A DESIGN METHODOLOGY OF BUFFER-MEMORY ARCHITECTURES FOR FFT COMPUTATION

Sheng-Ju Ku[†] *and Chin-Liang Wang*[†‡]

[†]Institute of Communications Engineering, National Tsing Hua University
[‡]Department of Electrical Engineering, National Tsing Hua University
Hsinchu, Taiwan 30013, Republic of China
phone: + (886) 3-5742567, fax: + (886) 3-5751787, email: clwang@ee.nthu.edu.tw

ABSTRACT

Memory-based architectures have received great attention for single-chip implementation of the fast Fourier transform (FFT). Basically, they can be roughly categorized as single-memory design, dual-memory design, and buffer-memory design. Among them, the buffer-memory design can balance the trade-off between memory size and control circuit complexity. In this paper, we present a design methodology of buffer-memory architectures for the radix-2 decimation-in-frequency FFT algorithm that can effectively reduce the needed memory. As compared to previous related works, the designs derived from the proposed methodology can reach the same throughput performance with a smaller memory size. These designs are rather attractive for long-length FFT applications, such as very-high-rate digital subscriber lines and digital video broadcasting.

1. INTRODUCTION

The discrete Fourier transform (DFT) is a very important tool or building block in areas of digital signal processing and communications. It has been adopted in some standards for modern wireline/wireless applications, such as very-high-rate digital subscriber lines (VDSL) [1] and digital video broadcasting (DVB) [2]. In these DFT applications, the transform length N is large. To meet the real-time requirements, it is necessary to develop dedicated fast Fourier transform (FFT) processors.

The FFT architectures based on radix-2^n algorithms can be divided into two categories: pipeline-based designs (see, for example, [3] and [4]) and memory-based designs (see, for example, [5]-[9]). In the pipeline-based designs, there is one radix-r butterfly unit (BU) and a local buffer at each stage of the signal flow graph (SFG) for the radix-r FFT algorithm. These designs can compute one transform sample per clock cycle and are suitable for very high throughput applications. However, they may consume a large chip area when the transform length N is very large. The memory-based designs usually adopt one radix-r BU to compute all the radix-r butterfly computations of the SFG and the corresponding architectures can be roughly categorized as three types: single-memory design [5], dual-memory design [6], and buffer-memory design [7]-[9]. Among these designs, the single-memory approach requires the least amount of memory, i.e. N words, but involves the most complicated control circuitry. On the contrary, the dual-memory method has the simplest control circuit design, but it needs the largest memory size of $2N$ words. In contrast, the buffer-memory design can balance the trade-off between memory size and control circuit complexity, where a buffer is located between the subtraction branch of the BU and the main memory to prevent memory-access-conflict. For existing buffer-memory designs based on the radix-2 decimation-in-frequency (DIF) FFT algorithm, the work of [7] has a total memory size of $1.5N$ words, a throughput of one transform sample per $\log_2 N$+1clock cycle, and 50% utilization efficiency in BU. The improved work presented in [8] doubles the throughput and BU utilization efficiency of those in [7], but with more memory of $2.5N$ words. As compared to the work of [8], the design of [9] decreases the total memory size to be $1.25N$ words with the same throughput and BU utilization efficiency. However, the critical path of this design includes one RAM read, one complex multiplier, one complex adder, and five 2-input MUX delays. It is longer than those of the previous two which include one RAM read, one complex multiplier, and three 2-input MUX delays.

In this paper, we propose a methodology to effectively reduce the buffer size and thus the total memory size of buffer-memory architectures for realizing the radix-2 DIF FFT algorithm. The designs based on the proposed methodology have the same throughput performance (in terms of the number of samples per clock cycle) as [9] and the same critical path as that in [8]. Dependent on the main memory partition, the needed buffer size ranges from 1 to $0.125N$ words; the more partition the main memory, the smaller buffer size with more cost in routing and control circuitry the design will involve. Finally, a design example of N=1024 with a total memory size of $1.125N$ words is given to verify the effectiveness of the proposed methodology, where the main memory is divided into four $N/4$-word banks.

