Hardware realization of the robust time-frequency distributions

Nikola Žarić, Srdjan Stanković, Zdravko Uskoković

University of Montenegro, Faculty of Electrical Engineering, 20000 Podgorica, Montenegro

+382 67 531 815

+382 20 242 667

zaric@ac.me

www.tfsa.ac.me

A hardware realization of the L-estimate forms of robust time-frequency distributions is proposed. This hardware realization can be used for instantaneous frequency estimation for signals corrupted by a mixture of impulse and Gaussian noise. The most complex part in the hardware implementation is the block that performs sorting operation. In addition to the continuous realization, a recursive realization of the Bitonic sort network is proposed as well. The recursive approach also provides a fast sorting operation with a significantly reduced number of components. In order to verify the results, the FPGA implementations of the proposed systems were designed.

Robust time-frequency distributions, Bitonic sort network, FPGA implementation

INTRODUCTION

Signals used in real applications are usually corrupted by noise. Depending on signals and applications, numerous techniques and approaches for noise reduction have been proposed. An important class of signals that appears in many practical applications has the time-varying spectra (eg. radar, sonar, biomedicine, and telecommunications). Usually, these signals are analyzed by using time-frequency distributions (TFDs). Note that the standard forms of the TFDs are appropriate for representation and instantaneous frequency (IF) estimation of signals corrupted by Gaussian noise. However, if the signal is corrupted by impulse noise or by a mixture of Gaussian and impulse noise, TFDs usually produce poor results. Then, the robust TFDs should be used, [1]-[6]. The robust TFDs were introduced in [1]-[3] to analyze signals contaminated with pure impulse noise. However, in the case of a mixture of the Gaussian and impulse noise, the L statistics based techniques were developed, [4] and [5]. It was shown in [4] that for the mixture

noise models the optimal distribution is obtained as a solution of the L1 optimization problem. However this is a computationally demanding iterative procedure. Namely, the distribution should be recalculated and compared with the previous one in each iteration. Once it reaches the desired accuracy, or when the number of iterations exceeds the maximal one, the corresponding robust distribution is obtained. On the other hand, the L-estimate based approach, proposed in [5], provides a computationally less demanding and accurate solution. The most computationally demanding part of this procedure deals with the sorting operations. Having in mind that it is highly time consuming, the software implementations of [11] are inappropriate for real-time applications. Therefore, its hardware realization is proposed in this paper.

Various hardware solutions for real-time implementation of the standard TFDs are proposed [7]-[13]. The short-time Fourier transform realization is the simplest and most commonly used [7]-[9]. However, STFT produces a poor time-frequency resolution for the majority of signals. To alleviate this issue, the Wigner distribution was introduced in signal analysis. A hardware implementation of the Wigner distribution has been proposed in [4]. However, it can be only used for the analysis of monocomponent signals. The Wigner distribution especially provides a good representation for linear frequency modulated signals. For multicomponent signals, the S-method based hardware realizations were developed, [11]. A hardware system proposed in [6] provides analysis of signals with fast frequency variations. Note that all of these solutions are designed for the standard TFDs and they cannot provide good results for signals affected by a mixture of Gaussian and impulse noise. Therefore, we propose a flexible hardware for real-time implementation of the L-estimate robust TFDs in this paper. The proposed system relies on the robust short-time Fourier transform (STFT) realization. The robust STFT can be used for the robust S-method (SM) calculation. The proposed system is suitable for the VLSI implementation that enables a high speed processing. The system is implemented on the FPGA chips from Stratix III family.

The main efforts were made to design a system for sorting procedure. The Bitonic sort algorithm was used, because it exhibits the best performance for data sets used in our application. It belongs to the class of sorting networks that are suitable for hardware realization, since they provide a high degree of parallelism, [14]-[21]. Various realizations of the Bitonic sort algorithm on the FPGA devices have been proposed, [19], [20], [22]. However, most of them are based on a direct implementation of Bitonic sort network. A recursive realization of Bitonic sort network was proposed here, since the amount of chip space is a limiting factor in the FPGA implementation. It significantly reduces the number of required components, without affecting the total computational time.

The paper is organized as follows. A short review of the robust STFT and the robust S-method is presented in Section II. The architecture for realization of the robust time-frequency distributions, including a recursive realization of the Bitonic sort network is given in Section III. The Section IV is devoted to the FPGA implementation of the proposed system. The concluding remarks are given in Section V.

Theoretical Background

The robust STFT (RSTFT) can be obtained by using the L-estimate approach as [5]:

$$\begin{aligned} STFT_L(n,k) &= \sum_{i=1}^{N} a_i (rs_i(n,k) + j \cdot is_i(n,k)), \\ & rs_i(n,k) \in Rs(n,k), \\ & is_i(n,k) \in Is(n,k), \end{aligned} \tag{1}$$

where $Rs(n,k)$ and $Is(n,k)$ are the sorted versions of the sets $R(n,k) = \left\{ r_m(n,k) : \mathrm{Re}(x(n+m)e^{-j2\pi km/N}), m \in [-N/2, N/2) \right\}$ and $I(n,k) = \left\{ i_m(n,k) : \mathrm{Im}(x(n+m)e^{-j2\pi km/N}), m \in [-N/2, N/2) \right\}$, respectively. The elements: $rs(n,k)$ and $is(n,k)$ are sorted in a non-decreasing order such that: $rs_i(n,k) \le rs_{i+1}(n,k)$ and $is_i(n,k) \le is_{i+1}(n,k)$. The coefficients a_i, for even N, are defined as:

$$a_i = \begin{cases} \dfrac{1}{N(1-2\alpha)+4\alpha}, & \text{for } i \in [N_\alpha, N-N_\alpha-1] \\ 0, & \text{elsewhere,} \end{cases} \tag{2}$$

where $N_\alpha = (N-2)\alpha$ and $\alpha \in [0,1/2]$. For α=0 and α=1/2, the standard STFT and marginal median RSTFT follow, respectively. The value of parameter α has to be chosen to satisfy a trade-off between noise suppression (large α) and spectral representation (small α).

The corresponding robust spectrogram can be obtained as:

$$SPEC_L(n,k) = \left|\mathrm{Re}(STFT_L(n,k))\right|^2 + \left|\mathrm{Im}(STFT_L(n,k))\right|^2 \tag{3}$$

The robust spectrogram, like the standard one, produces an ideal concentration only for constant frequency modulated signals. In order to provide an efficient tool for analysis of linear frequency modulated multicomponent signals, the L-estimate robust S-method (RSM) has been introduced in [6]. It is defined as:

$$RSM_L(n,k) = \sum_{l=-L}^{L} P(l) STFT_L(n,k+l) STFT_L^*(n,k-l), \tag{4}$$

where $P(l)$ is the frequency domain window, while STFT* denotes a complex conjugate of the STFT. The cross-terms free distribution will be obtained if the minimal distance between two auto-terms is higher than $2L$+1.

Architecture for robust time-frequency distribution realization

The architecture for the RSTFT and the RSM real-time implementation is proposed. The block scheme for realizations of the RSTFT and the RSM is given in Fig 1.a. A corresponding architecture is shown in Fig 1.b.

The samples of the input signal $x(n+m)$ are multiplied by the corresponding basis functions $e^{-j2\pi mk/N}$, within the BLOCK 1. In order to avoid complex multiplications, the real and imaginary parts of the basis functions are considered separately. Thus, the signals: $R(n,k)=\left\{r_m(n,k):\mathrm{Re}(x(n+m)e^{-j2\pi km/N}),m\in(-N/2,N2)\right\}$ and $I(n,k)=\left\{i_m(n,k):\mathrm{Im}(x(n+m)e^{-j2\pi km/N}),m\in(-N/2,N2)\right\}$ are obtained. These signals are sorted within the BLOCK 2. The sorted elements $Rs(n,k)=\left\{rs_p(n,k):rs_p(n,k)\le rs_{p+1}(n,k)),p\in[1,N]\right\}$ and $Is(n,k)=\left\{is_p(n,k):is_p(n,k)\le is_{p+1}(n,k)),p\in[1,N]\right\}$, are obtained as the output of this block . This is the most complex and computationally demanding part of the proposed procedure. Within the BLOCK 3 the L-estimate RSTFT is obtained. Finally, the BLOCK 4 is employed to obtain the robust S-method. It is realized by using the hardware for standard SM realization. In the sequel, the sorting block, as the most challenging part of the architecture, will be considered.

BLOCK 2 – Hardware realization of sorting procedure

The Bitonic sort network for $N=32$ elements is shown in Fig 2. It performs sorting operation through Bitonic merge (BM) and Bitonic split (BS) operations. For a given scheme we have five BM blocks of different dimensions (denoted as BM_2, BM_4, BM_8, BM_{16}, and BM_{32}). Within these framed blocks, the BS blocks perform the sorting operations, and they are denoted as BS_2, BS_4, BS_8, BS_{16}, and BS_{32}. Note that, in general, the number of BM operations, for a sequence of N elements, is equal to $q=\log_2 N$. The i-th BM block ($i\in[1:q]$) combines two ordered sequences of the size 2^{i-1} (one sorted in ascending, the other in a descending order) in one sequence of the size 2^i. Each BM_i block contains i BS blocks, and the total number of BS operations is $(1+q)q/2$. As denoted in Fig 2., the blue BS blocks sort elements in an ascending order, while the red blocks perform the sorting operation in the opposite direction.

