разработка аппаратных платформ

УДК 519.688, 004.315

Sergey I. Salishev,
Roman E. Shein

NOVEL ALGORITHMS FOR CONTINUOUS-FLOW MIX-RADIX IN-PLACE MULTI-BANK RAM-BASED FFT[1]

Abstract

A method of implementing in-place continuous-flow mix-radix FFT on multibank memory with additional constraints is investigated. Using this method four novel FFT architectures are proposed. Parallel butterflies in small radix stage allow substantial speed-up for mixed radix FFT. The single-port memory architecture provides in-place strategy for libraries without dual-port memory, effectively reducing memory requirement by 50%. Self-sorting architecture allows using overlapped I/O for natural order FFT reducing initiation interval up to 30%. A combined approach is also proposed.

Keywords: FFT, in-place, continuous-flow, mixed-radix, self-sorting.

1. INTRODUCTION

Fast Fourier Transform (FFT) is used by multiple communication applications such as 802.11, 802.16 and their modifications. FFT processor performance is crucial for overall performance of these applications. A common approach to FFT processor architecture is an in-place memory-based one. Use of this approach guarantees that for each butterfly or group of butterflies both inputs and results are stored in the same memory locations, so for FFT sampled at N points a dual-port multibank memory storing N complex words can be used. Since memory dominates area and power of memory-based FFT processor, such minimization of memory size is crucial for the processor to be efficient.

One butterfly should be initiated every clock to maximize throughput for given butterfly size. To do so each wing of the butterfly should read and write non-conflicting memory ports. This requires conflict-free bank assignment.

Johnson [1] suggested an in-place addressing strategy and architecture that allows launch of one butterfly per clock for pure-radix FFT.

Hsiao, Chen and Lee [2] suggested an in-place addressing strategy and architecture for arbitrary mix-radix FFT launching one butterfly per clock.

Jo and Sunwoo [3] suggested an in-place addressing strategy and architecture for radix 2/4 FFT launching 2 radix 2 butterflies in radix 2 stage that utilizes 2 single-port N-sized memories.

Xilinx LogiCORE IP FFT [4] and Altera MegaCORE [5] use radix 2/4 memory-based burst I/O architecture with both bit-reversed/digit-reversed and natural order of inputs and outputs.

[1] Logical circuit implementation is pending Russian and US patents.

Xilinx LogiCORE IP FFT uses a method for radix 2/4 FFT launching 2 radix 2 butterflies in parallel, based on the RTL evaluation.

A flexible approach that generalizes results presented in above works is proposed. Using this flexible approach a few novel FFT processor architectures with improved performance are suggested.

For FFT sampled at N points, where $N = r \cdot R^{n-1}$, $2r \le R$ and R, r are radixes of butterflies used in the FFT, the simple approach proposed by Johnson [1] is to calculate radix r butterflies using radix R butterfly engine with redundant inputs set to zero. It is possible to significantly speed up the calculation by using radix R butterfly engine to calculate multiple radix r butterflies simultaneously. Such an improvement is proposed by Jo and Sunwoo [3] for $r = 2$, $R = 4$. We generalize the approach for any r, R such that R is divisible by r.

A variation of the method that facilitates use of single-port memories instead of dual-port memories while still launching multiple small-radix butterflies per clock is suggested. Replacing dual-port memories with single-port memories while still granting the performance improvements allows area improvements as well. Notice that use of this architecture allows creation of in-place FFT processors with in-place addressing on libraries without dual-port memories, while usually two single-port memories of size N are used. Therefore, use of this approach allows 50 % reduction in memory area for libraries without dual-port memories.

Use of either decimation in time (DIT) or decimation in frequency (DIF) decompositions leads to input or output having different order. Therefore if the order is important, a digit reverse must be performed before or after the FFT calculation. It requires additional shuffling stage with complex memory access pattern. So it is desirable to blend the shuffle operations with computations.

Hegland [6] proposed generalized self-sorting in-place FFT decomposition summarizing previous works on the topic by mixing in-place transposition stages with computation stages. He used symmetric two-sided decomposition combining DIT and DIF. A similar method using asymmetric stage arrangement is proposed in this paper. Due to asymmetric property it is mapped to DIT(DIF) FFT processor architecture with only change in memory generation preserving all the benefits stated above. Using of this method allows up to 30 % reduction in clocks of initiation interval of burst I/O normal order FFT radix-4 of length 1024 compared to Xilinx LogiCORE IP FFT.

In section 2 a convenient notation for FFT addressing is given. A general formula for bank assignment is proposed. In section 3 a FFT processor architecture utilizing dual-port memories is proposed. In section 4 a FFT processor architecture utilizing self-sorting addressing is proposed. In section 5 a FFT processor architecture utilizing single-port memories is proposed. In section 6 a combination of self-sorting and single-port approaches is considered. Proofs of theorems are omitted in the sections and can be found in the appendix.

2. NOTATION

In this section a convenient notation for FFT addressing is proposed. For simplicity introduce the following string substitution: $[d]_{i,i+j} = d_i, d_{i+1}, ..., d_{i+j}$, $[d]_{i+j,i} = d_{i+j}, d_{i+j-1}, ..., d_i$.

If $d_0, ..., d_s$ are, accordingly, $r_0, ..., r_s$ radix digits, then let $[d_s, ..., d_0]$ be a mix-radix number constructed by concatenating the digits. If any d_i is a radix 1 digit, define $[d_s, ..., d_{i+1}, d_i, d_{i-1}, ..., d_0] = [d_s, ..., d_{i+1}, d_{i-1}, ..., d_0]$. This boundary case appears when proofs for mix-radix are applied for pure-radix case. More formally, $[d_s, ..., d_0] = d_0 + d_1 \cdot r_0 + d_2 \cdot r_0 \cdot r_1 + ... + d_s \cdot r_0 \cdot r_1 \cdot r_2 \cdot ... \cdot r_{s-1}$.

Consider a *FFT* sampled at $N = r_0 \cdot ... \cdot r_{n-1}$ points decomposed into radix $r_0, ..., r_{n-1}$ stages, where $r_i \le r_{i+1}$. Note: such ordering for radixes is not mandatory, but is used to simplify proofs and explanations.

Calculation of *FFT* can be viewed as two nested loops: outer loop iterating over stages and inner loop iterating over butterflies (or butterfly groups for stages with multiple butterflies executed simultaneously) within one stage.

Let $FFT(k_{n-1}, ..., k_0)$ stand for result of the *FFT* on input numbered $[k_{n-1}, ..., k_0]$ ($k_i \in 0..r_i$, k_0 being the least significant digit). Define a radix butterfly operation

$$B_s(f_{r-1}, ..., f_0) = \sum_{k=0}^{r-1} f_k \cdot (W_r)^{s \cdot k}. \quad (1)$$

Here $W_r = e^{-\frac{2\pi i}{r}}$ are complex roots of one.