2. THE RADIX-2 DIF FFT ALGORITHM

The N-point DFT is defined by

$$X_k = \sum_{n=0}^{N-1} x_n W_N^{nk} \qquad k = 0, 1, 2, \ldots, N\text{-}1 \tag{1}$$

where W_N=exp($-j2\pi/N$), called the twiddle factor, and the transform length N is assumed to be a power of two. In the radix-2 DIF FFT algorithm, the output sequence X_k will be divided into even- and odd-numbered samples and (1) can be reformulated as

$$X_{2s} = \sum_{n=0}^{(N/2)-1} \left(x_n + x_{n+N/2}\right) W_{N/2}^{ns} \tag{2}$$

$$X_{2s+1} = \sum_{n=0}^{(N/2)-1} \left(x_n - x_{n+N/2}\right) W_N^n W_{N/2}^{ns}$$

$$s = 0, 1, 2, \ldots, (N/2)\text{-}1 \tag{3}$$

This work was supported by the National Science Council of the Republic of China under Grants NSC 92-2213-E-007-072 and NSC 93-2213-E-007-097.

Equations (2) and (3) show that an N-point DFT can be decomposed into two $N/2$-point DFT's. Similarly, these two $N/2$-point DFT's can be further decomposed into four $N/4$-point DFT's. This kind of decomposition process can be repeated until $N/2$ 2-point DFT's are attained. At the rth stage of decomposition, there are 2^{r-1} m-point DFT's, where r =1, 2, …, $\log_2 N$ and $m=N/2^{r-1}$.

3. THE DESIGN METHODOLOGY FOR BUFFER-MEMORY DESIGNS OF FFT PROCESSOR

A buffer-memory design for FFT processor adopts a main memory to store the input data and the temporary results during the FFT computation, and a buffer to temporarily store the results from the subtraction branch of the BU for avoiding the memory-access-conflict of reading/writing two data from/to the same memory block, simultaneously. What is the minimum size of the required buffer for a buffer-memory design? To find the solution, we first assume that for a buffer-memory design of the radix-2 N-point DIF FFT algorithm the main memory consists of two $N/2$-word memory banks, R1 and R2, and the DFT input data x_i and $x_{(i+N/2)}$ are respectively stored in the address i of R1 and R2, where i= 0, 1, 2, ..., (N/2)-1. After the butterfly operations at the first stage of the SFG, the results from the addition and subtraction branches of the BU will form the input data of the two decomposed $N/2$-point DFT's at the second stage, respectively. To avoid the memory-access-conflict, the two input data for each butterfly of the $N/2$-point FFT algorithm must be stored into different memory banks so that they can be read out simultaneously. Hence, for each decomposed $N/2$-point DFT we should store one half input data into R1 and the other half into R2. Let each memory bank consist of two parts, the first half (FH) and the second half (SH), which include the addresses 0 ~ (N/4)-1 and $N/4$ ~ (N/2)-1, respectively. For simplicity of read/write addresses generation, we allocate the FH of R1 and R2 to one decomposed $N/2$-point DFT, and the SH of R1 and R2 to the other one. Without loss of generality, we use Fig. 1, a SFG of the 8-point radix-2 DIF FFT algorithm, to illustrate the memory arrangement for the butterfly operations. In Fig. 1, a pair of crossing lines represents a butterfly operation and the white node at the intersection represents the adder/subtractor of the BU. The upper and lower lines at the right hand side of a white node indicate the adder and subtractor outputs of the BU, respectively. The black node at a horizontal line terminal represents a storage location and the symbol above it indicates the address in a memory module. For example, R1[2] means in the address 2 of memory bank R1 and B[0] means in the address 0 of the buffer. The symbol below a horizontal line shows the twiddle factor that will be multiplied by the data stored at the address indicated by the left black node. From Fig. 1, we find that the sites of the FH of R2 (i.e., address 0 and 1) and the SH of R1 (i.e., address 2 and 3) at the SFG are exchanged after the butterfly operations of the first stage. This kind of memory arrangement will be referred to as the "half memory swapping strategy". However, this memory arrangement will cause memory-access-conflict because the two outputs of a butterfly computation will be stored into the different halves of the same memory bank, simultaneously. Hence, a buffer is inserted between the subtraction branch of the BU and the main memory, and the minimum required buffer is one half of a memory bank, i.e., $N/4$ words (two words for Fig. 1). The "half memory swapping strategy" can be also applied to the other decomposed DFT's of the SFG, as shown at the second stage of Fig. 1.

