа функция

$$\begin{array}{l} if\ \left(\overline{NS} \cap (x \neq y) \cap I\right)\ then\quad LT_{out} = \overline{LT},\ GT_{out} = \overline{GT},\ EQ_{out} = 0\ : \\ \qquad\qquad\quad else\quad LT_{out} = LT\ ,\ GT_{out} = GT\ ,\ EQ_{out} = EQ\ . \end{array}\ , \tag{11}$$

където с *I* (*inversion*) е означена логическата функция

$$I = RC \cap (x_{n-1} \cup y_{n-1})\ \cup\ \overline{RC} \cap (x_{n-1} \oplus y_{n-1})\ , \tag{12}$$

а с индекс *out* са означени крайните изходни за компаратора стойности на признаците.

Както се вижда от (11) и (12), за идентификация на всички изключения, логическата схема на компаратора е функция от два допълнителни сигнала:

1. Сигнал *NS* (*Not Signed numbers*), декларищ, че числата са без знак;
2. Сигнал *RC* (*Right Code*), декларищ, че числата са представени в прав код.

Б) Двоично-десетични числа

Десетичните числа се представят като наредена последователност от кодовите комбинации на своите десетични цифри – (8421), (8421^{+3}), (2421) и др. Най-широко се прилага първият от посочените. Машинните кодове на цели числа със знак се дефинират така

$$[x]_{OK} = \begin{cases} x, & if\ \ x \geq 0\ ; \\ q^n - 1 - |x|, & if\ \ x \leq 0\ . \end{cases} \qquad [x]_{ДК} = \begin{cases} x, & if\ \ x \geq 0\ ; \\ q^n - |x|, & if\ \ x < 0\ . \end{cases}\ , \tag{13}$$

където с q е означена основата на бройната система, в която е представено числото, като в частност тук се има предвид q=10. С n отново е изразена дължината на мултиформатната разрядна мрежа. За всеки 2/10-чен код е известна [6], [7] алгоритмична схема за постигане на зависимостта (13), която изхожда от техния прав код. Както и за двоичните числа, знаковите нули в прав или в обратен код следва предварително да бъдат дешифрирани и уеднаквени.

Прилагането на алгоритъма на без знаково сравнение за проверка на отношението $[x]_{ДК} * [y]_{ДК}$ (или $[x]_{OK} * [y]_{OK}$) на 2/10-чни числа напълно съответства на зависимостите (2), (3), (4) и (6). При изпълнение на този циклически алгоритъм, на сравнение се подлагат две цифри в текущия *i*-ти десетичен разряд $dx_i * dy_i$. Тъй като тези цифри са представени с двоични комбинации, това отношение се проверява бит по бит по силата на алгоритъма на двоичното без знаково сравнение. От това следва, че отношението за тези две цифри ще бъде вярно, ако за техния 2/10-чен код е в сила свойството монотонност [6]. Това свойство е в сила и за кодовите комбинации на десетичните цифри в инверсни машинни кодове на числата със знак. От тук следва окончателният извод, че алгоритъма на двоичното без знаково сравнение може да се приложи и върху различно кодирани 2/10-чни числа. Алгоритъмът е инвариантен към положението на запетаята на числата.

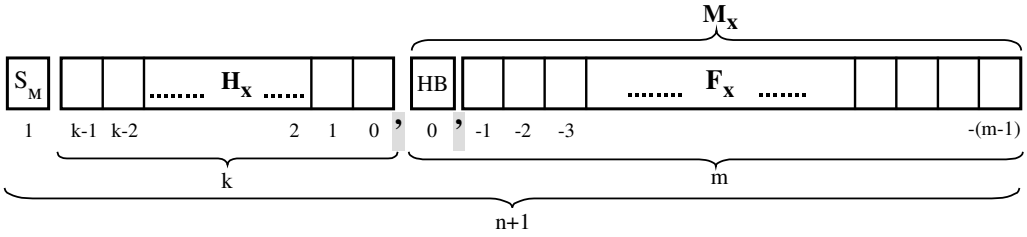
Правият код на 2/10-чните числа се дефинира както следва

$$[x]_{ПК} = \begin{cases} x, & if\ \ x \geq 0\ ; \\ q^{n-1} + |x|, & if\ \ x \leq 0\ . \end{cases} \tag{14}$$

Всичко казано преди това за сравнението на двоичните числа в прав код, е в сила и за 2/10-чните числа в прав код. Ето защо логическата функция (9) е в сила и за тези числа. Следва, че тя е инвариантна към основата на бройната система, в която са представени числата.