Let $F_{c+1}([d]_{0,n-c-2}, k_c, [k]_{c-1,0})$ be stage c output numbered $[[d]_{0,n-c-2}, k_c, [k]_{c-1,0}]$, where k_i are already processed digits and d_i are digits that are yet to be processed, $k_i < r_i$, $d_i < r_{n-i-1}$, $F_0(d_0, ..., d_{n-1})$ are input sample points. Then $FFT(k_{n-1}, ..., k_0) = F_n(k_{n-1}, ..., k_0)$.

For DIT decomposition the FFT stage formula is

$$F_{c+1}\left([d]_{0,n-c-2}, k_c, [k]_{c-1,0}\right) = B_{k_c}(w_0, ..., w_{r_{n-c-1}-1}), \quad (2)$$

$$w_u = W_{r_{n-1} \cdot ... \cdot r_{n-c-1}}^{u \cdot [[k]_{c-1,0}]} \cdot F_c\left([d]_{0,n-c-2}, u, [k]_{c-1,0}\right). \quad (3)$$

For DIF decomposition the stage formula is is

$$F_{c+1}\left([d]_{0,n-c-2}, k_c, [k]_{c-1,0}\right) = W_{r_0 \cdot ... \cdot r_{n-c-1}}^{k_c \cdot [[d]_{0,n-c-2}]} \cdot B_{k_c}(\overline{w}_0, ..., \overline{w}_{r_{n-c-1}}), \quad (4)$$

$$\overline{w}_u = F_c\left([d]_{0,n-c-2}, u, [k]_{c-1,0}\right). \quad (5)$$

DIT decompositions leads to digit reverse order of the input points, and DIF decomposition leads to digit reverse order of output points.

Notice that formulae for DIF and DIT differ only in whether multiplication by twiddle factors is performed before or after the butterfly operation. Choice of decomposition type is insignificant further, so for convenience suppose DIF is used.

A radix r_c butterfly in stage utilizes inputs with numbers $[k_{n-1}, ..., k_{c+1}, k_c, k_{c-1}, ..., k_0]$, where k_c varies from 0 to $r_c - 1$. Then the butterfly can be numbered by $[k_{n-1}, ..., k_{c+1}, k_{c-1}, ..., k_0]$.

The approach adopted in this paper implies use of memory split into r_{n-1} banks in order to allow pipelining butterfly execution: each radix r butterfly operation requires r memory reads and r memory writes. Define bank and address assignments depending only on sample point numbers (it is convenient to use such in-place notation even for self-sorting FFT, which is not actually in-place). Let $m(k_{n-1}, ..., k_0)$ be bank assignment and $a(k_{n-1}, ..., k_0)$ be address assignment within the bank for number $[k_{n-1}, ..., k_0]$. In this paper any correct address assignment may be used, for simplicity suppose everywhere $a(k_{n-1}, ..., k_0) = [k_{n-2}, ..., k_0]$. Let

$$I_c([d_{n-1}, ..., d_{c+1}, d_{c-1}, ..., d_0], d) = [d_{n-1}, ..., d_{c+1}, d, d_{c-1}, ..., d_0], \quad (6)$$

This notation can be used to conveniently separate butterfly number from wing number:

$$m(k_{n-1}, ..., k_1, k_0) = m(I_c([k_{n-1}, ..., k_{c+1}, k_{c-1}, k_1], k_0)). \quad (7)$$

While there is a dependency between subsequent stages, butterflies within one stage are independent from each other and therefore can be calculated in arbitrary order. Suppose q_c butterflies are run simultaneously in stage c. Stage $n-1$ obviously has only one butterfly run simultaneously, because only r_{n-1} memory banks are available. For any stage c that runs q_c butterflies per clock the inner loop iterates over butterfly groups numbered $[k_{n-1}, ..., k_{c+2}, \overline{k_{c+1}}, k_{c-1}, ..., k_0]$, where $k_i < r_i$, $\overline{k_{c+1}} < \left\lceil \frac{r_{c+1}}{q_c} \right\rceil$.

Let $T_c(k_{n-1},...,k_{c+2},\overline{k_{c+1}},\overline{\overline{k_{c+1}}},k_{c-1},...,k_0)$, where $k_i < r_i$, $\overline{k_{c+1}} < \left\lceil \frac{r_{c+1}}{q_c} \right\rceil$, $\overline{\overline{k_{c+1}}} < q_c$, $[\overline{k_{c+1}},\overline{\overline{k_{c+1}}}] < r_{c+1}$ be number of $\overline{\overline{k_{c+1}}}$ 'th butterfly executed in $[k_{n-1},...,k_{c+2},\overline{k_{c+1}},k_{c-1},...,k_0]$ 'th iteration of loop iterating over butterfly groups in stage c. Essentially k_{c+1} is split into $[\overline{k_{c+1}},\overline{\overline{k_{c+1}}}]$ and $\overline{k_{c+1}}$ is used as a part of butterfly group number, while $\overline{\overline{k_{c+1}}}$ is used to enumerate butterflies within the group.

The trivial traverse order for all stages is

$$T_c(k_{n-1},...,k_{c+2},\overline{k_{c+1}},\overline{\overline{k_{c+1}}},k_{c-1},...,k_0) = [k_{n-1},...,k_{c+2},\overline{k_{c+1}},\overline{\overline{k_{c+1}}},k_{c-1},...,k_0]. \quad (8)$$

Let $M_c([k_{n-1},...,k_0])$ be memory bank used in iteration k of butterfly loop in stage c. If q radix r_c butterflies are run in parallel, M_c can be obtained as

$$M_c([k_{n-1},...,k_0]) = m(I_c(T_c(k_{n-1},...,k_{c+2},\overline{k_{c+1}},\overline{\overline{k_{c+1}}},k_{c-1},...,k_0),k_c)). \quad (9)$$

The hypothesis we will exploit is that the following bank assignment is conflict free and allows multiple butterflies per clock in small radix stages for mixed radix FFT on dual-port memories with trivial traverse order

$$m(k_{n-1},...,k_0) = \left(\sum_{i=0}^{n-1} g_i \cdot k_i\right) mod\ r_{n-1}. \quad (10)$$

Here g_i are some constants depending on radixes chosen for stages.

This bank assignment generalizes bank assignments proposed by Johnson [1], Hsiao, Chen and Lee [2] and Jo and Sunwoo [3]. It is also used as base for self-sorting and single-port memory architectures.