There is one drawback for the above memory arrangement: The second $N/2$-point DFT of the second stage can not be executed until the $N/4$ buffered data of the first $N/2$-point FFT have been multiplied by the twiddle factors and written back to R1. This will prolong the execution time and degrade the BU utility efficiency. This drawback can be solved by dividing the main memory into two pairs of $N/4$-word memory banks and each pair of banks are responsible to one $N/2$-point DFT computation at the second stage. This new memory arrangement for the 8-point DIF FFT algorithm is shown in Fig. 2. The operations of the two decomposed $N/2$-point DFT can be overlapped to increase both of the throughput rate and the BU utilization. We can easily derive that at the rth stage of the SFG, if the input data of an $N/2^{r-1}$-point decomposed DFT are sequentially stored in four different memory banks for a buffer-memory design that adopts the half memory swapping strategy, the minimum needed buffer size is $(N/2^{r-1})/4= N/2^{r+1}$ words, where r= 1, 2, 3, ..., $\log_2 N$-1. It is clear that the minimum buffer size decided at a given stage is applicable to the later stages.

The needed buffer size can be further reduced if the in-placement strategy is also adopted in the buffer-memory designs. Let the main memory consist of four $N/4$-word banks and the first half DFT input data are stored in the FH of the four banks, sequentially. The second half DFT input data are accessed from the external input buffer and sent to the BU directly for the butterfly computation of the first stage. After the butterfly executions of the first stage, the results from the addition branch of the BU are directly written back to the FH of the four memory banks by using the in-placement strategy. The results from the subtraction branch, which will be sequentially stored into the SH of the four banks, should be buffered first for avoiding the memory-access-conflict and the minimum needed buffer is $N/8$ words. The $N/8$-word buffer is also valid for the decomposed $N/2$-point DFT's at the second stage, where the half memory swapping strategy is adopted. Fig. 3 shows the SFG of the 8-point FFT which uses the new memory arrangement. Similarly, we can derive that the required buffer will be decreased to $N/16$ words if the main memory is divided to eight $N/8$-word memory banks, where the first two stages of FFT adopt the in-placement strategy and the half memory swapping strategy is used from the third stage, and so on.

Based on the above discussion, we can conclude a design methodology for buffer-memory designs of the N-point DIF FFT algorithm:

1. Evenly partition the N-word main memory into 2^q $N/2^q$-word memory banks, where q = 2, 4, 8, …, $\log_2 N$-1.
2. Insert a buffer of $N/2^{q+1}$ words between the subtraction branch of the BU and the main memory.
3. Store the first $N/2$ DFT input data into the FH of the 2^q memory banks, sequentially. Fetch the second $N/2$ input data and send them to the BU directly for the butterfly computation at the first stage of the SFG for the N-point FFT algorithm.
4. During the butterfly computation of the FFT algorithm, the in-placement strategy is used at the first q-1 stages and the half memory swapping strategy is adopted from the qth to the last second stage. The DFT outputs can be obtained from the two branches of the BU during the operations of the last stage.

4. A DESIGN EXAMPLE BASED ON THE PROPOSED DESIGN METHODOLOGY

4.1 Architecture

Based on the above derived design methodology, a buffer-memory architecture for the N-point FFT is presented in Fig. 4, where the main memory consists of four $N/4$-word memory banks, R1, R2, R3, and R4, and the buffer, BUF, has $N/8$ words. The memory banks and the buffer are composed of dual-port RAM. An $N/2$-word ROM is used to store the twiddle factors. During the FFT operations, the in-placement strategy is used at the first stage and the half memory swapping strategy is adopted from the second stage to the last second stage. The operations of this design can be summarized as the following three steps:

Step 1: Load and store the first $N/2$ input data of DFT into the FH of the four memory banks in the main memory.

The writing order of the four banks is R1, R2, R3 and R4.

Step 2: Load the second $N/2$ input data and execute the butterfly computation at the first stage of the SFG.

The outputs of the adder are directly stored into the FH of the four banks by using the in-placement strategy. The outputs of the subtractor are first buffered in the BUF temporarily, accessed for twiddle factor multiplication at $N/8$ clock cycles later, and then stored into the SH of the four banks.

Step 3: Execute the butterflies at the rth stage of FFT, where $r = 2, 3, \ldots, \log_2 N$.

