Upgrading FPGA Implementation of Isolated Word Recognition System for a Real-Time Operation

T. Sledevic[1], G. Tamulevicius[1], D. Navakauskas[1]
[1]*Department of Electronic Systems, Vilnius Gediminas Technical University, Naugarduko St. 41–422, LT-03227 Vilnius, Lithuania*
dalius.navakauskas@vgtu.lt

***Abstract*—The article reports on the upgrading of the FPGA based isolated word recognition system for real-time tasks. All recognition system components (except some feature calculation steps) were implemented using VHDL. Some high precision calculations were implemented on soft core processor. The employed Dynamic time warping algorithm was speeded-up 2.8 times by restricting the calculated error matrix size. This enabled us to reduce the average word recognition time to 12.81 ms. Linear predictive coding, linear predictive coding cepstral and linear frequency cepstral coefficients feature analyses were investigated for 100 Lithuanian word recognition. In speaker dependent experiments linear predictive coding cepstral analysis gave the highest average recognition rate of 95 % and the highest robustness to white noise in speech. 15 dB noise level lowered average recognition rate to 86.2 %.**

***Index Terms*—Cepstral analysis, dynamic time warping, field programmable gate array, intellectual property core, isolated word recognition, linear predictive coefficients.**

I. INTRODUCTION

Despite of recent software-based Lithuanian speech recognition [1] and synthesis [2] implementations on personal computers and servers there is an unaddressed need of embedded systems for mobile and stand-alone devices, interactive voice controlled systems, disabled person equipment, etc. Embedded systems bring in their specific requirements for speech recognizers: the limited speed of processing and of memory, the low power consumption.

The recognition of large vocabulary and continuous speech requires complicated algorithms with huge amounts of calculations, large quantities of memory [3], [4]. This can result in enlarged power consumption, longer recognition time and higher recognition error rate.

Many automatic speech recognition systems for the languages of minor use are now developed. Presented in [5] Croatian speech recognizer uses acoustic models based on context-dependent triphone hidden Markov models (HMM) and phonetic rules. Experimentally it is shown that the system can be used for speech recognition with word error rate below 5 %. In [6] a speaker independent speech recognition system for Estonian language is described. Clustered triphones with Gaussian mixture components are used there for acoustic modelling. The error rate of the system is improved to 27.3 %. In [7] Finnish speech recognition based on sub-word decoders is presented. The pursued task was to find the most probable HMM state sequence. The word error rate there is decreased up to 32 % for very large vocabulary. For Czech speech recognition in [8] investigation on usability of publicly available n-gram corpora to create Czech language models is carried out. The experiments on large test data illustrate the impact of Czech as highly inflective language on the perplexity. The best achieved average error rate is 20 %. The multilingual Italian – Lithuanian small vocabulary speech recognition is implemented using multilingual transcription in [9]. The average recognition accuracy of ten spoken numbers for the Lithuanian language is 93 % and for the Italian – 98 %. It is important to acknowledge that here analysed all speech recognizers are implemented in software.

In some cases the robustness and correctness of recognition, together with the low power consumption are preferred against the size of the vocabulary [10]. Then the natural choice is an isolated word recognition approach leading to lower hardware requirements: much smaller vocabulary (less memory), simpler classification (lower speed and power consumption), potentially higher recognition rate (correctness), and at the same time an ability to use advanced noise cancellation (robustness) [11].

The field programmable gate array (FPGA) platform lets to employ the parallelization and pipelining technique in speech recognition. It is a flexible architecture to develop systems in comparison to implementations on the ASIC devices. The embedded processor-based solutions on the market have an average 80 % recognition rate and limited size of the dictionary: 32 (EasyVR [12]) or 75 (NLP [13]) commands. The main issue in such recognizer is to ensure real-time requirements for the feature extraction (especially in comparison stages). Contrary for this approach in FPGA [3], [4], [10], [14] or GPU [15], [16] implementation feature extraction and matching processes can run independently.

First softcore implementations on Virtex-4 family FPGA of Lithuanian isolated word recognizer were done by this article authors in [17]. The use of soft-core processor

Manuscript received February 27, 2013; accepted June 09, 2013.
This research was funded by a grant (No. MIP-092/2012) from the Research Council of Lithuania.

Microblaze together with intellectual property (IP) cores for signal processing enabled us to accelerate word recognition process by 1.55 times [18], [19], but it was still not enough for a real-time operation.