3. FFT PROCESSOR UTILIZING DUAL-PORT MEMORIES

Consider a FFT sampled at $N = r \cdot R^{n-1}$ points using radix r, $R = r \cdot q$ butterfly operations, i. e. $r_0 = r, r_1 = ... = r_{n-1} = R$. It can be calculated utilizing a FFT processor with the following architec-ture similar to one presented in [1]. The corresponding block structure is shown on Fig. 1. It consists of Address Generation Unit (AGU), Random Access memory (RAM), Switchable Interconnect (IC), Butterfly Processing Unit (PU), and twiddle memory.

The key feature of the architecture is AGU implementing addressing strategy that allows execution of q butterflies simultaneously in radix r stage. Launching multiple butterflies

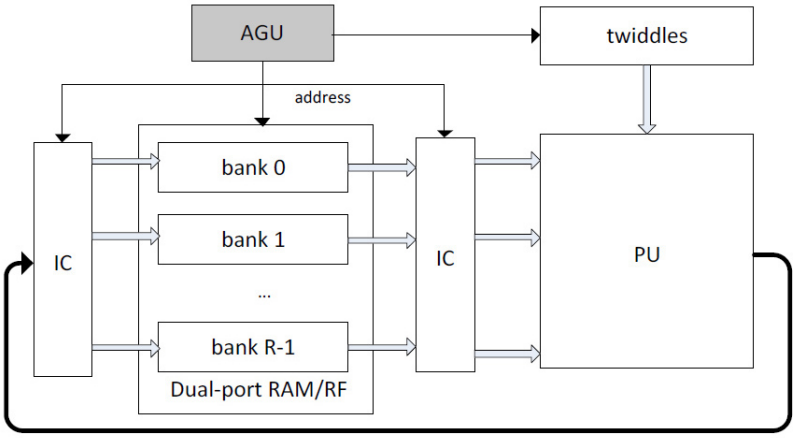

Fig. 1. Architecture for a FFT processor utilizing dual-port memories

Tab. 1. Estimated clocks count for different modifications of the approach

	One butterfly per clock		Proposed approach	
	Clocks	Radix	Clocks	Radix
512-point	192	8	192	8
1024-point	896	2/8	512	2/8
2048-point	1280	4/8	1024	4/8
4096-point	2048	8	2048	8

simultaneously makes radix r calculation q times faster, therefore granting significant performance improvement.

The AGU may use trivial traverse order (8) and bank assignment

$$m(k_{n-1},...,k_0)=\left(\sum_{i=0}^{n-1}k_i+qk_0\right) mod\ R\,. \quad (11)$$

Notice that the bank assignment (11) is a special case of formula (10) and equals to bank assignment introduced in [1] for $r = R$.

Theorem 1. *The bank assignment* m (11) *with trivial traverse order* T_c (8) *guarantees no conflicts for dual-port memory FFT processor.*

Values of n and r can be adjusted at run-timne to use one FFT processor to calculate transforms (and reverse transforms) of different sizes. Performance gain in comparison to other modifications of Johnson's approach for some sample lengths is addressed in a table below. Notice that the numbers are estimates: pipeline length and, probably, some other constant modifiers must be added in order to obtain real clock count. Although only values for power of 2 radixes are listed, the approach can be used with non-power of 2 radixes as well Tab. 1.

4. FFT PROCESSOR UTILIZING SELF-SORTING ADDRESSING

Consider a FFT sampled at $N = r \cdot R^{n-1}$ points using radix r, R butterfly operations, i. e. $r_0 = r, r_1 = ... = r_{n-1} = R$, where $R = r \cdot q$. Both of common decompositions DIT and DIF lead to either input or output having reversed digit order, i.e. in order to obtain the result an explicit digit reverse operation must be performed. An improved architecture that mixes digit reverse into a FFT processor is proposed. So no explicit digit reverse is required, while it is still running multiple butterflies per clock in radix r stage.

The same bank assignment (11) as in section 3 is used, first $\frac{n+1}{2}$ stages use trivial traverse order (8) as well. However, for radix R stages outputs of butterflies are transposed: output numbered $[\overline{\overline{w}},\overline{w}]$ is written as if it was numbered $[\overline{w},\overline{\overline{w}}]$, where $\overline{w}\in 0..r-1$, $\overline{\overline{w}}\in 0..q$. Starting from stage $\frac{n+1}{2}$, a permutation of outputs is introduced: for stage c, where $c \neq n-1$, butterfly with inputs numbered

$$[\overline{\overline{k_{n-1}}},\overline{k_{n-1}},...,\overline{\overline{k_{c+1}}},\overline{k_{c+1}},\overline{\overline{k_c}},\overline{k_c},\overline{k_{c-1}},\overline{\overline{k_{c-1}}},...,\overline{k_{n-c}},\overline{\overline{k_{n-c}}},\overline{k_{n-c-1}},\overline{\overline{k_{n-c-1}}},\overline{k_{n-c-2}},\overline{\overline{k_{n-c-2}}},...,k_0]. \quad (12)$$

Outputs are stored in memory addresses calculated as for outputs numbered

$$[\overline{\overline{k_{n-1}}},\overline{k_{n-1}},...,\overline{\overline{k_{c+1}}},\overline{k_{c+1}},\overline{k_{n-c-1}},\overline{\overline{k_{n-c}}},\overline{k_{c-1}},\overline{\overline{k_{c-1}}},...,\overline{k_{n-c}},\overline{\overline{k_c}},\overline{k_c},\overline{\overline{k_{n-c-1}}},\overline{k_{n-c-2}},\overline{\overline{k_{n-c-2}}},...,k_0]. \quad (13)$$

Notice that which half of the stages performs reverses is insignificant (Fig. 2).

The resulting FFT processor has the following architecture (Fig. 3).