Observe that the BS steps generally appear repeatedly in the BM stages. For example, the BS_2 step is used in all the BM stages, while the BS_4 step appears in the BM_4, BM_8,

BM_{16}, and BM_{32} stages. Obviously, there is a large number of circuits, repeated to perform the same operations. Thus, the number of circuits can be reduced by using the recursive realization.

A recursive realization for a single processing line is given in Fig 3.a. Note that for a complete realization of N=32, one should use sixteen BS_2 blocks, eight BS_4 blocks, four BS_8 blocks, two BS_{16} blocks, and one BS_{32} block. The time lines of signals that control the whole system are presented in Fig 3.b-Fig 3.e. The signals: en_2, en_4, en_8, en_{16}, and en_{32} are the enable signals for the circuits BS_2, BS_4, BS_8, BS_{16}, and BS_{32}. The time line that shows the appearance of the enable signals is given in Fig 3.b. Note that only one BS circuit is active in one clock cycle.

Multiplexers are used to select the signal for the current BS step, either from the previous or from the next BS step. For example, we can observe that the input of BS_2 circuit is the input sorting element in the BM_2 stage or, if used for split operation, the output of BS_4 circuit. The role of multiplexers is similar for all other steps. The signals: sel_2, sel_4, sel_8, and sel_{16} are the address signals for multiplexers. They are shown in Fig 3.c. For sel=0 and sel=1, the inputs S_0 and S_1 are selected, respectively. The signal "do not care" ("x") stands when the corresponding BS circuit is not active.

Note that in a particular processing line, BS_2, BS_4, BS_8, BS_{16}, and BS_{32} should provide sorting operations in ascending or descending order. The order of sorting operations varies for different processing lines. To ensure the correct sorting order in each particular processing line, the signal "*dir*" is used. During the execution cycles, the signals "*dir*" for BS_2, BS_4, BS_8, BS_{16}, and BS_{32} circuits are given in Fig 3.d. The logical value zero provides sorting in an ascending, while the logical value one enables sorting in a descending order. It is important to observe that a "*dir*" signal can be used for various BS blocks by selecting the necessary number of bits (active bits are framed in Fig 3.d). Thus, for BS_2 circuits the signal "*dir*" is a 16 bit signal, with bits ($dir(0)$, $dir(1)$,…,$dir(15)$), for BS_4 circuits it is 8 bit signal ($dir(1)$, $dir(3)$, $dir(5)$,…, $dir(15)$), etc.

The control signal "*out*" is used to enable storing of sorted signal when the last BM stage is completed (Fig 3.e).

In order to provide automatic selection of all control signals, the finite state Moore machine is proposed in Fig 4. The input parameter for the state machine is the required number of BM stages: $q=\log_2 N$. The number of states in the finite state machine is the same as the number of BS circuits. Note that only for the BS_2 state the transition to the next state depends on the current BM stage. In all other cases, the next state is the previous BS state (for example: current state $BS_8 \rightarrow$ next state BS_4).

The values of all control signals for various stages are presented in Fig 4. As stated before, the sorting direction control signal "*dir*" is the same within one BM stage. Thus, its values depend only on the current BM stage. The values of signal "dir" can be read from a look-up table (LUT).

BLOCK 3 – architecture for realization of the L-estimate STFT

The sorted signals $Rs(n,k)$ and $Is(n,k)$ are obtained at the output of BLOCK 2. These signals are used to compute the L-estimate RSTFT within the BLOCK 3. Observe that the coefficients a_i (defined by (2)) differ from zero only for $i \in [N_\alpha, N-N_\alpha-1]$. Also, the coefficients a_i are constant within this interval, and they have the value $a=1/(N(1-2\alpha)+4\alpha)$. Thus, the real and imaginary parts of the L-estimate RSTFT can be obtained as:

$$RSTFT_{L\{\mathrm{Re}\}}(n,k) = a\sum_{i=N\alpha}^{N-N\alpha-1} rs_i(n,k),\ \ rs_i(n,k) \in Rs(n,k), \tag{5}$$

$$RSTFT_{L\{\mathrm{Im}\}}(n,k) = a\sum_{i=N\alpha}^{N-N\alpha-1} is_i(n,k),\ \ is_i(n,k) \in Is(n,k). \tag{6}$$

An architecture for the real part of L-estimate RSTFT realization is shown in Fig 5. The same architecture can be used to compute the imaginary part. Note that for α=0.5 the median form of the RSTFT follows, while the standard STFT can be obtained for α=0. Thus, the proposed architecture is applicable for the robust, as well as for the standard, distribution realization.

BLOCK 4 – Realization of the robust S-method

The L-estimate RSTFT can be further used for the quadratic time frequency distributions realization. Here, we will consider the robust S-method. Since the real and imaginary parts are separately obtained at the output of the L-estimate circuits, the RSM form given by (4) can be rewritten in a more appropriate form:

$$\begin{aligned} RSM(n,k) = & \left|RSTFT(n,k)\right|^2 \\ & +2\sum_{i=1}^{L} RSTFT_{Re}(n,k+i)RSTFT_{Re}(n,k-i) \\ & +2\sum_{i=1}^{L} RSTFT_{\mathrm{Im}}(n,k+i)RSTFT_{\mathrm{Im}}(n,k-i). \end{aligned} \tag{7}$$

The architecture for the RSM realization is given in Fig 6. Note that this hardware solution is flexible. Taking L=0 the robust spectrogram follows, while for L=N/2, the robust Wigner distribution is obtained. In practical applications L=3 is used and it is shown that this value represents a good trade-off between auto terms concentration and cross terms reduction

Although it appears that the number of multiple consecutive adders in Fig. 5 and Fig. 6 could be reduced by using an iterative procedure and block pipelining, it is not the case. Namely, since the number of adders depends on the values of parameters L and N_α., the iterative procedure would require the additional L+N_α multiplexers with at least L/2 and N_α/2 inputs. Also, it would require very complex control logic. Therefore the resource reduction will not be achieved.

Analysis of the proposed system

In order to analyze complexity of the system with recursive realization, the latency (Table I) and the total number of circuits (Table II) are considered.

The BLOCK 1 has the latency of one clock cycle. The total number of multipliers is $2N^2$ (N^2 for multiplication with the real parts of the basis functions and N^2 for multiplication with their imaginary parts). Also, one LUT of the size $2N^2$, containing real and imaginary parts of the basis functions, is required.

The latency caused by the Bitonic sort algorithm (BLOCK 2) is $((\log_2 N+1)\log_2 N)/2$ clock cycles. Since the real and imaginary parts are considered separately, the total number of comparators is $N\log_2 N$. The total number of multiplexers is $2N(\log_2 N-1)$, because they are not required for the BS_N circuits.

To obtain the L-estimate RSTFT (within the BLOCK 3) the signal passes through N_α adders and through a divide circuit. Total number of adders is $4N_\alpha-2$.

In order to obtain the RSM, the $STFT_L(n,k\pm L)$ have to pass through a multiplier and $L+1$ adders. The total number of circuits for the RSM realization is $2L$ adders and $2(L+1)$ multipliers.

Note that almost the same longest path holds for the system with a continuous realization. However, the total number of circuits is significantly higher. Namely, for the continuous realization the $N(\log_2 N+1)\log_2 N$ comparators are used, while the multiplexers are not necessary. Thus, the total number of circuits is smaller for the system with recursive realization.

FPGA implementation of the proposed systems

Prototypes of the proposed systems are implemented on the FPGA devices. The FPGAs are chosen among various hardware devices, such as digital signal processor and ASIC, since they offer high degree of parallelism, reconfigurability, and inbuilt special hardware. In addition, they are associated with short production time and low costs. Implementations of the systems with continuous and recursive realizations are performed. In both cases the input sequence with $N=64$ and $N_\alpha=4$ is considered. The 16 bit fixed-point IEEE 754 number format is used.

Both systems are implemented on the FPGA device EP3SE50F780C2 from Stratix III family produced by Altera, while the simulations are performed in Quartus II v9.0 software.

The available chip characteristics and utilized resources for one channel of the design with continuous realization are given in Table III. A part of the design scheme is given in Fig 7.

The chip characteristics for one channel of the system with recursive realization are given in Table IV. The finite state machine that generates necessary control signals for the recursive realization is implemented, as well. A part of the design scheme is shown in Fig 8.

The FPGA implementation of the proposed system with recursive realization is tested on the signal:

$$x(t) = e^{j64\cos(2.5\pi t)/3}.$$

The time interval $t \in [-0.5,0.5]$, with sampling rate $T=1/512$ is used. This sampling frequency is chosen to provide as much as possible better visualization of the results, but in real applications it can be much higher, even up to the order of MHz. Observe that this sampling frequency does not influence the latency of the proposed system. As it is shown in Table I, the parameters influencing latency are N, L, and α.