The paper presents upgraded FPGA implementation of isolated word recognition system for a real-time operation. In Section II isolated word recognition algorithm and a new way of its parallel execution are presented. In Section III numerous original soft-core implementations of individual recognition stages are described in details. In Section IV results on extensive experimental investigation of created isolated word recognition system working in a real-time are summarized. General conclusions are stated in Section V.

II. Isolated Word Recognition Procedure

The proposed in [18] and adopted here isolated word recognition algorithm (Fig. 1) is shortly described below.

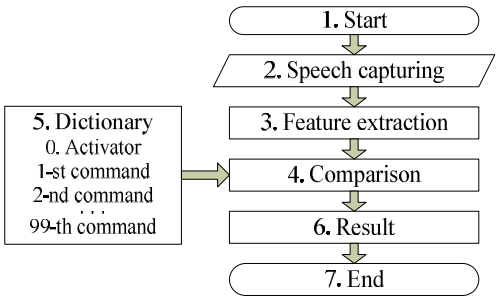

Fig. 1. Isolated word recognition algorithm.

1. When system is turned on the audio chip is configured to sample the incoming speech signal at 44100 Hz. If real-time command recognition is not selected then system stays in idle mode and is waiting for the press of start button;

2. After the start button is pushed speech signal is quantized in 8 bits and saved in four memory buffers coherently. Each buffer contains 256 data samples of speech signal (23.22 ms duration). There is no need to save the whole isolated word signal because features are extracted in real-time immediately after last voice signal sample is written into buffer. For the testing purposes there is an option to read the speech signal directly from memory card;

3. Because of buffers partial overlapping speech features are extracted every 11.61 ms. Three feature sets can be alternatively extracted: linear predictive coefficients (LPC); linear predictive cepstral coefficients (LPCC); linear frequency cepstrum coefficients (LFCC). The 12th order linear prediction is calculated by Levinson-Durbin recursion, while cepstrum coefficients are estimated by

$$C = \text{real}\left(\text{FFT}\left(\log_2\left|\text{FFT}\left(X\right)\right|\right)\right), \quad (1)$$

where C – cepstrum coefficients vector; X – discretized speech signal vector; FFT – fast Fourier transform;

4. Comparison of two feature sequences is implemented through dynamic time warping algorithm (DTW) [2], [17], [18]. DTW compares words with duration up to 1.5 s;

5. Each vector in dictionary can be updated by pronouncing a new word in the microphone and saving the signal at the desired command address. The dictionary is implemented in FPGA internal block RAM. Thus the initial dictionary is programmed together with FPGA content file;

6. After the test word features are compared with reference features, a reference number of the best matching word is displayed on LCD.

The steps 2–4 are performed in a parallel manner. The system constantly captures speech, extracts features and compares them with activating reference features saved in address No. 0. If activation word is recognized, then system generates a short acoustic signal. Afterwards speaker must pronounce a command.

III. IP Core Implementations

The steps of isolated word recognition algorithm are implemented in hardware using VHDL. Soft-core processor is used only for Levinson-Durbin and LPCC calculation. The whole algorithm is divided into modules (see Fig. 2).

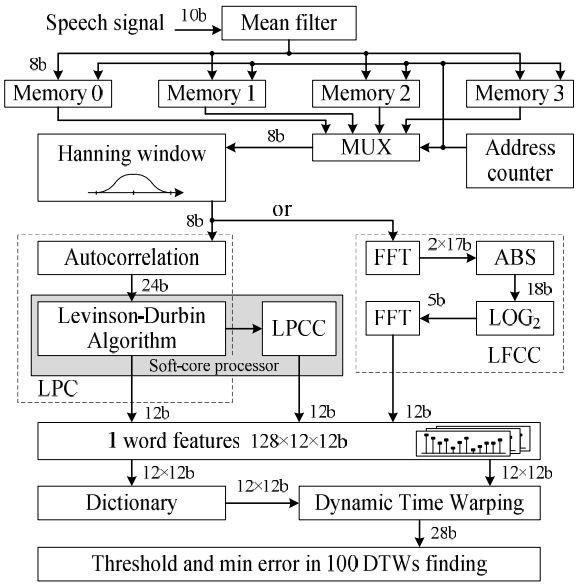

Fig. 2. Block diagram of isolated word recognition.