Consider the following traverse order

$$T_c(k_{n-1},...,k_{c+1},0,k_{c-1},...,[\overline{k_{n-c}},\overline{\overline{k_{n-c}}}],[\overline{k_{n-c-1}},\overline{\overline{k_{n-c-1}}}],k_{n-c-2},...,\overline{k_1},\overline{\overline{k_1}},k_0) =$$

$$=[k_{n-1},...,k_{c+1},k_{c-1},...,[\overline{k_{n-c}},k_0],[\overline{k_1},\overline{\overline{k_{n-c-1}}}],k_{n-c-2},...,\overline{k_1},k_{n-c},\overline{\overline{k_{n-c}}},\overline{k_{n-c-1}}],\ c<n-2, \quad (14)$$

$$T_{n-2}(k_{n-1},0,k_{n-3},...,[\overline{k_2},\overline{\overline{k_2}}],[\overline{k_1},\overline{\overline{k_1}}],k_0)=[k_{n-1},k_{n-3},...,\overline{k_2},k_0,\overline{k_1},\overline{\overline{k_2}},\overline{k_1}], \quad (15)$$

$$T_{n-1}(0,k_{n-2},...,k_2,k_1,k_0)=[k_{n-2},...,k_2,k_1,k_0]. \quad (16)$$

Theorem 2. *The bank assignment* m (11) *and traverse order* T_c (14), (15) *guarantee no memory conflicts for self-sorting FFT processor.*

5. FFT PROCESSOR UTILIZING SINGLE-PORT MEMORIES

Consider a FFT sampled at $N = r \cdot R^{n-1}$ points using radix r, R butterfly operations, i. e. $r_0 = r, r_1 = ... = r_{n-1} = R$, where $R = r \cdot q$ is even. As shown in section 3, an FFT processor providing significant performance improvements over pure-radix approach suggested in [1] can be constructed. The AGU can be further modified in order to allow use of $2R$ 1rw memory banks without increase of overall memory words count. Replacing dual-port memories with single-memories improves the architecture in terms of area while preserving the performance advantage over [1].

The modified FFT processor has the following architecture (Fig. 4).

The absence of memory conflicts is guaranteed by select of such memory assignment and traverse order that read/write operations for memory banks interleave in subsequent clocks. Let

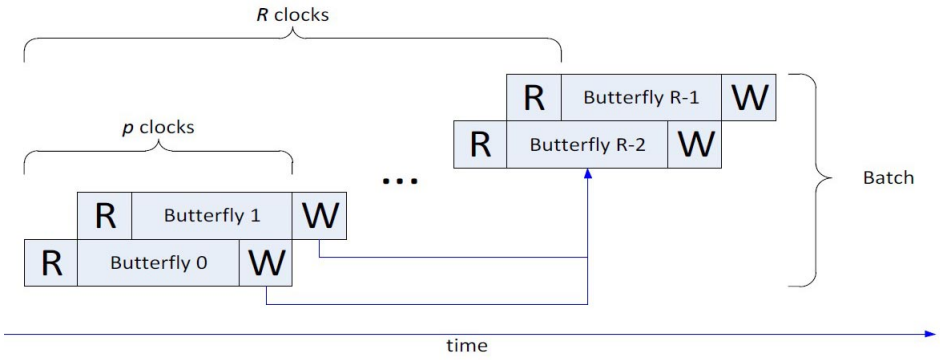

Fig. 2. Delay of write operations in stages performing reverse

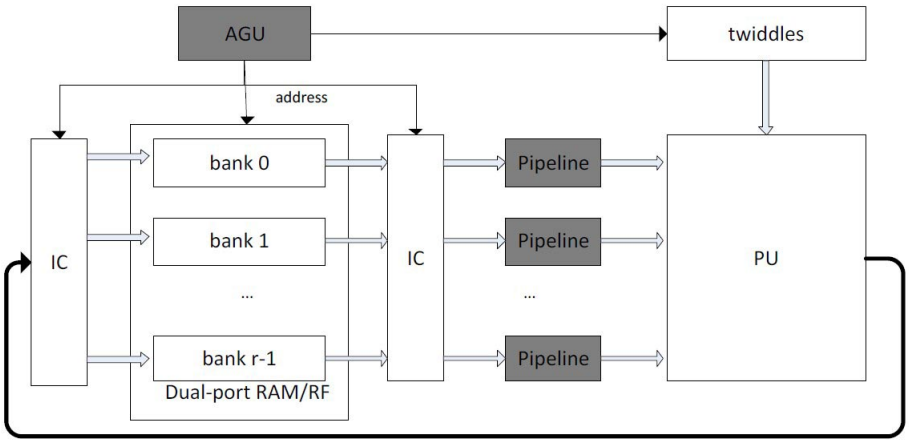

Fig. 3. Architecture for a self-sorting FFT processor

$$m(k_{n-1},...,k_0)=\left(2\sum_{i=2}^{n-1}k_i-(k_0 \bmod 2)\right) \bmod 2R. \quad (17)$$

Let traverse order for stage 0 be

$$T_0(k_{n-1},...,k_2,\overline{k_1},\overline{\overline{k_1}})=[k_{n-1},...,k_2,\sum_{i=2}^{n-1}k_i+\overline{k_1}+\overline{\overline{k_1}}\cdot r) \bmod R], \quad (18)$$

For other stages trivial traverse order is used:

$$T_c(k_{n-1},...,k_{c+2},\overline{k_{c+1}},\overline{\overline{k_{c+1}}},k_{c-1},...,k_0)=[k_{n-1},...,k_{c+2},\overline{k_{c+1}},\overline{\overline{k_{c+1}}},k_{c-1},...,k_0). \quad (19)$$

Theorem 3. *If the design's pipeline length is odd, the bank assignment* m (17) *used with traversal orders* T_c (18), (19) *guarantees no memory conflicts for single-port FFT processor.*

Notice that since every butterfly in radix r stage utilizes all possible values of k_0 and absence of conflicts in radix R stage is guaranteed by interleave of $d_0 \bmod 2$ values for subsequent butterflies, it is required to wait for the pipeline in radix r stage to finish before launching the first radix R stage.

6. SELF-SORTING FFT PROCESSOR WITH SINGLE-PORT MEMORIES

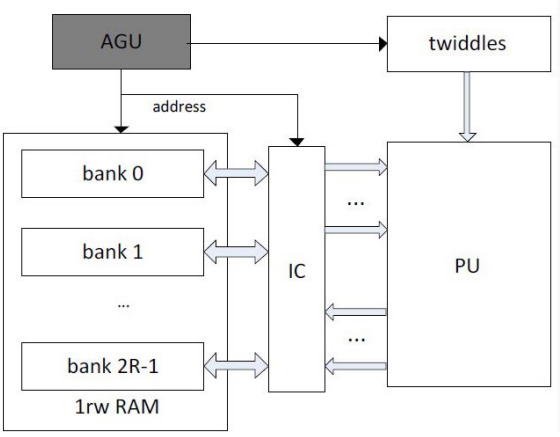

Fig. 4. Architecture for a FFT processor using 1rw memories

A self-sorting architecture on single-port memories is proposed for N-points FFT processor. It combines benefits of architectures proposed in section 5 and section 4. The proposed processor has a following architecture (Fig. 5).