An example of simulation for a sample signal, including the control signals, is given in Fig 9. The output result is indicated with the control signal *stft*. Thus, the L-estimate RSTFT has the value 116 (for $t=0.146$ and $N=64$). For the same signal sample the software simulation (in Matlab 7) produces the value 115.999. Here, we will also mention the results: 145, 98, 33 (for other three samples), obtained by the proposed hardware. Note that the corresponding values in the software simulation are: 145.0002, 98.00005 and 32.99999. The average differences between the software simulation (with the floating point precision) and the proposed hardware realization (with the fixed point arithmetic) results are almost negligible, and they are of order 10^{-4} (this is a mean value for several signals with a large number of samples).

The aforementioned error is a consequence of the fixed point signal representation causing truncation of some signal values. Namely, the input signal values lower than 2^{-16} are truncated, as well as the values of the basis functions (sine and cosine). Since values

of these functions are lower than 1, products of signal and basis functions do not require normalization. Again, the products lower than 2^{-16} will be truncated, but in any case they are not included in the L-estimate calculation, and thus truncation does not affect the output. Therefore, the error in hardware realization is only caused by initial signal truncation and it is almost negligible when compared with output of software realization. If higher precision is required, the proposed system can be easily adjusted for higher precision (e.g., 32-bits signal representation).

The results for standard spectrogram, the L1 based optimization and L-estimate robust spectrogram are given in Fig. 10. The results for the L-estimate spectrogram (Fig. 10.c) are obtained by using the proposed chip and are visualized with Matlab 7.0, while results for other two forms are obtained directly in Matlab. We may observe that in the presence of mixed Gaussian and impulse noise the standard spectrogram (Fig. 10.a) and its instantaneous frequency estimation (Fig. 10.d) are useless, while the L1 based optimization provides better representation (Fig. 10.b) and IF estimation (Fig. 10.e). As expected, the L-estimate robust spectrogram produces the best signal representation (Fig. 10.c) and the IF estimation (Fig. 10.f). Additional advantages of the L-estimate over iterative L1 optimization solution are lower computational time and calculation complexity. Namely, in the case of L1 based optimization, distribution of the corresponding windowed signal part should be recalculated in each iteration for each point in the time-frequency plane. Then the reciprocal absolute difference between two consecutive distributions is calculated. It is further multiplied with Fourier transform of the windowed signal part and the product is summed with the reciprocal absolute difference. If the difference is smaller than the predefined value (usually 1% of the maximal distribution value) or the number of iterations exceeds the maximal one (usually 15), the iterative procedure is stopped (more details could be found in [4]). The average number of iterations for the considered case is nine. The latency of the software simulation (in Matlab 7.0 on Pentium IV with 2GHz and 2Gb of RAM) for iterative L1

optimization is 20 ms, for the L-estimate robust spectrogram 340 μs, while for the proposed hardware it is 560 ns.

Conclusion

The FPGA implementations of the L-estimate robust time-frequency distributions, based on continuous and recursive realizations of the Bitonic sort network, are proposed. The continuous realization of the Bitonic sort network provides fast sorting operation with simple realization, but with a high number of components. In order to reduce the hardware complexity, the recursive realization of the Bitonic sort network is proposed. It requires smaller number of components than a continuous realization, but without affecting the computational time. A finite state machine that provides automated selection of control signals for the recursive realization is designed, as well. The proposed system can produce standard, median, and L-estimate forms of time frequency distributions, by appropriate selection of the parameter α.

[1] V. Katkovnik, "Robust M-periodogram", *IEEE Trans. on Signal Processing*, vol. 46, no. 11, pp.3104-3109, 1998.

[2] I. Djurović, LJ. Stanković, "Robust Wigner distribution with application to the instantaneous frequency estimation*", IEEE Trans. on Signal Processing*, vol. 49, no. 12, pp. 2985-2993, 2001.

[3] I. Djurović, V. Katkovnik, LJ. Stanković, "Median filter based realizations of the robust time-frequency distributions", *Signal Processing*, vol. 81, no. 7, pp. 1771-1776, 2001.

[4] V. Katkovnik, I. Djurović, Lj. Stanković, "Instantaneous frequency estimation using robust spectrogram with varying window length", *International Journal on Electronic Communication*, AEU, vol.54, no.4, pp.193-202, 2000.

[5] I. Djurović, LJ. Stanković, J. F. Böhme, "Robust L-estimation based forms of signal transforms and time-frequency representations", *IEEE Trans. on Signal Processing*, vol. 51, no. 7, pp.1753-1761, 2003.

[6] I. Djurović, LJ. Stanković, B. Barkat, "Robust Time-Frequency Distributions based on the robust short time Fourier transform", *Annales des Telecommunications*, vol. 60, no. 5-6, pp.681-697, 2005.

[7] K. J. R. Liu, "Novel parallel architecture for Short-time Fourier transform", *IEEE Trans. on Circuits and Systems-II*, vol. 40, no. 12, pp. 786-789, 1993.

[8] M. G. Amin, K. D. Feng, "Short time Fourier transform using cascade filter structures", *IEEE Trans.on Circuits and Systems*, vol. 42, pp.631-641, 1995

[9] S. Stanković, LJ. Stanković, "An architecture for the realization of a system for time-frequency analysis", *IEEE Trans. on Circuits and Systems-II*, vol. 44, no. 7, pp. 600-604, 1997.

[10] B. Boashash, J.B. Black, "An efficient real time implementation of the Wigner-Ville distribution", *IEEE Trans.on Acoustic, Speech, Signal Processing*, vol. 35, no. 11, pp. 1611-1618, 1987.

[11] S. Stanković, LJ. Stanković, V. Ivanović, and R. Stojanović, "An architecture for the VLSI design of systems for time-frequency analysis and time-varying filtering", *Annals of Telecommunication*, vol. 57, no. 9/10, pp. 974-995, 2002.

[12] D. Petranovic, S. Stankovic, LJ. Stankovic, "Special purpose hardware for time frequency analysis," *Electronics Letters*, Vol.33, No.6, pp.464-466, Mar.1997.

[13] N. Žarić, I. Orović, S. Stanković, " Hardware realization of generalized time-frequency distribution with complex-lag argument" *EURASIP Journal on Advances in Signal Processing*, Vol. 2010, Article ID 879874, 10 pages, 2010.

[14] K. Batcher, "Sorting Networks and their Applications", *Proceedings of the AFIPS Spring Joint Computing Conference*, vol 32, 1968.

[15] J. JaJa, *An Introduction to Parallel Algorithms*. Addison-Wesley, Reading Massachusetts, 1992.

[16] V. Kumar, A. Grama, A. Gupta, G. Karypis, "Introduction to Parallel Computing*"* Benjamin Cummings, 1994.

[17] D. E. Knuth "The Art of Computer Programming," vol. 3, Addison Wesley, Reading Massachusetts, 1973.

[18] D. E. Culler, A. Dusseau, R. Martin, K. E Schauser, "Fast parallel sorting under LogP: from Theory and Practice", in "Portability and Performance of Parallel Programming" edited by T. Hey and J. Ferante, Wiley, 1994.

[19] M. Florin, K. E. Schauser, “Optimizing parallel bitonic sort“, *11th International Parallel Processing Symposium* (IPPS '97), pp. 303 - 309 , Geneva, Switzerland, 1997.

[20] R. Mueller, J. Teubner, G. Alonso, “Data Processing on FPGAs**”,** *Proc. of the 35th Int'l Conference on Very Large Data Bases (VLDB)/ Proc. of the VLDB Endowment*, vol. 2, Lyon, France, 2009.

[21] J. Harkins, T. El-Ghazawi, E. El-Araby, M. Huang, "Performance of Sorting Algorithms on the SRC 6 Reconfigurable Computer", *IEEE International Conference On Field-Programmable Technology* (FPT 2005), pp. 295-296, Singapore, 2005.

[22] J. Zhu, P. Sutton, “An FPGA implementation of Kak's instantaneously-trained, Fast-Classification neural networks”, *in Proc of IEEE International Conference on Field-Programmable Technology (FPT)***,** pp. 126- 133, Tokyo, Japan, 2003.

[23] LJ. Stanković, “A method for time-frequency analysis,” *IEEE Trans. on Signal Processing*, vol. 42, no. 1, pp. 225-229, 1994.

Fig 1 a) Block scheme of the proposed system, b) Architecture for realization of the robust STFT and robust SM

Fig 2 Detailed scheme of the Bitonic sort network for set of 32 elements

Fig 3 Recursive realization of the Bitonic sort network: a) recursive realization for one node, b) enable signals, c) multiplexer control signals, d) sorting direction signal “dir”, e) output valid signal

Fig 4 The finite state machine that produces control signals for architecture in Fig 3.a.