В) Числа във форма с плаваща запетая

Разглежда се структура на разрядната мрежа според стандарта IEEE-754 с техника на скрития бит [6], [8], имаща вида, представен на фигура 1.



Фиг. 1. Структура на разрядната мрежа

в която означенията са:

S_M – знаков бит на мантисата (на числото);

H_X – характеристика. За характеристиката са в сила следните зависимости:

$$H_x = p_x + (D-1) = (p_x - 1) + D\,, \qquad D = 2^k\,; \tag{15}$$

НВ – скрит бит;

M_X – нормализирана мантиса. Има вида: $M_X = 1,F_X$;

F_X – фракция.

Мултиформатността се осигурява от променящите се дължини k (8, 11 или 15 бита) и m (23, 52 или 64 бита)., дефинирани от стандарта.

Мантисата е число със знак и се представя в прав код. Това означава, че отношението (1) може да бъде определено чрез алгоритъма на без знаковото сравняване. Това отношение обаче следва да бъде съобразено с характеристиките на числата. Характеристиките се възприемат като числа без знак и по стойност също могат да се сравнят чрез алгоритъма на без знаковото сравняване. Тяхното отношение обаче трябва да се приложи върху стойностите на порядъците, които са числа със знак.

Според (15) отношението $H_x * H_y$ се разглежда във вида

$$(p_x - 1) + 2^k \;\; * \;\; (p_y - 1) + 2^k\;, \tag{16}$$

който може да бъде сведен до

$$p_x \;\; * \;\; p_y\;. \tag{17}$$

Последното отношение обаче не винаги е еквивалентно на отношението на характеристиките, тъй като порядъците са числа със знак, а те могат да се случват в 4 различни комбинации.

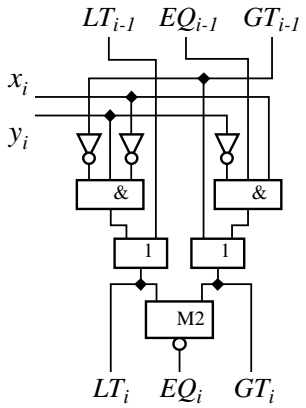
Комбинациите от различни знаци съответстват на правилно определено чрез сравняване на характеристиките отношение, тъй като отрицателният порядък води до операция изваждане (виж (15)), което силно отдалечава по стойност съответната характеристика от другата. В случай, че знаците са положителни, отношението на характеристиките е в пълно съответствие с отношение (17). В последния случай, когато знаците на порядъците са отрицателни, в резултат на операция изваждане, отношение (17) също в съответствие с отношение (16). В случай на две отрицателни числа, по-голямо е числото с по-малкия порядък.

Сравнението на мантисите, като числа със знак, представени в прав код, напълно съответства на казаното вече за числата с фиксирана запетая. Тъй като при тези условия сравнението генерира същите изключения, следва да се приложи логиката на функция (11).

Логическа схема на компаратора

Логическата схема на компаратора е синтезирана в съответствие с алгоритъма за без знаковото сравнение и представлява последователност от еднобитови компаратори. Принципната логическа схема за всички средни i-ти разряди $i = \overline{(n-2),1}$ е представена на фиг. 2. Тя е синтезирана според методиката, представена в [5] и има следната логика

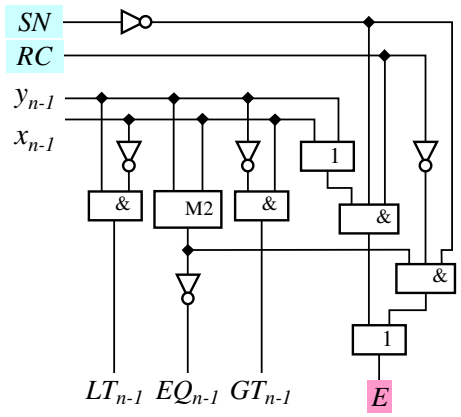
$$
\begin{aligned}
LT_i &= LT_{i+1} \cup (\overline{GT_{i+1}} \cap \overline{x_i} \cap y_i);\\
GT_i &= GT_{i+1} \cup (EQ_{i+1} \cap x_i \cap \overline{y_i});\\
EQ_i &= \overline{LT_i \oplus GT_i}\ .
\end{aligned}
\tag{18}
$$