Consider a FFT sampled at $N=r\cdot R^{n-1}$ points, where $R=r\cdot q$, $n\geq 3$, r is even. Let p be pipeline length, suppose p is odd. The idea is to combine approaches presented in the above sections: have read/write operations interleave for each memory bank and eliminate external digit reverse by reversing digits in stages starting from stage numbered $\left\lfloor\frac{n+1}{2}\right\rfloor$. Since addressing for self-sorting FFT does not

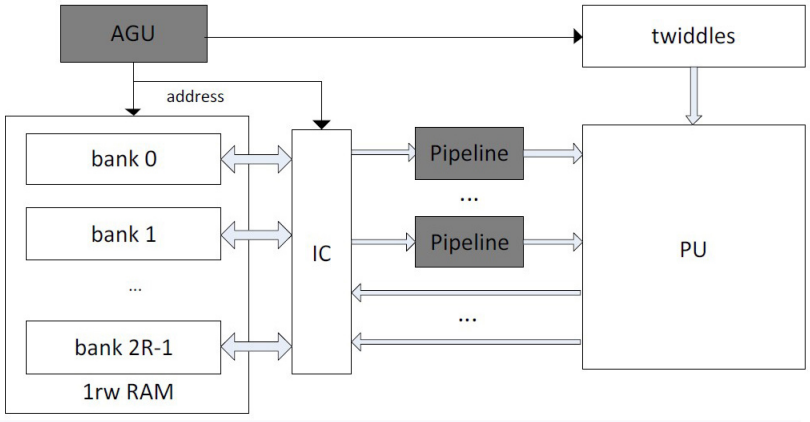

Fig. 5. Architecture for a self-sorting FFT processor on single-port memories

differ from plain FFT in stages that don't perform digit reverse, the substantial task is to combine the approaches in digit reversing stages. It can be done by grouping butterflies into batches of size $2R$ in a specific manner. In single-port approach no read/write conflicts are granted by interleave in some digit k_i. In stage one size $2R$ batch is constructed from two size R batches covering all values of k_c, k_{n-c-1} such that values of k_i interleave between the batches (butterflies from different batches interleave).

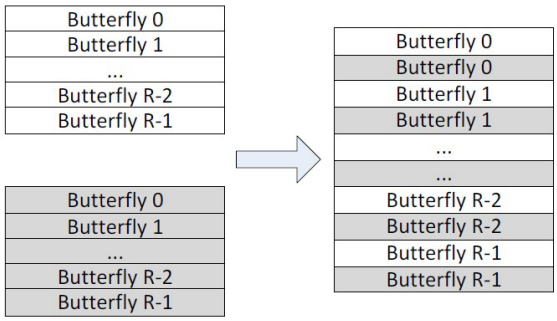

Fig. 6. Butterfly batches merging scheme

Similarly to self-sorting approach for dual-port memories outputs of radix R butterflies are transposed: output numbered $[\overline{w}, \overline{\overline{w}}]$ is written as if it was numbered $[\overline{\overline{w}}, \overline{w}]$.

Then with use of pipeline delay of length $2R-1-p$ there can be no read/write conflicts and no write before read conflicts (by batch construction).

However, it can be proven that in order for this approach to be successful in the last stage for radix 2, the bank assignment must be invariant with respect to switch of the last digit k_{n-1} and the first digit k_0. The bank assignment used for single-port memories does not comply with this requirement and heavily relies on asymmetry to grant read/write operations interleave. Therefore, a new bank assignment is required (Fig. 6).

The proposed bank assignment is

$$m(k_{n-1},...,k_0)=\left(2\sum_{i=1}^{n-2}k_i+2\left\lfloor\frac{k_0}{2}\right\rfloor+2\left\lfloor\frac{k_{n-1}}{2}\right\rfloor\right)+(k_0+k_{n-1})\bmod 2)\bmod R. \quad (20)$$

It is easy to prove that m is a correct bank assignment (the proof is similar to one presented in section 3 and is omitted). The traverse orders proposed for stages is

$$T_c(k^c)=\begin{cases} c=0, & \left[k_{n-1},...,k_2,\left(\sum_{i=1}^{n-1}k_i\right)\bmod r+\left\lfloor\frac{k_1}{r}\right\rfloor+2\cdot(k_1\bmod r)\right] \\ 0<c<\left\lfloor\frac{n+1}{2}\right\rfloor, & [k_{n-1},...,k_1,(k_0+k_{n-1})\bmod r] \\ \left\lfloor\frac{n+1}{2}\right\rfloor\le c<n-1, & \left[k_{n-1},...,k_{n-c+1},\left[k_{n-c}\bmod r,\left\lfloor\frac{k_{mrg}}{r}\right\rfloor\right],\left[k_{mrg}\bmod r,\left\lfloor\frac{k_{n-c}}{r}\right\rfloor\right],\right. \\ & \left.k_{n-c-2},...,k_2,k_{n-c-1},\left[\left\lfloor\frac{k_1}{2\cdot q}\right\rfloor,(k_0+k_{n-1})\bmod 2\right]\right] \\ c=n-1, & [k_{n-2},...,k_2, \\ & \left(\sum_{i=2}^{n-2}k_i+\left[\left[\frac{k_1}{2q},k_0\bmod 2\right],\left[\frac{k_1\bmod q}{2},\left(k_0+\frac{k_1}{q}\right)\bmod 2\right]\right]\right)\bmod r,\left[k_1\bmod 2q,\left\lfloor\frac{k_0}{2}\right\rfloor\right]\Bigg] \end{cases} \quad (21)$$

$$k^c=k_{n-1},...,k_{c+1},k_{c-1},...,k_0, \quad (22)$$

$$k_{mrg}=(k_1\bmod(2\cdot q))\cdot\frac{r}{2}+\frac{k_0}{2}. \quad (23)$$

Theorem 4. *For FFT sampled at $N=r\cdot R^{n-1}$ points, where r is even, if pipeline length p is odd, the single-port self-sorting FFT processor with pipeline delays postponing writes for $2R-p-1$ clocks with bank assignment (20) and traverse order T_c (21) has no memory conflicts.*

7. RESULTS

In this paper we generalized Johnson's approach [1] and considered not only conflict-free bank assignment, but also butterfly traverse order within a stage. We proposed a new parameterized conflict-free bank assignment generalizing the previous results on relatively prime mixed radix [2] and multiple butterflies per clock [3].

Using these results we considered four new FFT architecture modifications supporting run-time change of transform length (up to implementation-dependent maximum length) and direction. Correctness of used address assignments is proven. For architecture of a FFT processor with dual-port memories (section 3) High Level Synthesizable (HLS) SystemC model was created. The RTL obtained has reasonable area characteristics compared to commercially available cores, proving that the architecture can be effectively used to create actual designs. The results of gate-level synthesis show that the AGU utilizes negligible size and power compared to RAM and PU. For other architectures only reference models were developed.