Fig 5 Realization of the L-estimate circuit

Fig 6 Architecture for the robust S-method realization (SH denotes shift operation)

Fig 7 A part of the design scheme of FPGA implementation of the proposed system with parallel realization of the Bitonic sort network

Fig 8 A part of the design scheme of FPGA implementation of the proposed system with recursive realization of the Bitonic sort network

Fig 9 Illustration of simulation for a system with recursive realization of the Bitonic sort network

Fig 10 a) Standard spectrogram, b) robust spectrogram based on iterative L1 optimization problem, c) L-estimate robust spectrogram, d) estimated IF for standard spectrogram, e) estimated IF based on iterative L1 optimization problem, f) estimated IF for L-estimate robust spectrogram

Table 1 The latency of the system with recursive realization

	Adders	Multipliers	Comparators	Other
BLOCK 1		1		
BLOCK 2			$((\log_2 N + 1)\log_2 N)/2$	
BLOCK 3	N_α			divide
BLOCK 4	L+1	1		

Table 2 Number of circuits for the system with recursive realization

	Adders	Multipliers	Comparators	MUX	Other
BLOCK 1		$2N^2$			LUT
BLOCK 2			$N\log_2 N$	$2N(\log_2 N - 1)$	
BLOCK 3	$4N_\alpha$-2				2 divide
BLOCK 4	2L	2(L+1)			

Table 3 FPGA chip characteristics for the system with parallel realization

EP3SE50F780C2	**No. of pins**	**Logic elements**	**Speed**	**Power consumption**
Available	488	38000	500 MHz	2.5 W

Utilized	321	26402	400 MHz	2.5 W

Table 4 FPGA chip characteristics for the system with recursive realization

EP3SE50F780C2	**No. of pins**	**Logic elements**	**Speed**	**Power consumption**
Available	488	38000	500 MHz	2.5 W
Utilized	70	15870	400 MHz	2.38 W

Figure 1

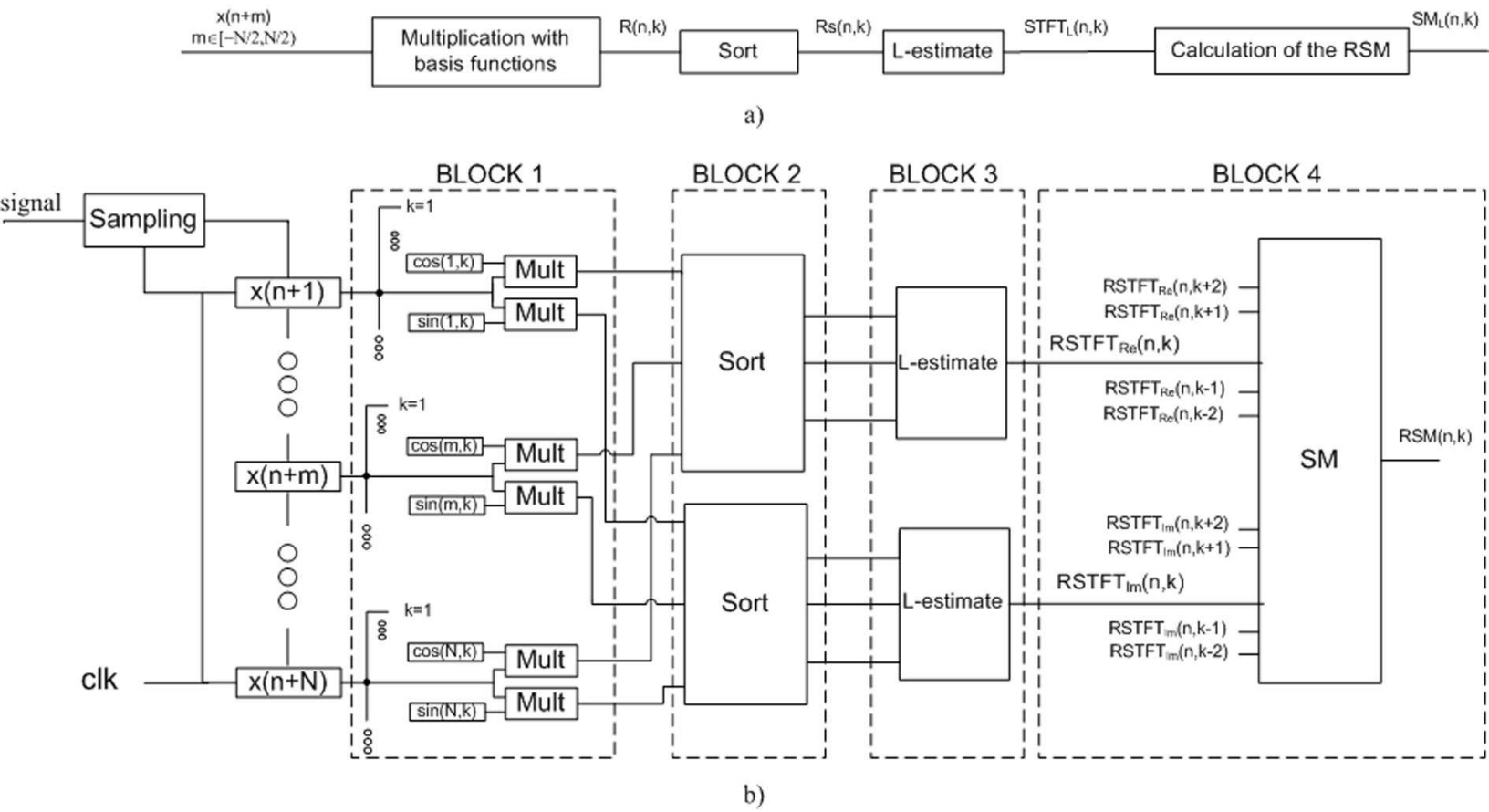

x(n+m)
m∈[−N/2,N/2)
Multiplication with basis functions
R(n,k)
Sort
Rs(n,k)
L-estimate
STFT_L(n,k)
Calculation of the RSM
SM_L(n,k)
a)
BLOCK 1
BLOCK 2
BLOCK 3
BLOCK 4
signal
Sampling
clk
x(n+1)
x(n+m)
x(n+N)
k=1
cos(1,k)
sin(1,k)
cos(m,k)
sin(m,k)
cos(N,k)
sin(N,k)
Mult
Sort
L-estimate
RSTFT_Re(n,k+2)
RSTFT_Re(n,k+1)
RSTFT_Re(n,k)
RSTFT_Re(n,k-1)
RSTFT_Re(n,k-2)
RSTFT_Im(n,k+2)
RSTFT_Im(n,k+1)
RSTFT_Im(n,k)
RSTFT_Im(n,k-1)
RSTFT_Im(n,k-2)
SM
RSM(n,k)
b)