Фиг. 2. Логическа схема на *i*-тия бит

Логическата схема на компаратора в най-старшия (*n-1*)-ви разряд (фиг. 3) е синтезирана като функция и от инициализиращите сигнали *I* и *RC*. По този начин, когато числата генерират описаните изключения и изходните признаци трябва да се инвертират, в този разряд се генерира разрешението *E* (*Enable*), което се използва в най-младшия (нулевия) разряд на компаратора. Логиката на схемата е следната

$$
\begin{aligned}
&LT_{n-1} = \overline{x_{n-1}} \cap y_{n-1};\\
&GT_{n-1} = x_{n-1} \cap \overline{y_{n-1}};\\
&EQ_{n-1} = \overline{x_{n-1} \oplus y_{n-1}};\\
&E = \overline{SN} \cap \left[RC \cap (x_{n-1} \cup y_{n-1}) \ \cup\ \overline{RC} \cap (x_{n-1} \oplus y_{n-1})\right].
\end{aligned}
\tag{19}
$$



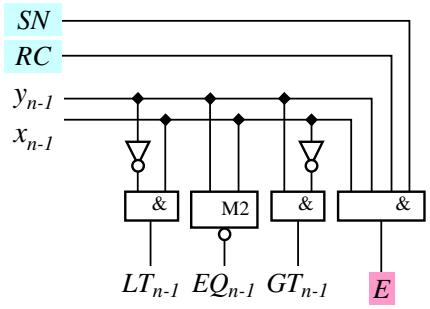
Фиг. 3. Логическа схема на старшия бит

Изключенията, които генерират числата със знак в прав и в допълнителен код, могат да бъдат игнорирани, ако знаковите битове на числата бъдат инвертирани преди сравнение. С други думи, предварителната инверсия на знаковите битове заменя смисъла на кодовите цифри на знаците с числена стойност, водеща до правилно определяне на признаците. Така при сравнение на числа, представени в инверсни машинни кодове, не се генерират изключения. Изключение се генерира единствено при две отрицателни числа, представени в прав код. Логиката за дешифриране на това изключение е представена от функция (11), в която обаче следва да се използва следната логическа функция

$$I = RC \cap x_{n-1} \cap y_{n-1} \ . \tag{20}$$

Логическата схема на компаратора в най-старшия (*n-1*)-ви разряд (фиг. 4) е синтезирана аналогично, но при отчитане на зависимостта (20) за инициализиращия сигнал *I*. Логиката на схемата в този втори вариант е следната

$$\begin{aligned} LT_{n-1} &= x_{n-1} \cap \overline{y_{n-1}}; \\ GT_{n-1} &= \overline{x_{n-1} \cap y_{n-1}}; \\ EQ_{n-1} &= \overline{x_{n-1} \oplus y_{n-1}}; \\ E &= \overline{SN} \cap \left[RC \cap x_{n-1} \cap y_{n-1}\right]. \end{aligned} \tag{21}$$



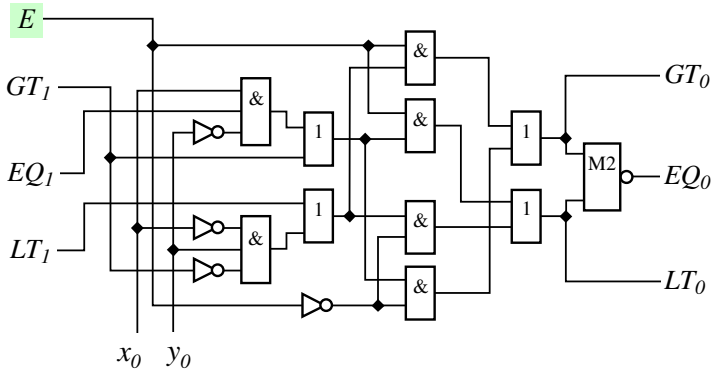
Фиг. 4. Логическа схема на старшия бит (с предварителна инверсия на знака)

Очевиден е изводът, че апаратната реализация на (21) е по-икономична в сравнение с тази на (19).