For traditional dual-port memory architecture we considered modification for running multiple butterflies per clock for radixes other than 2/4. It substantially improves architecture performance if small and big radixes have large difference.

We also considered mix-radix self-sorting architecture. At the cost of $radix$ pipeline stages in the butterfly processing unit it allows simpler integration with other computations as part of reconfigurable DSP and allows using of overlapped I/O as stand-alone block improving the initiation interval up to 30%.

We considered single-port memory in-place architectures. The basic single-port architecture allows using in-place strategy for libraries without dual-port memories, effectively reducing memory area by 50% with modest requirement of odd pipeline length of butterfly processing unit. We also considered the hybrid architecture combining both self-sorting and single-port memory. It provides benefits of both approaches at the cost of $(2 \cdot radix - 1)$ pipeline stages.

8. SUMMARY

The architecture under consideration is one of the common architectures for computing long FFT. The novel algorithms are of practical interest as they increase possible design space by providing new area/performance trade-offs for practically useful scenarios like latest OFDM-based protocols for ground cable networks and 4G wireless.

The approach can be applied to building architectures for non-2^n lengths if it is of practical interest. The algorithms aren't unique and are defined up to transposition of some digits.

Only one of the developed architectures was implemented in RTL, so the practical implementation for other architectures is still required. There is a possibility that synthesis will imply some modifications in algorithms to make them more hardware-friendly.

Bibliography/References

1. *Johnson, L.G.* Conflict free memory addressing for dedicated FFT hardware // Circuits and Systems II: Analog and Digital Signal Processing, IEEE Transactions on, May 1992. Vol. 39, № 5. P. 312–316.

2. *Chen-Fong Hsiao, Yuan Chen, Chen-Yi Lee.* A Generalized Mixed-Radix Algorithm for Memory-Based FFT Processors // Circuits and Systems II: Express Briefs, IEEE Transactions on, Jan. 2010. Vol. 57, № 1. P. 26–30.

3. *Jo B.G., Sunwoo M.H.* New continuous-flow mixed-radix (CFMR) FFT Processor using novel in-place strategy // Circuits and Systems I: Regular Papers, IEEE Transactions on, May 2005. Vol. 52, № 5. P. 911–919.

4. http://www.xilinx.com/support/documentation/ip_documentation/ds808_xfft.pdf (дата обращения: 29.04.2013).

5. http://www.altera.com/literature/ug/ug_fft.pdf (дата обращения: 29.04.2013).

6. *M. Hegland.* A self-sorting in-place fast Fourier transform algorithm suitable for vector and parallel processing // Numerische Mathematik, 1994. Vol. 68, № 4. P. 507–547.

APPENDIX

Theorem 1. *If the bank assignment used with trivial traverse order*

$$m(k_{n-1},...,k_0)=\left(\sum_{i=1}^{n-1}k_i-qk_0\right)mod\ R, \tag{24}$$

$$(T_c(k_{n-1},...,k_{c+2},\overline{k_{c+1}},\overline{\overline{k_{c+1}}},k_{c-1},...,k_0)=[k_{n-1},...,k_{c+2},\overline{k_{c+1}},\overline{\overline{k_{c+1}}},k_{c-1},...,k_0]). \tag{25}$$

It guarantees no conflicts for dual-port FFT processor. See section 3 *for the processor's description.*

Proof: For radix R stage numbered c conflicts can only occur between wings of one butterfly. Suppose a conflict occurred on wings $\overline{k_c},\overline{\overline{k_c}}$, i. e.

$$m(k_{n-1},...,k_{c+1},\overline{k_c},k_{c-1},...,k_0)=m(k_{n-1},...,k_{c+1},\overline{\overline{k_c}},k_{c-1},...,k_0). \tag{26}$$

From definition of m it means $\overline{k_c}\equiv\overline{\overline{k_c}}(mod\ R)$, which leads to $\overline{k_c}=\overline{\overline{k_c}}$, since $\overline{k_c},\overline{\overline{k_c}}<R$. So conflicts in radix R stages are impossible. It can be shown in the same manner that in radix r stage there are no conflicts within one butterfly.

Suppose 2 butterflies in the same butterfly group in radix r stage have a conflict, i. e.

$$m(k_{n-1},...,k_2,\tilde{k_1}\cdot q+\overline{k_1},\overline{k_0})=m(k_{n-1},...,k_2,\tilde{k_1}\cdot q+\overline{\overline{k_1}},\overline{\overline{k_0}}), \tag{27}$$

where $\overline{k_1},\overline{\overline{k_1}}<q$. From definition of m it means $\overline{k_1}+\overline{k_0}\cdot q\equiv\overline{\overline{k_1}}+\overline{\overline{k_0}}\cdot q(mod\ R)$. Since $\overline{k_0},\overline{\overline{k_0}}<r$, it leads to $\overline{k_1}=\overline{\overline{k_1}}$, $\overline{k_0}=\overline{\overline{k_0}}$, therefore, conflict is impossible. ■

Theorem 2. *If the bank assignment and traverse order are used*

$$m(k_{n-1},...,k_0)=\left(\sum_{i=0}^{n-1}k_i+qk_0\right)mod\ R, \tag{28}$$

$$T_c(k_{n-1},...,k_{c+1},0,k_{c-1},...,[\overline{k_{n-c}},\overline{\overline{k_{n-c}}}],[\overline{k_{n-c-1}},\overline{\overline{k_{n-c-1}}}],k_{n-c-2},...,\overline{k_1},\overline{\overline{k_1}},k_0)=$$

$$=[k_{n-1},...,k_{c+1},k_{c-1},...,[\overline{k_{n-c}},k_0],[\overline{\overline{k_1}},\overline{\overline{k_{n-c-1}}}],k_{n-c-2},\overline{k_1},k_{n-c},\overline{\overline{k_{n-c}}},\overline{k_{n-c-1}}],\quad c<n-2, \tag{29}$$

$$T_{n-2}(k_{n-1},0,k_{n-3},...,[\overline{k_2},\overline{\overline{k_2}}],[\overline{k_1},\overline{\overline{k_1}}],k_0)=[k_{n-1},k_{n-3},...,\overline{k_2},k_0,\overline{\overline{k_1}},\overline{\overline{k_2}},\overline{k_1}], \tag{30}$$

$$T_{n-1}(0,k_{n-2},...,k_2,k_1,k_0)=[k_{n-2},...,k_2,k_1,k_0]. \tag{31}$$

It guarantees no memory conflicts for self-sorting FFT processor. See section 4 *for the processor's description.*

Proof: T_c is a transposition of digits in butterfly number, therefore it can be used as a traverse order.