Figure 2

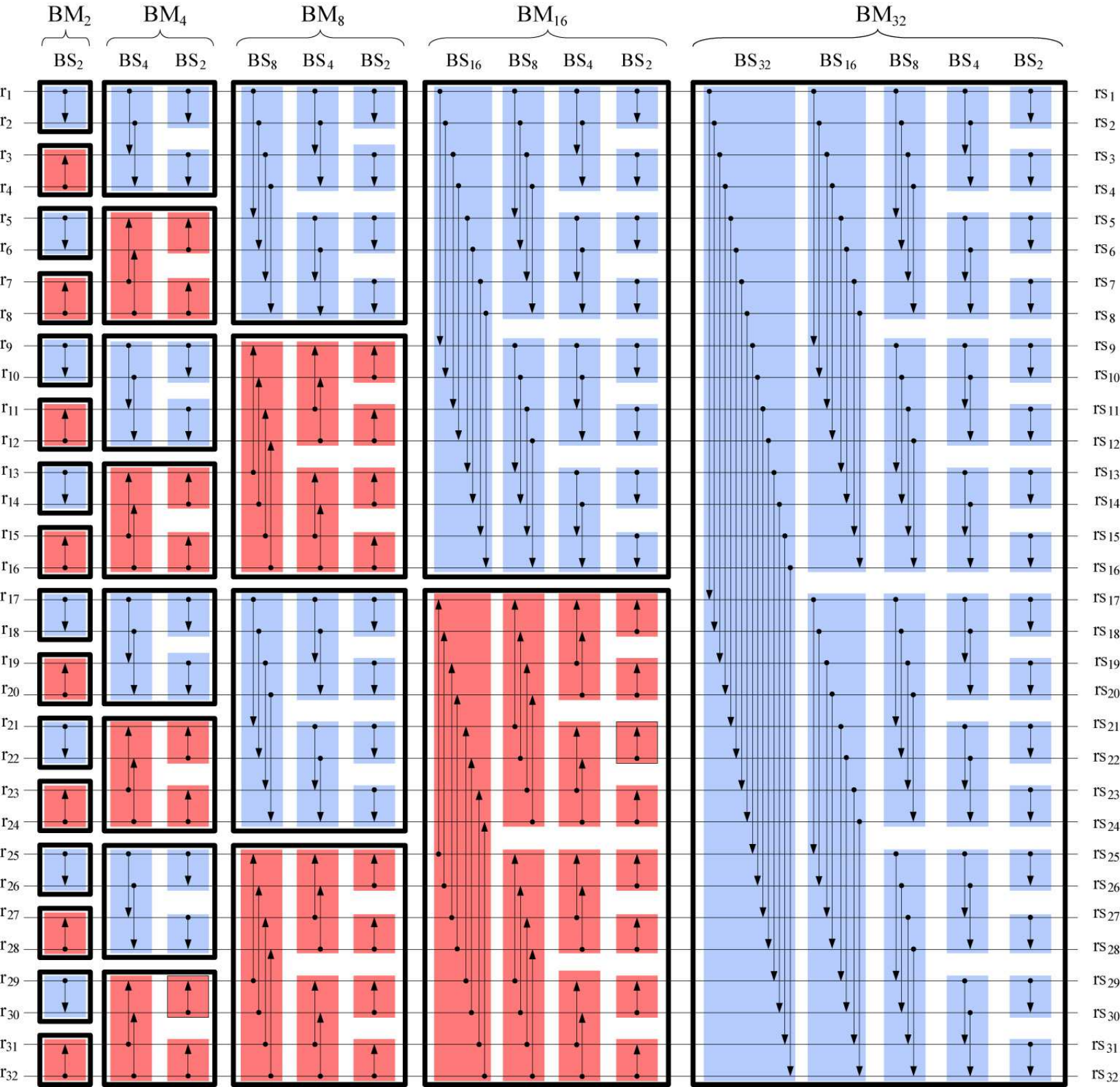
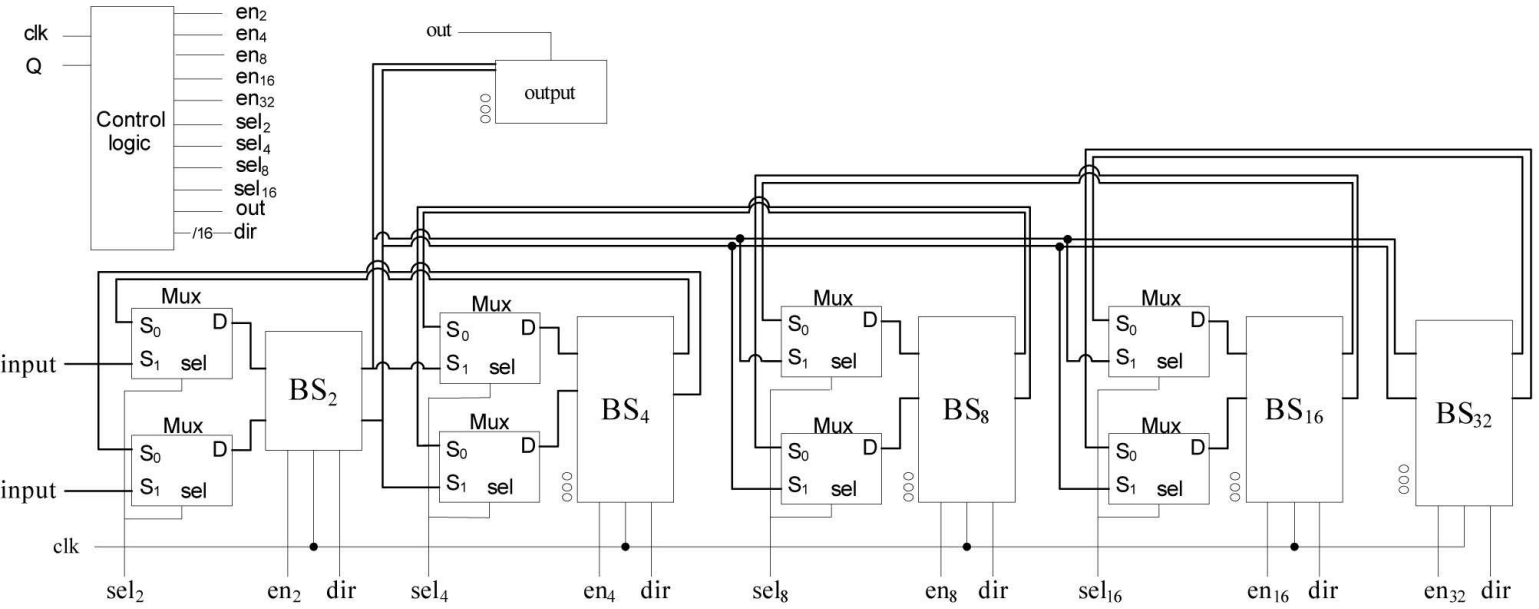
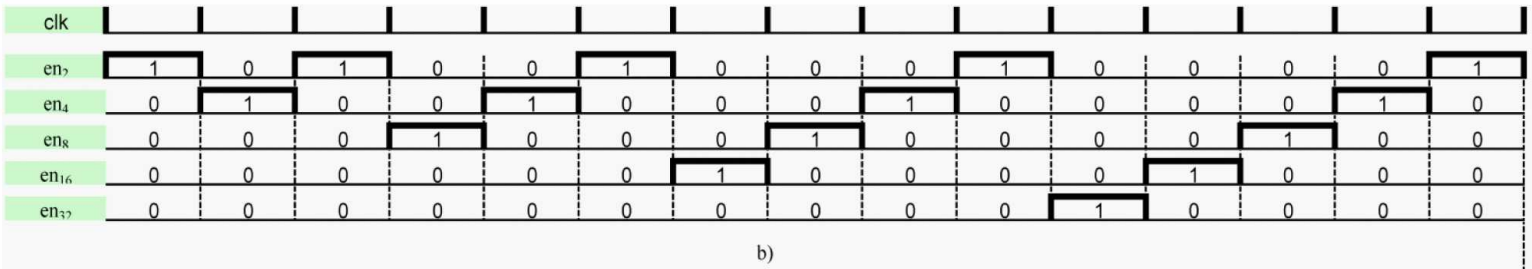

BM2
BM4
BM8
BM16
BM32
BS2
BS4
BS2
BS8
BS4
BS2
BS16
BS8
BS4
BS2
BS32
BS16
BS8
BS4
BS2
r1
r2
r3
r4
r5
r6
r7
r8
r9
r10
r11
r12
r13
r14
r15
r16
r17
r18
r19
r20
r21
r22
r23
r24
r25
r26
r27
r28
r29
r30
r31
r32
rs1
rs2
rs3
rs4
rs5
rs6
rs7
rs8
rs9
rs10
rs11
rs12
rs13
rs14
rs15
rs16
rs17
rs18
rs19
rs20
rs21
rs22
rs23
rs24
rs25
rs26
rs27
rs28
rs29
rs30
rs31
rs32

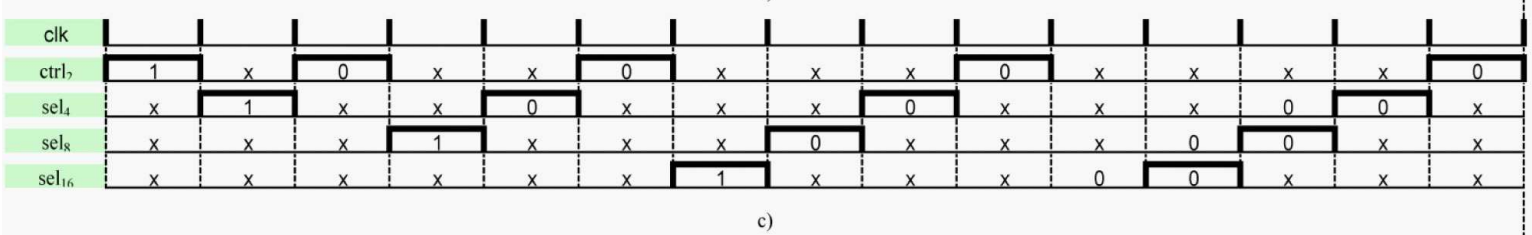
Figure 3



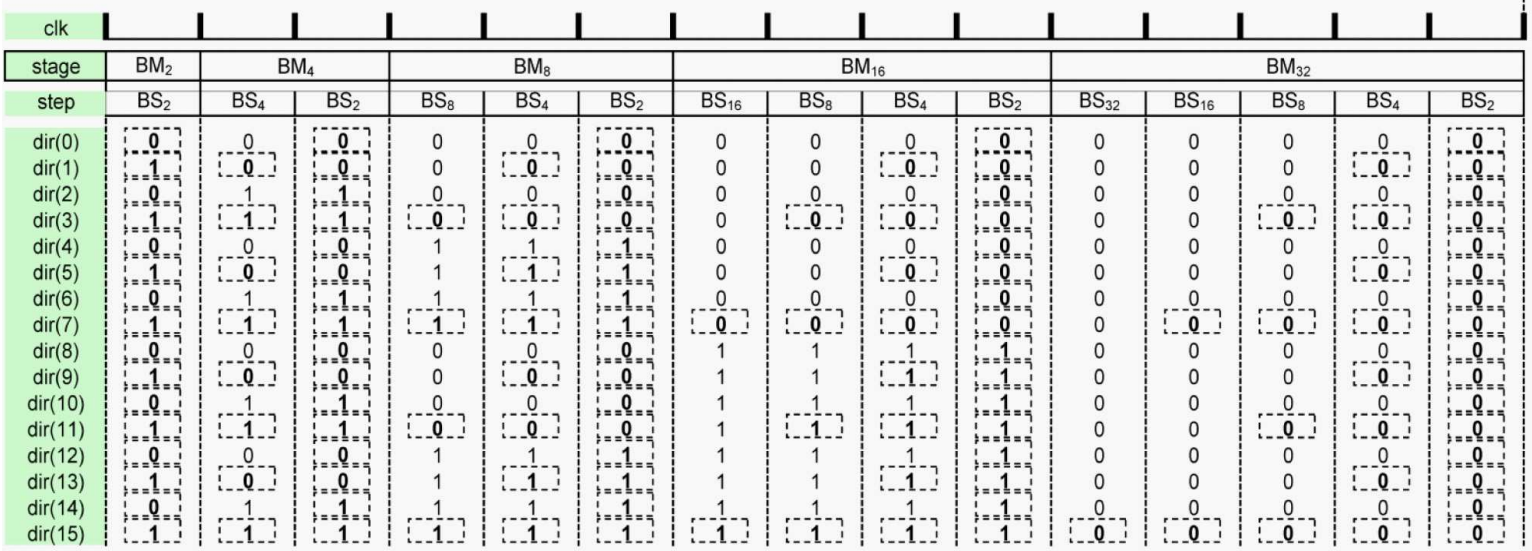
a)



b)



c)



d)

e)