The half memory swapping strategy is adopted in this step instead of the last stage. At each stage the execution of the first $N/8$ butterflies will be overlapped by the twiddle factor multiplications of the last $N/8$ subtraction results at the previous stage. After the second stage, the SFG can be decomposed to be four $N/4$-point DFT's and the operations of adjacent $N/4$-point DFT's can be also overlapped. When the butterflies of the last stage are being executed, the outputs of DFT can be directly acquired from the two outputs of the BU because the twiddle factor at the final stage is 1. The outputs produced by the subtraction branch will be first written back to R3 or R4 and then be accessed when the butterfly executions of the last stage are completed. When the stored outputs in the R3 and R4 are being accessed, the first $N/2$ input data of the next N-point DFT can be simultaneously loaded and stored in the FH of the four memory banks. It means that the step 3 can be overlapped with the step 1 of the next N-point DFT operations for $N/2$ clock cycles.

For an N-point FFT, the computation time of each stage is $N/2$ clock cycles and the latency to complete the three steps are $(N/2)\times(\log_2 N+2)$ clock cycles. Furthermore, due to overlapping the step 3 with the step 1 of the next N-point DFT, the average throughput will reach two transform samples per $\log_2 N+1$ clock cycles and the BU utilization efficiency can reach 100%, for which are the same as those of [9]. The critical path of Fig. 4 includes one RAM read, one complex multiplier, and three 2-input MUX delays, which is the same as those of [7] and [8]. For the other architectures derived from the proposed design methodology also have the same throughput rate, the BU utilization efficiency, and the critical path length as the example in Fig. 4.

Let a phase be equal to $N/2$ clock cycles and $n= \log_2 N$. To complete an N-point DFT needs $n+2$ phases, numbered from 0 to $n+1$. Like the works of [7]-[9], the control signals can be simply generated by an up-counter with bit number m= Ceiling($\log_2((N/2)\times(n+2))$. Assume that the output of the m-bit up-counter is $(a_{m-1}a_{m-2}\ldots a_2a_1a_0)$. The phase number is generated by the m-n+1 bits, $(a_{m-1}a_{m-2}\ldots a_na_{n-1})$; the other n-1 bits, $(a_{n-2}a_{n-3}\ldots a_2a_1a_0)$, are used to generate the memory read- and write-enable signals, the memory addresses, and the selection signals of multiplexers. The read-address, RA, for the four memory banks is equal to $(0a_{n-4}\ldots a_2a_1a_0)$ in phase 1 and equal to $(a_{n-2}a_{n-4}a_{n-5}\ldots a_2a_1a_0)$ from phase 2. TABLE I lists the write-enable and write-address for R1/R3, where the bit a_{n-1-p} will be selected from the n-1 bits, $(a_{n-2}a_{n-3}\ldots a_2a_1a_0)$, to generate the write-enable signals and the remaining n-2 bits will be used to generate the write-address, WA_a, in phase p, where p is from 1 to n-1. The writing operations of R2/R4 are performed at $N/8$ clock cycles later than those of R1/R3. Hence, the writing signals for R2/R4 are generated by the other m-bit signal, $(b_{m-1}b_{m-2}\ldots b_2b_1b_0)$, which is $N/8$ clock cycles later than $(a_{m-1}a_{m-2}\ldots a_2a_1a_0)$. In TABLE I, if we replace the signal 'a' by 'b', we can obtain the write signals of R2/R4. TABLE II lists the selection signals for the five multiplexers and the read-address for ROM for Fig. 4. In TABLE I and II, the phase numbers and the signals listed in each column are determined by the same m-bit signal. The write- and read- addresses for BUF are respectively equal to $(a_{n-4}a_{n-5}\ldots a_2a_1a_0)$ and $(b_{n-4}b_{n-5}\ldots b_2b_1b_0)$ during the FFT computation.

4.2 Chip Design

A prototype chip for the case of N=1024 is being developed by using the proposed buffer-memory architecture in Fig. 4, where the word lengths of the real and imaginary parts are 16 bits. The chip contains four RAM's of 256×32 bits each, one buffer of 128×32 bits, one ROM of 512×32 bits, one 16-bit×16-bit complex multiplier, two 16-bit complex adders, and some logic circuitry for generating the control signals. This design has been verified by the Verilog code and synthesized by the *Synopsys Design Vision* based on the standard cell library of UMC 0.18 μm CMOS technology. The chip contains about 400K transistors with a critical path of 4.73ns. Its clock rate can reach 211 MHz with the corresponding throughput rate of 38.3M transform samples per second. It meets the speed requirement of a VDSL transceiver [1].