Parts of every digit $k_i=[\overline{\overline{k_i}},\overline{k_i}]$ are permuted with symmetric parts of digits k_{n-i} and k_{n-i-1}, which results in digit reverse for k considered as a number constructed of digits $[\overline{k_i},\overline{\overline{k_i}}]$. Since original decomposition reversed digits k_i and butterfly outputs transposition restores order of $[\overline{\overline{k_i}},\overline{k_i}]$, it grants that input and output have the same digit order.

With this approach stages performing reverses are not in-place, therefore it must be ensured that during the stage computation a memory location is written only after it is read by a butterfly. For each stage performing reverse the correct order of read/write operations is guaranteed by reordering butterflies within the stage so that all butterflies with coinciding values of $k_{n-1},...,k_{c+1},k_{c-1},...,\overline{k_{n-c}},\overline{\overline{k_{n-c-1}}},k_{n-c-2},...,k_0$ are executed sequentially in one batch and adding pipeline delays postponing write operations for $R-p$ clocks, where p is pipeline length. Since write operations of butterflies from one batch can only corrupt values read in the same batch and the butterfly loop is pipelined, these measures are enough to grant correct read/write order.

The bank and address assignments used are the same as in section 3. Since in terms of addressing only the butterfly traverse order is modified, there are no memory conflicts (see Theorem 1). ■

Theorem 3. *If the design's pipeline length is odd and the bank assignment is used with traversal orders*

$$m(k_{n-1},...,k_0)=\left(2\sum_{i=0}^{n-1}k_i-(k_0 \bmod 2)\right) \bmod 2R\,, \tag{32}$$

$$T_0(k_{n-1},...,k_2,\overline{k_1},\overline{\overline{k_1}})=[k_{n-1},...,k_2,\sum_{i=2}^{n-1}k_i+\overline{k_1}+\overline{\overline{k_1}}\cdot r) \bmod R], \tag{33}$$

$$T_c(k_{n-1},...,k_{c+2},\overline{k_{c+1}},\overline{\overline{k_{c+1}}},k_{c-1},...,k_0)=[k_{n-1},...,k_{c+2},\overline{k_{c+1}},\overline{\overline{k_{c+1}}},k_{c-1},...,k_0)\,,\ c\neq 0\,. \tag{34}$$

It guarantees no memory conflicts for single-port FFT processor. See section 5 for the processor's description.

Proof: For radix R stage numbered memory conflicts can occur between read operations of different wings of one butterfly, write operations of different wings of one butterfly, or write operation of a butterfly and read operation of some subsequent butterfly. Read/write conflicts within one butterfly on wings $\overline{k_c}$, $\overline{\overline{k_c}}$ would mean

$$2\sum_{i=0,i\neq c}^{n-1}k_i+2\overline{k_c}-(k_0 \bmod 2)\equiv 2\sum_{i=0,i\neq c}^{n-1}k_i+2\overline{\overline{k_c}}-(k_0 \bmod 2)(\bmod 2R)\,, \tag{35}$$

which implies

$$\overline{k_c}\equiv\overline{\overline{k_c}}(\bmod R)\,,\ \text{i. e. }\ \overline{k_c}=\overline{\overline{k_c}}\,. \tag{36}$$

so conflicts within one radix R butterfly are impossible.

Since trivial traverse order is used in radix R stages and r is odd, values of k_0 interleave for subsequent butterflies. With pipeline having odd length, it guarantees that any 2 butterflies that have read and write operations within the same clock have different parity of k_0, and therefore use banks with different parity, therefore there are no conflicts on wings of different butterflies in radix R stages. The above reasoning holds when the butterflies are from different radix R stages as well.

For radix r stage consider memory bank assignment for an arbitrary wing of arbitrary butterfly:

$$m(T_0(k_{n-1},...,k_2,\overline{k_1},\overline{\overline{k_1}}),k_0)=(4\sum_{i=2}^{n-1}k_i+2\overline{k_1}+2\overline{\overline{k_1}}\cdot r+2k_0-k_0 \bmod 2) \bmod 2R=$$
$$=\left(4\left(\sum_{i=2}^{n-1}k_i+\left\lfloor\frac{k_0}{2}\right\rfloor+\overline{k_1}\cdot\frac{r}{2}+\left\lfloor\frac{\overline{k_1}}{2}\right\rfloor\right)+2(\overline{k_1} \bmod 2)+k_0 \bmod 2\right) \bmod 2R\,. \tag{37}$$

Points used in butterflies from one group have coinciding values of $k_{n-1},...,k_2,\overline{k_1}$ and differ only in $\overline{\overline{k_1}},k_0$. Since $\overline{\overline{k_1}}\le q-1$, $\left\lfloor\frac{k_0}{2}\right\rfloor\le\frac{r}{2}-1$ it is enough to consider

$$m_0=4\left\lfloor\frac{k_0}{2}\right\rfloor+2r\cdot\overline{k_1}+k_0 \bmod 2\,. \tag{38}$$

Since $4\left\lfloor\frac{k_0}{2}\right\rfloor<2r$, values of m_0 coincide only for coinciding values of $\overline{\overline{k_1}},k_0$. Hence there are no conflicts within one butterfly group.

Values of $\overline{k_1}$ interleave for subsequent butterfly groups. With pipeline having odd length, it guarantees that any 2 butterfly groups that have read and write operations within the same clock have different parity of $\overline{k_1}$, therefore use banks with different second bit in radix 2 representation of the bank's number. Hence there are no conflicts on wings of butterflies from different butterfly groups in radix r stage. ■

Theorem 4. *For FFT sampled at $N=r\cdot R^{n-1}$ points, where r is even, if pipeline length p is odd, the single-port self-sorting FFT processor with pipeline delay postponing writes for $2R-p-1$ clocks with the following bank assignment.*

$$m(k_{n-1},...,k_0)=\left(2\sum_{i=1}^{n-2}k_i+2\left\lfloor\frac{k_0}{2}\right\rfloor+2\left\lfloor\frac{k_{n-1}}{2}\right\rfloor\right)+(k_0+k_{n-1}) \bmod 2) \bmod R\,. \tag{39}$$

The following traverse order has no memory conflicts. See section 6 for the processor's description.