Figure 4

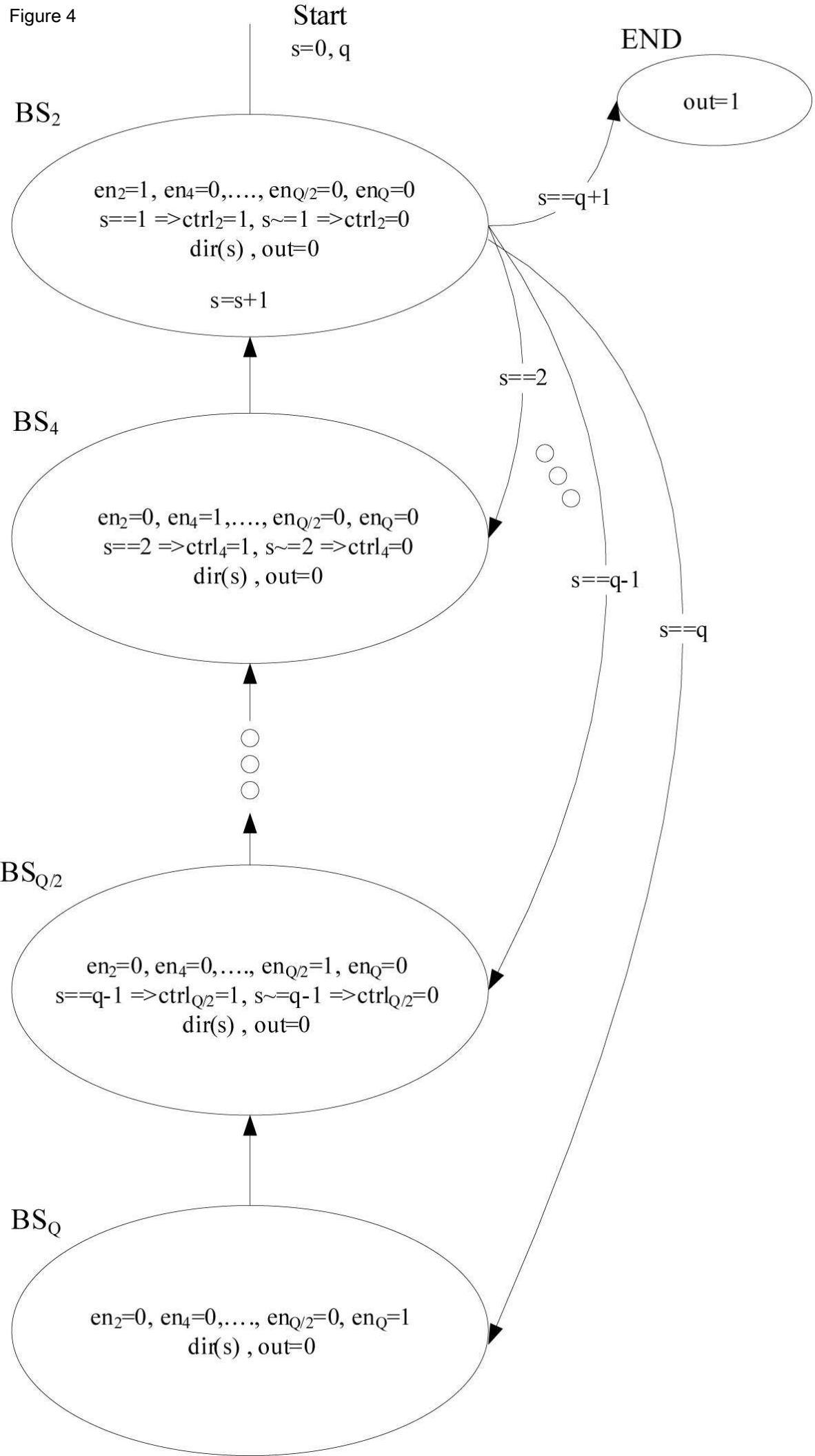

Start
s=0, q
END
out=1
BS2
en2=1, en4=0,…., enQ/2=0, enQ=0
s==1 =>ctrl2=1, s~=1 =>ctrl2=0
dir(s) , out=0
s=s+1
s==q+1
s==2
BS4
en2=0, en4=1,…., enQ/2=0, enQ=0
s==2 =>ctrl4=1, s~=2 =>ctrl4=0
dir(s) , out=0
s==q-1
s==q
BSQ/2
en2=0, en4=0,…., enQ/2=1, enQ=0
s==q-1 =>ctrlQ/2=1, s~=q-1 =>ctrlQ/2=0
dir(s) , out=0
BSQ
en2=0, en4=0,…., enQ/2=0, enQ=1
dir(s) , out=0

Figure 5

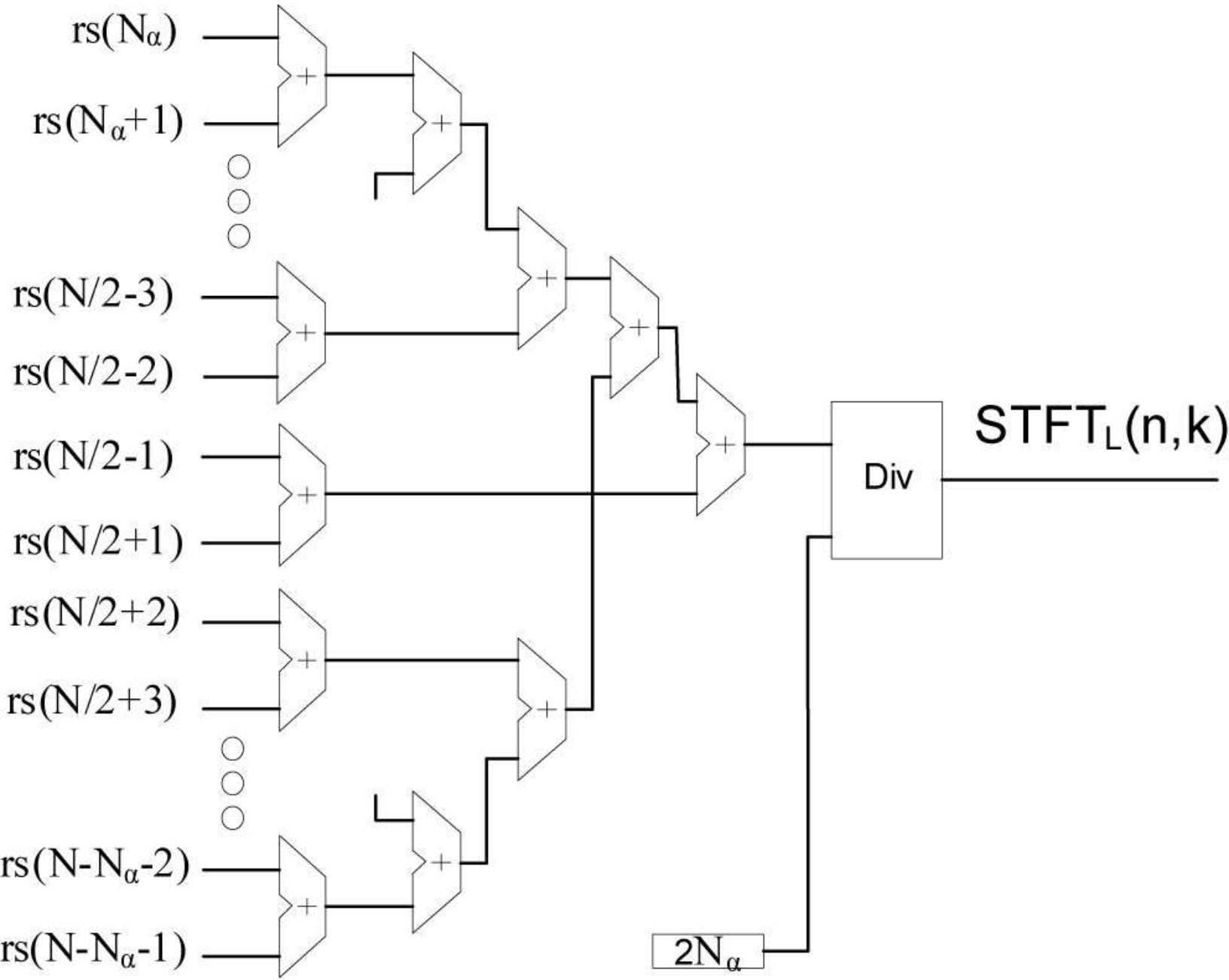

rs(Nα)
rs(Nα+1)
rs(N/2-3)
rs(N/2-2)
rs(N/2-1)
rs(N/2+1)
rs(N/2+2)
rs(N/2+3)
rs(N-Nα-2)
rs(N-Nα-1)
Div
STFTL(n,k)
2Nα