5. CONCLUSIONS

In this paper, a new design methodology has been presented to effectively reduce the memory size of buffer-memory architectures for the radix-2 DIF FFT algorithm. Based on the proposed methodology, a family of buffer-memory architectures for FFT computation can be derived. As compared to previous related designs, these derived architectures can reach the same throughput performance and BU utilization efficiency with less memory. Finally, a design example has been given to verify the design methodology. Based on 0.18 μm CMOS technology, the design for the 1024-point FFT algorithm consists of about 400K transistors with a throughput of 38.3M transform samples per second. It is very suitable for single-chip design of long-length FFT applications.

REFERENCES

[1] VDSL Alliance SDMT VDSL Draft Standard Proposal, ETSI STC/TM6, April 1998.

[2] *Digital Video Broadcasting: framing structure, channel coding, and modulation for digital terrestrial television*, ETSI Standard, EN 300 744 v1.4.1, 2000.

[3] S. He and M. Torkelson, "Design and implementation of a 1024-point pipeline FFT Processor," in *Proc. IEEE Custom Integrated Circuits Conf.*, May 1998, pp. 131-134.

[4] Y.-N. Chang and K. K. Parhi, "An efficient pipelined FFT architecture," *IEEE Trans. Circuits and Systems-II: Analog and Digital Signal Processing*, vol. 50, pp. 322-325, June 2003.

[5] L. G. Johnson, "Conflict free memory addressing for dedicated FFT hardware," *IEEE Trans. Circuits and Systems-II: Analog and Digital Signal Processing*, vol. 39, pp. 312-316, May 1992.

[6] A. Delaruelle, J. Huisken, J.van Loon, and F. Welten, "A channel demodulator for digital audio broadcasting," in *Proc. IEEE Custom Integrated Circuits Conf.*, May 1994, pp. 47-50.

[7] C.-H. Chang, C.-L. Wang, and Y.-T. Chang, "Efficient VLSI architectures for fast computation of the discrete Fourier transform and its inverse," *IEEE Trans. Signal Processing*, vol. 48, pp. 3206-3216, Nov. 2000.

[8] C.-L. Wang and C.-H. Chang, "A new memory-based FFT processor for VDSL transceivers," in *Proc. IEEE Circuits and Systems Conf.*, vol. 4, May 2001, pp. 670-673.

[9] C.-K. Chang, C.-P. Hung, and S.-G. Chen, "An efficient memory-based FFT architecture," in *Proc. IEEE Circuits and Systems Conf.*, vol. 2, July 2003, pp. 25-28.

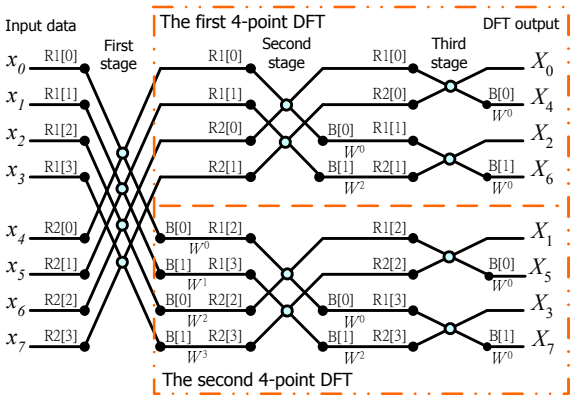

Figure 1. The SFG for the buffer-memory design of N=8 with two memory banks in the main memory.

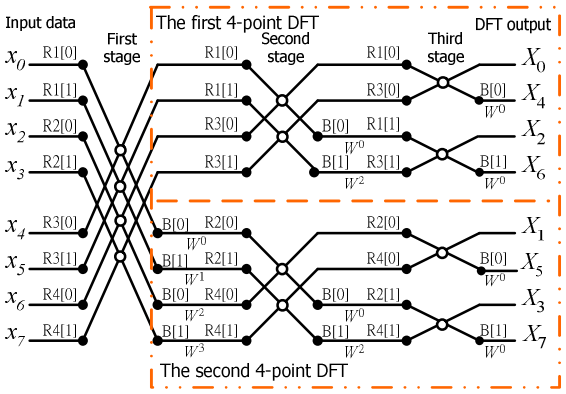

Figure 2. The SFG for the buffer-memory design of N=8 with four memory banks in the main memory.