Логическата схема на компаратора в най-младшия *0*-лев разряд е представена на фиг. 5. Както се вижда, изходните стойности на признаците са функция от разрешението *E*, чиято логика е следната

$$\begin{aligned} LT_0 &= \overline{E} \cap \left(LT_1 \ \cup \ (\overline{GT_1} \cap \overline{x_0} \cap y_0)\right) \ \cup \ E \cap \left(GT_1 \ \cup \ (EQ_1 \cap x_0 \cap \overline{y_0})\right); \\ GT_0 &= \overline{E} \cap \left(GT_1 \ \cup \ (EQ_1 \cap x_0 \cap \overline{y_0})\right) \ \cup \ E \cap \left(LT_1 \ \cup \ (\overline{GT_1} \cap \overline{x_0} \cap y_0)\right); \\ EQ_0 &= \overline{LT_0 \oplus GT_0} \quad . \end{aligned} \tag{22}$$

Разрешението влияе на окончателните резултати само в случай, че числата са във форма с плаваща запетая и са отрицателни и не са напълно еднакви.



Фиг. 5. Логическа схема на младшия бит

Статистически подход за идентификация на закона на разпределение на латентността на компаратора

Времето за превключване на логическата схема на компаратора е основен технически параметър, който го характеризира. В случая, времето за превключване е пропорционално на дължината на верижката, образувана от срещнатите в посока към младшите разряди

последователно еднаквите старши цифри в двете числа. При среща на различни цифри в текущия бит, признаците се определят еднозначно и останалите до най-младшия бит цифри, не се нуждаят от сравнение. Така, разглеждана като случайна величина, времето за превключване на логическата схема, т.е. на латентността й, се нуждае от специално изследване. За целта беше организиран и проведен числен експеримент чрез програмен модел на компаратора, създаден на програмния език C++. В модела на сравнение са подлагани двойки числа, генерирани като псевдослучайни с помощта на функция *boost::random* от библиотека "Boost" [9]. Сравняваните числа са генерирани в 32-битовия диапазон на разрядната мрежа, като получените дължини за латентността на компаратора при всяко сравнение, са натрупани в извадки с обем 1000 и 5000 сравнения.

Статистическото изследване на получените извадки има за цел да намери закона за разпределение на случайната величина, както и неговите параметри. Последните са необходими за оценка на производителността на компаратора, когато е използван в други изчислителни структури.

Като модел на верижката от фактически извършени сравнения с резултат равно (*EQ*) се използва величината (*w-1*). Всеки бит, в който очакването да бъде прекратена верижката от последователни съвпадения не се реализира, се оценява с вероятност (*1-p*). Тогава вероятността, верижката да не бъде прекъсната след (*w-1*)-тия бит е $(1\text{-p})^{(w-1)}$. С отчитане на условието, че в бит *w* събитието прекъсване ще се състои с вероятност [*1*-(*1-p*)]=*p*, то законът за разпределение има вида

$$f = p.(1-p)^{(w-1)} \; ; \quad w = \overline{1,n}, \quad p \geq 0 \; . \tag{23}$$

Видът на тази функция дава основание да бъде издигната нулева хипотеза H_0 за статистическата апроксимация на латентността на компаратора със закона за геометрично разпределение. За определяне на параметъра *p* в (23) и неговата адекватност към експерименталните данни е използван критерия на Пирсон [11]

$$\chi^2 = \sum_{i=1}^{k} \frac{(n_i - f_i.N)^2}{f_i.N} \; , \tag{24}$$

където: *i* - е номер на интервала; *N* - обем на статистическата извадка;

n_i - брой попадения в *i*-тия интервал. Случайната величина *w* има 32 възможни стойности, които могат да се разпределят в 32 интервала на 32 битовата разрядна мрежа;

f_i - честота на попадения по теоретичния закон.