Figure 6

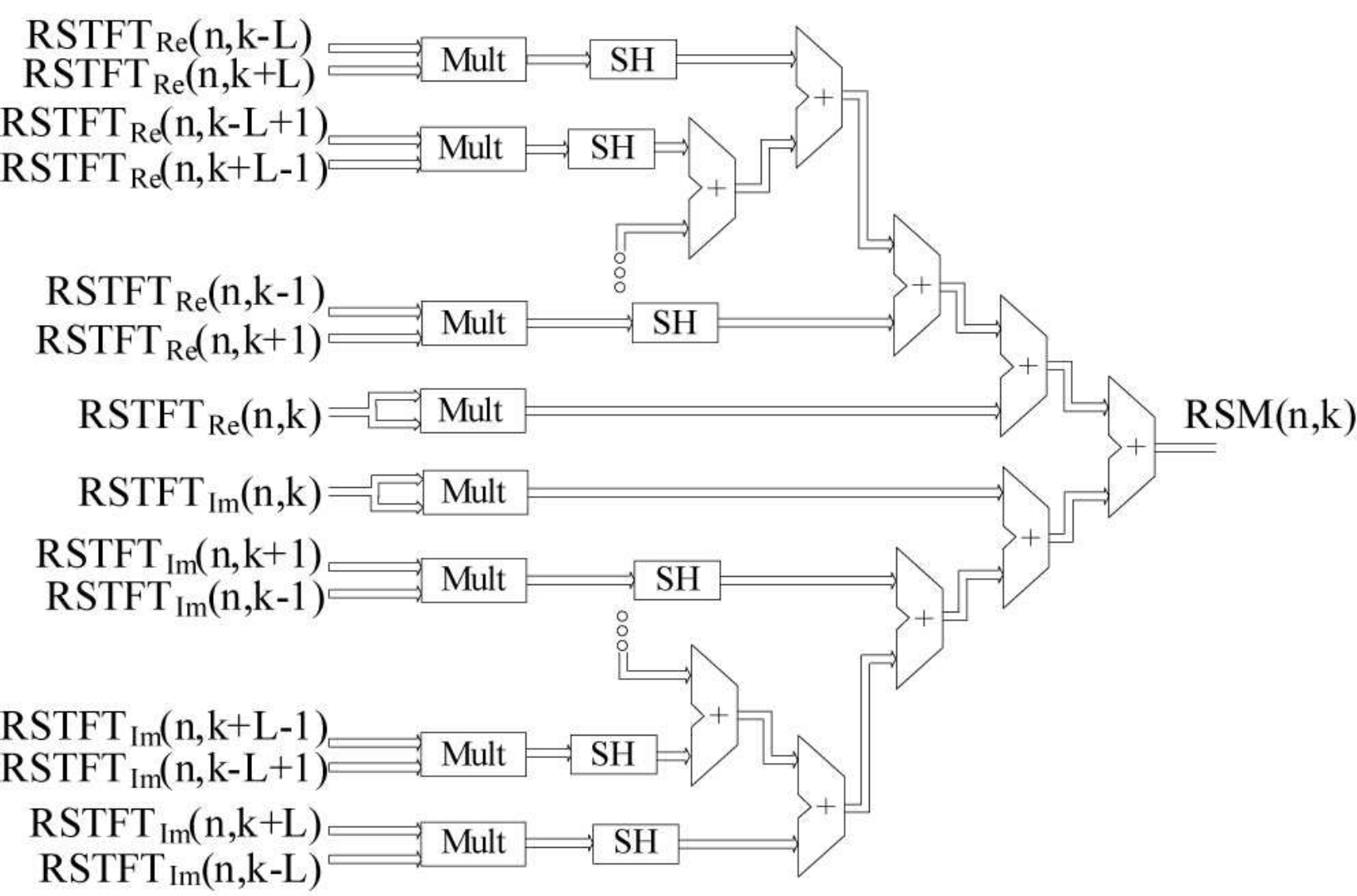

RSTFT Re(n,k-L)
RSTFT Re(n,k+L)
RSTFT Re(n,k-L+1)
RSTFT Re(n,k+L-1)
RSTFT Re(n,k-1)
RSTFT Re(n,k+1)
RSTFT Re(n,k)
RSTFT Im(n,k)
RSTFT Im(n,k+1)
RSTFT Im(n,k-1)
RSTFT Im(n,k+L-1)
RSTFT Im(n,k-L+1)
RSTFT Im(n,k+L)
RSTFT Im(n,k-L)
Mult
SH
+
RSM(n,k)

Figure 7

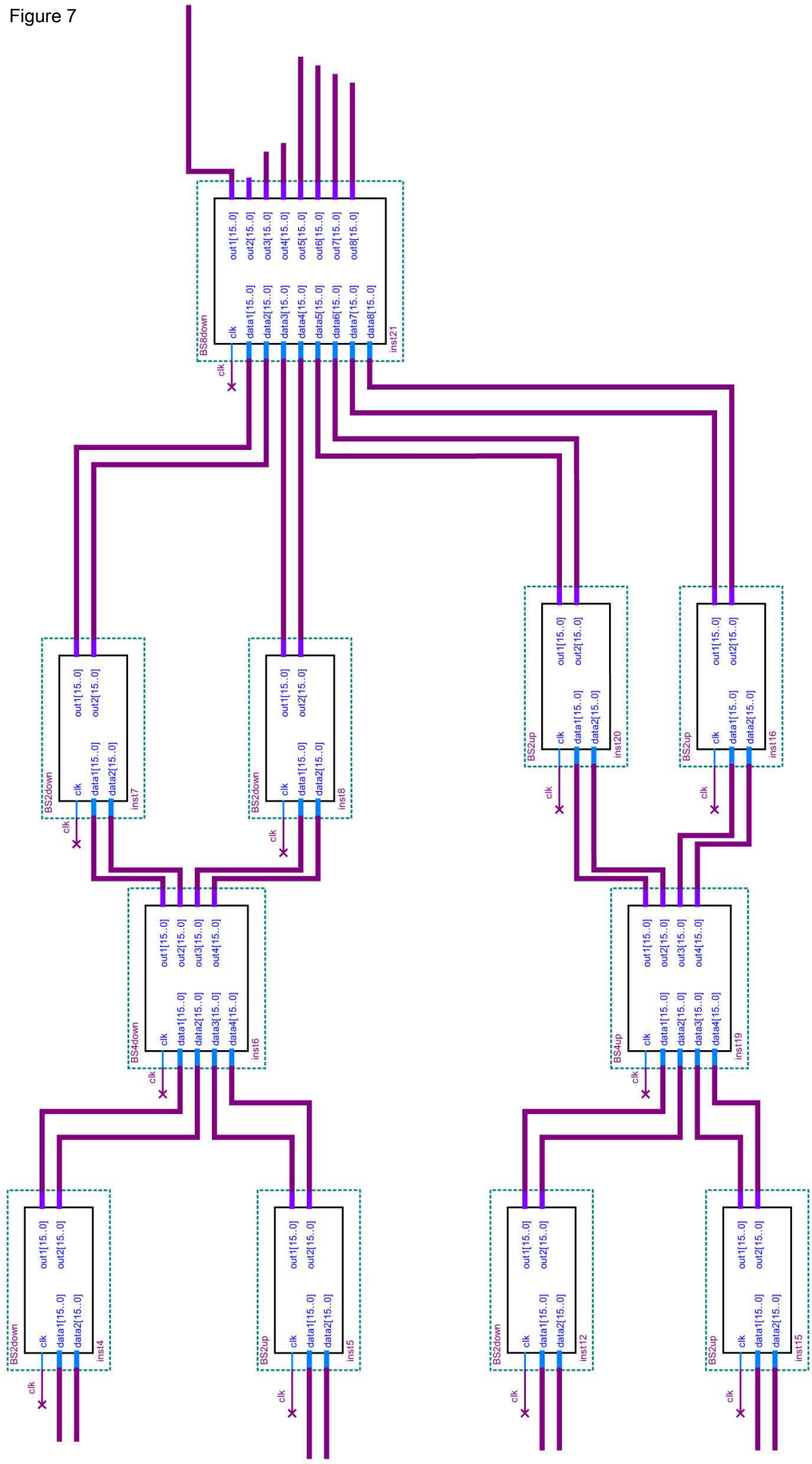

BS8down
inst21
out1[15..0]
out2[15..0]
out3[15..0]
out4[15..0]
out5[15..0]
out6[15..0]
out7[15..0]
out8[15..0]
clk
data1[15..0]
data2[15..0]
data3[15..0]
data4[15..0]
data5[15..0]
data6[15..0]
data7[15..0]
data8[15..0]
BS2down
inst7
inst8
BS2up
inst20
inst16
BS4down
inst6
BS4up
inst19
inst4
inst5
inst12
inst15