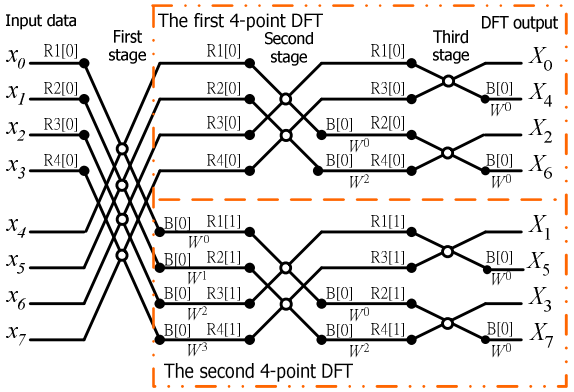

Figure 3. The SFG with the in-placement strategy for the first stage and the half memory swapping strategy for the second stage.

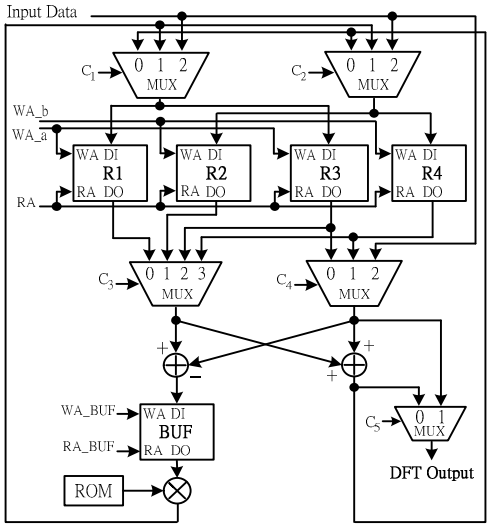

Figure 4. The proposed buffer-memory design with a buffer of N/8 words.

TABLE I. WRITE SIGNALS OF R1 AND R3 (–: MEANS "NEGLECTED")

Phase #	Write Address WA_a	Write Enable R1	Write Enable R3
0	$0a_{n-4}a_{n-5}\ldots a_1a_0$	$a_{n-2}+a_{n-3}$	$\overline{a}_{n-2}+a_{n-3}$
1	$a_{n-3}a_{n-4}a_{n-5}\ldots a_1a_0$	a_{n-2}	$\overline{a}_{n-2}$
2	$a_{n-2}a_{n-4}a_{n-5}\ldots a_1a_0$	a_{n-3}	$\overline{a}_{n-3}$
3	$a_{n-2}a_{n-3}a_{n-5}\ldots a_1a_0$	a_{n-4}	$\overline{a}_{n-4}$
⋮	⋮	⋮	⋮
n-2	$a_{n-2}a_{n-4}a_{n-5}\ldots a_{n-3}a_0$	a_1	$\overline{a}_1$
n-1	$a_{n-2}a_{n-4}a_{n-5}\ldots a_1a_{n-3}$	a_0	$\overline{a}_0$
n	$a_{n-3}a_{n-4}a_{n-5}\ldots a_1a_0$	1	0
n+1	–	1	1

TABLE II. CONTROL SIGNALS OF THE FIVE MULTIPLEXERS AND ROM

Phase #	C_1	C_2	C_3	C_4	C_5	Read Address of ROM
0	10	10	–	–	–	–
1	$0a_{n-3}$	$0b_{n-3}$	$a_{n-2}a_{n-3}$	10	–	$b_{n-2}b_{n-3}b_{n-4}\ldots b_2b_1b_0$
2	00	00	$0a_{n-3}$	$0a_{n-3}$	–	$b_{n-3}b_{n-4}b_{n-5}\ldots b_1b_0 0$
3	$0a_{n-3}$	$0b_{n-3}$	$0a_{n-3}$	$0a_{n-3}$	–	$b_{n-4}b_{n-5}b_{n-6}\ldots b_0 00$
⋮	⋮	⋮	⋮	⋮	⋮	⋮
n-1	$0a_{n-3}$	$0b_{n-3}$	$0a_{n-3}$	$0a_{n-3}$	–	$b_0 00\ldots000$
n	01	01	$0a_{n-3}$	$0a_{n-3}$	0	000...000
n+1	–	–	–	$0a_{n-3}$	1	–