Таблица 1. За извадка с обем 1000 сравнения

w	n_i	$p_i=n_i/1000$	f_i	fn_i
1	473	0,473	0,493	493
2	260	0,260	0,25	250
3	142	0,142	0,1267	127
4	75	0,075	0,0642	64
5	28	0,028	0,0326	33
6	22	0,022	0,0165	17

Числените резултати от проведеното изследване, получени в средата MATLAB, са представени в таблици 1 и 2, където са използвани следните означения:

w - номер на интервала, съответстващ на номера на изходящия бит на верижката;

p_i - относителна честота на попаденията в *i*-тия интервал;

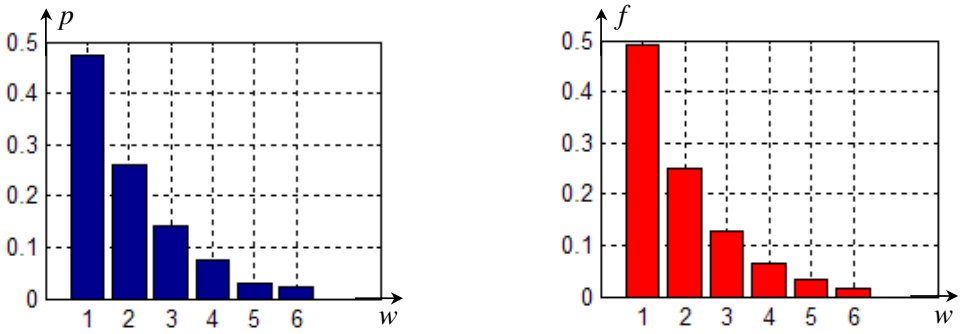
fn_i - брой попадения, определени по теоретичния закон.

За всяка извадка са определени максималната и минимална стойност на случайната величина, математическото очакване M, както и нейната дисперсия S^2.

В първата извадка максималната дължина на латентността е 13 бита, а минималната 1 бит. Математическото очакване има стойност M=2,035, а дисперсията – S^2=2,0658.

Вероятността, събитието да се състои, е p=0,493. Числената стойност на критерия $\chi^2 = 7{,}1018$ е по-малка от теоретичната $\chi_T{}^2(0{,}05;5) = 11{,}07$, определена при ниво на значимост v=0,05 и степени на свобода k=max(w)-1=5, откъдето следва, че представените данни не противоречат на хипотезата за геометрично разпределение.

На фиг. 6 са представени хистограмите на честотата на попадение p_i (вляво) и теоретичното разпределение f (вдясно).



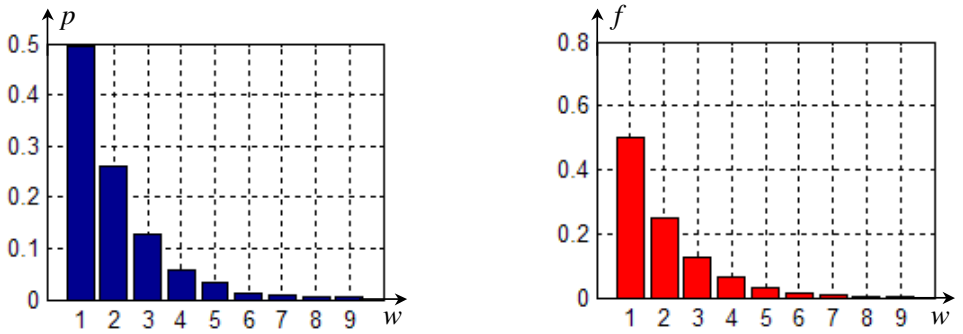
Фиг. 6. Разпределение на латентността в извадка с обем 1000 сравнения

Във втората извадка максималната дължина на латентността е 17 бита, а минималната 1 бит. Математическото очакване има стойност M=1,9998, а дисперсията – S^2=1,9918. Вероятността, събитието да се състои, е p=0,5004. Числената стойност на критерия $\chi^2 = 7{,}146$ е по-малка от теоретичната $\chi_T{}^2(0{,}05;8) = 15{,}507$, определена при ниво на значимост v=0,05 и степени на свобода k=max(w)-1=8, откъдето следва, че представените данни не противоречат на хипотезата за геометрично разпределение.