Figure 8

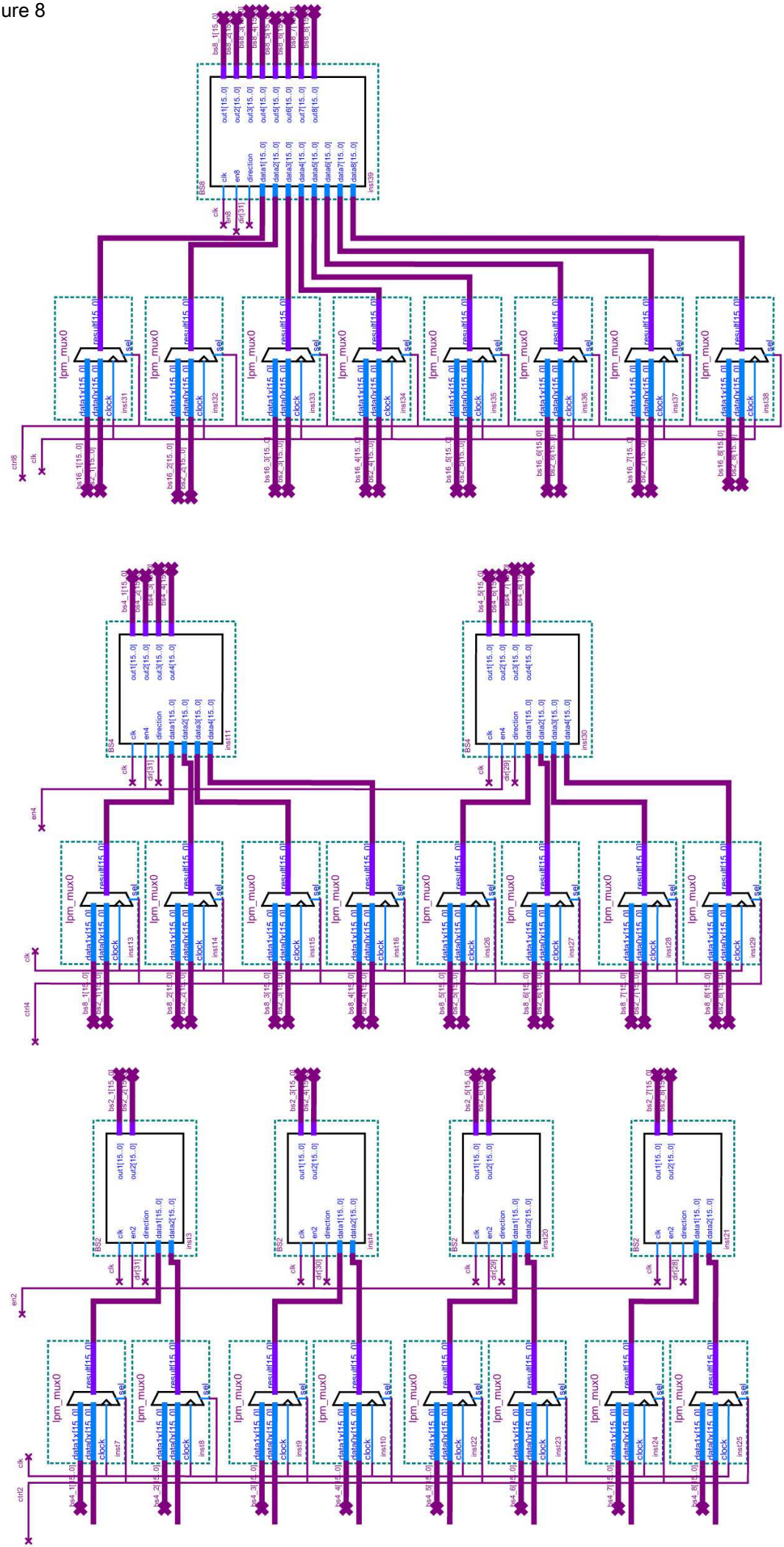

BS8
BS4
BS2
lpm_mux0
clk
en8
en4
en2
sel
result[15..0]
data1x[15..0]
data0x[15..0]
clock

Figure 9

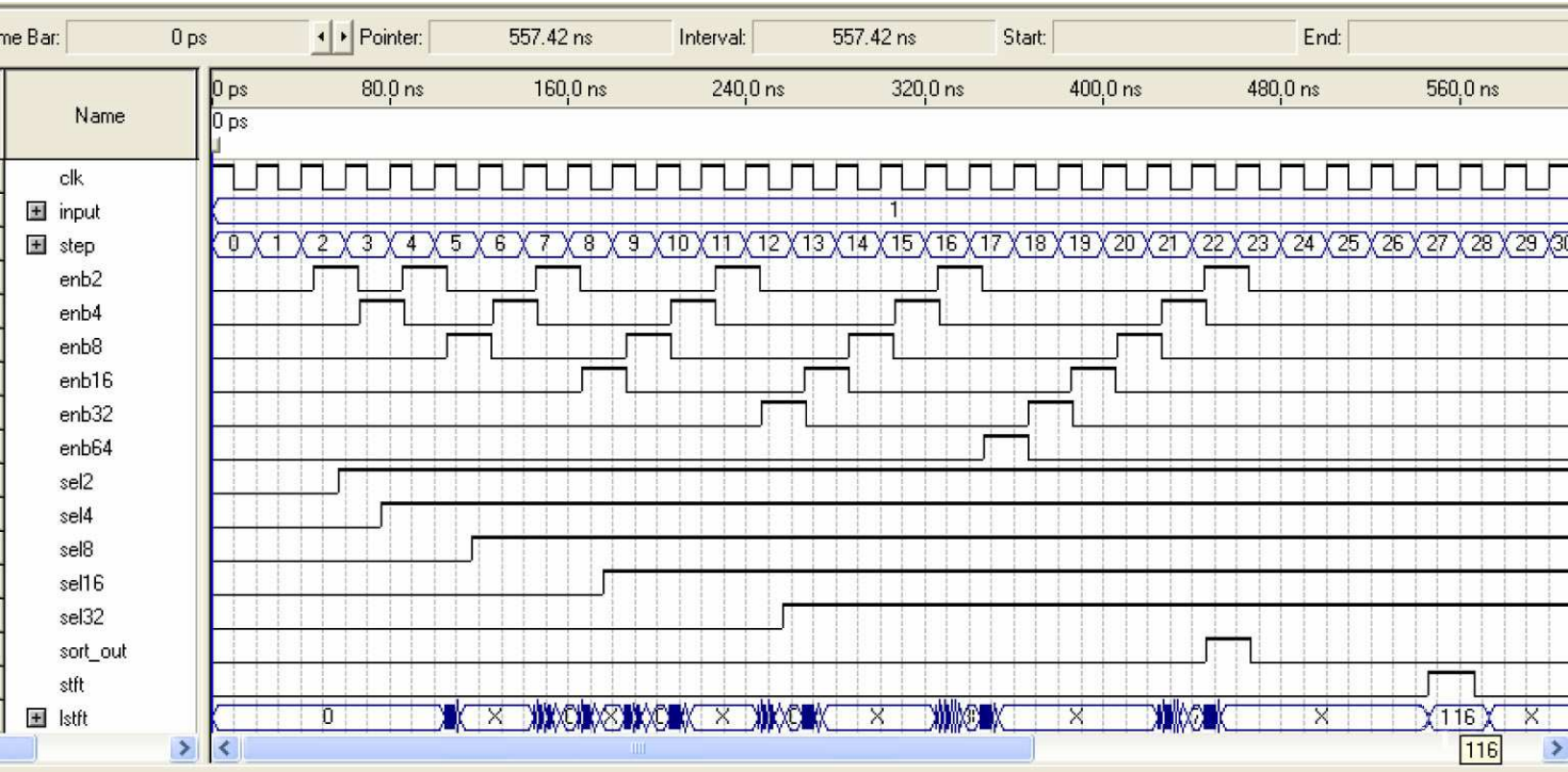

me Bar:
0 ps
Pointer:
557.42 ns
Interval:
557.42 ns
Start:
End:
Name
0 ps
80.0 ns
160.0 ns
240.0 ns
320.0 ns
400.0 ns
480.0 ns
560.0 ns
clk
input
step
enb2
enb4
enb8
enb16
enb32
enb64
sel2
sel4
sel8
sel16
sel32
sort_out
stft
lstft
116

Figure 10