$$T_c(k^c)=\begin{cases} c=0, & \left[k_{n-1},...,k_2,\left(\sum_{i=1}^{n-1}k_i\right) mod\ r+\left\lfloor\frac{k_1}{r}\right\rfloor+2\cdot(k_1\ mod\ r)\right] \\ 0<c<\left\lfloor\frac{n+1}{2}\right\rfloor, & [k_{n-1},...,k_1,(k_0+k_{n-1})\ mod\ r] \\ \left\lfloor\frac{n+1}{2}\right\rfloor\le c<n-1, & \left[k_{n-1},...,k_{n-c+1},[k_{n-c}\ mod\ r,\left\lfloor\frac{k_{mrg}}{r}\right\rfloor],\left[k_{mrg}\ mod\ r,\left\lfloor\frac{k_{n-c}}{r}\right\rfloor\right],\right. \\ & \left. k_{n-c-2},...,k_2,k_{n-c-1},\left[\left\lfloor\frac{k_1}{2\cdot q}\right\rfloor,(k_0+k_{n-1})\ mod\ 2\right]\right] \\ c=n-1, & [k_{n-2},...,k_2, \\ & \left(\sum_{i=2}^{n-2}k_i+\left[\left[\frac{k_1}{2q},k_0\ mod\ 2\right],\left[\frac{k_1\ mod\ q}{2},\left(k_0+\frac{k_1}{q}\right)mod\ 2\right]\right]\right)mod\ r,\left[k_1\ mod\ 2q,\left\lfloor\frac{k_0}{2}\right\rfloor\right]\end{cases} \quad (40)$$

$$k^c=k_{n-1},...,k_{c+1},k_{c-1},...,k_0, \quad (41)$$

$$k_{mrg}=(k_1\ mod\ (2\cdot q))\cdot\frac{r}{2}+\frac{k_0}{2}. \quad (42)$$

Proof: Notice that write before read conflicts are impossible for in-place stages (numbered 0..$\left\lfloor\frac{n+1}{2}\right\rfloor-1$).

Consider stage numbered 0. The bank assignment for butterfly executed at iteration $k_{n-1},...,k_1$ is:

$$m(T_0(k^0),k_0)=4\left(\sum_{i=2}^{n-2}k_i+\left\lfloor\frac{k_{n-1}}{2}\right\rfloor+\left\lfloor\frac{k_0}{2}\right\rfloor+\left\lfloor\frac{k_1}{r}\right\rfloor\right)+2(k_1\ mod\ r)+(k_0+k_{n-1})\ mod\ 2. \quad (43)$$

Since the pipeline length is odd and subsequent butterflies have interleaving values of $k_1\ mod\ 2$, there are no read/write conflicts.

Consider bank assignment for stage numbered c, where $0<c<\left\lfloor\frac{n+1}{2}\right\rfloor$:

$$m(I_c(T_c(k^c),k_c))=2\left(\sum_{i=1}^{n-1}k_i+2\left\lfloor\frac{k_0}{2}\right\rfloor\right)+(k_0\ mod\ 2). \quad (44)$$

Subsequent butterflies have interleaving values of $k_0\ mod\ 2$. Since pipeline length is odd, there are no read/write conflicts in stage c.

Consider bank assignment for stage numbered c, where $\left\lfloor\frac{n+1}{2}\right\rfloor\le c<n-1$:

$$m(I_c(T_c(k^c),k_c))=2\left(\sum_{i=2}^{n-c-1}k_i+\sum_{i=n-c+1}^{n-1}k_i+\left[k_{n-c}\ mod\ r,\left\lfloor\frac{k_{mrg}}{r}\right\rfloor\right]+\left[k_{mrg}\ mod\ r,\left\lfloor\frac{k_{n-c}}{r}\right\rfloor\right]+\right.$$
$$\left.+2\left\lfloor\frac{k_1}{2\cdot q}\right\rfloor+2\left\lfloor\frac{k_{n-1}}{2}\right\rfloor\right)+k_0\ mod\ 2. \quad (45)$$

Subsequent butterflies have interleaving values of $k_0\ mod\ 2$, so there are no read/write conflicts. By replacing k_{n-c-1} with $\left[k_{mrg}\ mod\ r,\left\lfloor\frac{k_{n-c}}{r}\right\rfloor\right]$ and k_{n-c} with $\left[k_{n-c}\ mod\ r,\left\lfloor\frac{k_{mrg}}{r}\right\rfloor\right]$ the traverse order builds size $2R$ butterfly batches covering all values of digits to be swapped by the reverse. Since the first and the last butterflies originate from different size R batches, a pipeline delay of length $2R-p-1$ is enough to guarantee no write before read conflicts.

Consider bank assignment for stage $n-1$:

$$m(k_{n-1}, T_{n-1}(k^{n-1})) = 4\left(\sum_{i=2}^{n-2} k_i + \left\lfloor \frac{k_1}{2q} \right\rfloor \cdot q + \left\lfloor \frac{k_1 \bmod q}{2} \right\rfloor + \left\lfloor \left[k_1 \bmod 2q, \left[\frac{k_0}{2} \right] \right] \cdot 1/2 \right\rfloor\right) +$$

$$+ 2(k_0 \bmod 2) + \left(k_{n-1} + \left[k_1 \bmod 2q, \left\lfloor \frac{k_0}{2} \right\rfloor \right] \right) \bmod 2 . \quad (46)$$

Subsequent butterflies have interleaving values of $k_0 \bmod 2$, so there are no read/write conflicts. The above proof of absence of write before read conflicts holds. ■

НОВЫЕ АЛГОРИТМЫ ДЛЯ КОНВЕЙЕРНОГО ВЫЧИСЛЕНИЯ БПФ ПО СМЕШАННОМУ ОСНОВАНИЮ БЕЗ КОПИРОВАНИЯ НА МНОГОБАНКОВОЙ ПАМЯТИ С ПРОИЗВОЛЬНЫМ ДОСТУПОМ

Аннотация

В статье рассматривается метод реализации конвейерного вычисления БПФ по смешанному основанию на многобанковой памяти с дополнительными ограничениями. На основе рассмотренного метода предлагаются новые аппаратные архитектуры вычисления БПФ. Параллельное вычисление «бабочек» в стадиях с меньшим основанием позволяет существенно ускорить вычисления по смешанному основанию. Архитектура на основе однопортовой памяти позволяет реализовать некопирующую стратегию вычислений на библиотеках элементов без многопортовой памяти, обеспечивая уменьшение используемой памяти в 2 раза. Самоупорядочивающая архитектура позволяет использовать перекрывающиеся операции загрузки и выгрузки данных, обеспечивая уменьшение задержки вычислений до 30%. Также рассматривается архитектура, комбинирующая оба этих свойства.

Ключевые слова: конвейерное БПФ, БПФ по смешанному основанию, некопирующее БПФ, самоупорядочивающееся БПФ.

Салищев Сергей Игоревич,
старший преподаватель кафедры информатики СПбГУ,
инженер лаборатории Intel,
sergey.i.salishev@gmail.com,

Шеин Роман Евгеньевич,
аспирант кафедры системного программирования математико-механического факультета СПбГУ
marso.des@gmail.com.