Таблица 2. За извадка с обем 5000 сравнения

w	n_i	p_i=n_i/1000	f_i	fn_i
1	2467	0,4934	0,5004	2502
2	1296	0,2592	0,25	1250
3	635	0,127	0,1249	624
4	290	0,058	0,0624	312
5	162	0,0324	0,0312	156
6	64	0,0128	0,0156	78
7	47	0,0094	0,0078	39
8	20	0,004	0,0039	19
9	19	0,0038	0,0019	10

На фиг. 7 са представени хистограмите на честотата на попадение p_i (вляво) и теоретичното разпределение f (вдясно).



Фиг. 7. Разпределение на латентността в извадка с обем 5000 сравнения

Заключение

Синтезът на цифров компаратор с множеството различни характеристики се оказа възможен. Синтезираната логическа схема прилага последователно сравнение, но може да бъде преструктурирана чрез подходи като описания в [7].

Схемата е мултиформена, защото е инвариантна към запетаята на числа, представени в позиционни бройни системи. Тя е инвариантна още към различните машинни кодове (прав, обратен, допълнителен и техните модификации).

Схемата е мултиформатна, тъй като е инвариантна към дължината на разрядната мрежа, в която са представени числата, което се дължи на свойствата на инверсните машинни кодове.

Схемата е инвариантна още към кода за представяне на десетичните цифри, стига той да притежава свойството монотонност, което е показано тук за няколко от най-прилаганите.

Схемата проявява своите универсални възможности и върху числа представени във форма с плаваща запетая според стандарта IEEE-754. Доказано е, че синтезираната схема успешно се справя в три от четирите случая на комбиниране на знаците на такива числа. Изключението, което може да се яви при числа с отрицателни знаци, успешно се дешифрира, при това с минимални апаратни средства, не влияещи на латентността й. Синтезирани са два различни варианта за корекция на признаците. Мултиформатността на схемата е в сила и върху числата с плаваща запетая благодарение на адитивната функционалност, според която е дефинирана характеристиката на числата.

Времето за превключване на цифровия компаратор е функция от самите числа, ето защо то е променлива величина, която има случаен характер и геометрично разпределение. Статистическите резултати налагат обобщението, че в рамките на допустимото ниво на значимост за критерия на Пирсон, очакваното време за превключване на компаратора е в границите на 1/3 от неговата дължина.

Литература

[1]. Hennessy, J.L., D.A. Patterson. *Computer Architecture. A Quantitative Approach*. 4-ed., ISBN: 1-55860-724-2, Morgan Kaufman Publishers, Amsterdam, 2007.

[2]. Tanenbaum, A.S. *Structured Computer Organization.* 5-ed, Prentice Hall, ISBN 0-13-148521-0, New Jersey, 2006.

[3]. Wakerly, J.F. *Digital Design. Principles & Practices*. 3-ed, Prentice Hall, ISBN 0-13-769191-2, New Jersey, 2000.

[4]. Lala, P.K. *Principles of modern digital design*. John Wiley & Sons, ISBN 978-0-470-07296-7, New Jersey, 2007.

[5]. Enoch O. Hwang. *Digital Logic and Microprocessor Design with VHDL*. La Sierra University, Riverside, ISBN: 0-534-46593-5, 2005.

[6]. Тянев, Д.С. *Организация на компютъра. Том 1,* ISBN: 978-954-20-0412-7. Технически Университет - Варна, 2008.

[7]. Stine, J.E., M.J. Schulte. *A Combined Two's Complement and Floating-Point Comparator*, IEEE International Symposium on Circuits and Systems, 2005. Kobe. ISBN: 0-7803-8834-8, vol. 1, pp. 89-92.

[8] Standards Committee of the IEEE Computer Society, *IEEE Standard 754 for Binary Floating Point Arithmetic*. IEEE Press, August 1985.

[9]. C++ source libraries: http://www.boost.org/ .

[10]. BCD 2421-code. http://www.tyanev.com/home.php?lang=bg&mid=18&mod=1&b=7&s=403

[11]. Генов. Д.Г. *Моделиране и оптимизация на производствени процеси,* ISBN: 954-20-0143-6. Технически Университет - Варна, 2000.