The speech signal is sampled at 44100 Hz intentionally. 11025 Hz sampling rate is achieved using mean filter with the length of 4 samples. Four bank memories are used to save 8 highest bits of the signal in partially overlapped frames. The address counter controls in which bank and address the incoming data must be stored. At the end of frame the signal from one bank is directed to Hanning window module. The linear prediction and cepstrum calculations are implemented in different files. LPC are calculated using the autocorrelation and the recursive Levinson-Durbin algorithms. LPCC are recalculated from LPC. The LFCC are calculated using two FFT, ABS and LOG_2 units. The 12 highest bits forms feature value. One reference word feature array contains 128 × 12 × 12 b. The DTW algorithm is used for comparison of feature sequences (words). For each compared word DTW outputs a 28 b width matching error. Minimal error finding unit compares incoming errors one-by-one and refreshes the minimal value corresponding to the ordering number of the reference word in the dictionary. Additional error threshold value is used in order to prevent misidentification of activator.

Module 1. The 12th order autocorrelation algorithm is

implemented using 13 shift registers, as shown in Fig. 3.

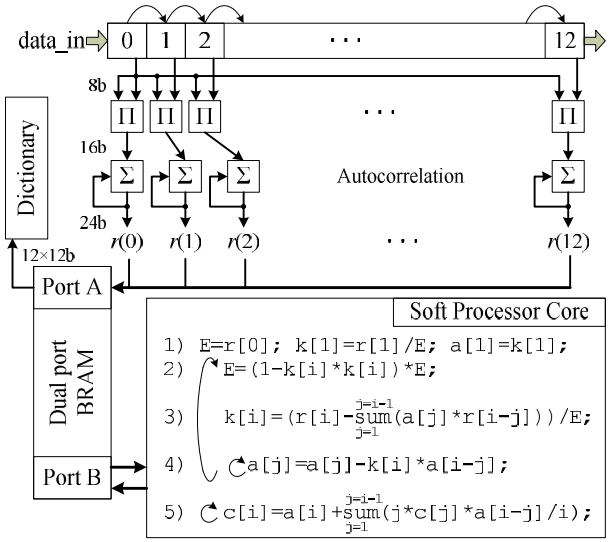

Fig. 3. Implementation of autocorrelation, Levinson-Durbin and LPCC calculation algorithms.

Each element of this register is accessible in parallel by 13 DSPs slices. The multiplication result is accumulated until the last byte No. 256 reaches the shift register. Autocorrelation coefficients $r(i)$ are stored in dual port BRAM buffer via port A. This memory is used to exchange the data between hardware and software parts of the system. For the soft-core processor the autocorrelation coefficients are accessible via port B. At the first iteration of Levinson-Durbin algorithm the values of energy coefficient E, first reflection coefficient $k(1)$ and first LPC are initialized. For each i^{th} iteration energy coefficient E is updated and used as divisor for calculation of reflection coefficient $k(i)$. The steps 2–4 are repeated 12 times giving the 12^{th} order LPC. At step 5 the LPCC are calculated from LPC. The soft-core processor calculates both types of coefficients and returns them to BRAM buffer for storing in the dictionary.

Autocorrelation implementation can be parallelized, but Levinson-Durbin algorithm is fully sequential. Therefore it can be described by finite state machine (FSM, see Fig. 4).

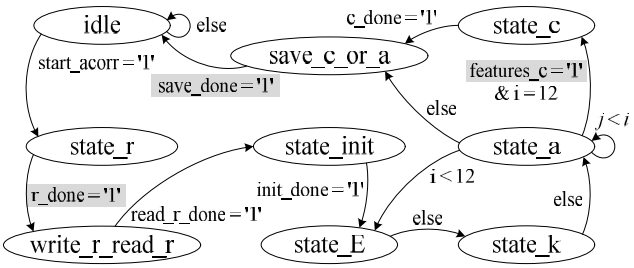

Fig. 4. Finite state machine of LPC and LPCC calculation algorithm.

FSM has 9 states that are used to synchronize the calculation process. Grey rectangles in FSM marks the additional signals used for communication between hardware and software. Signal `r_done` starts interrupt process of recursive Levinson-Durbin algorithm. Signal `features_c` allows switching between LPC and LPCC analysis. Signal `save_done` puts soft-core processor into a sleep mode (features are saved in the dictionary).

Module 2. LFCC extraction (Fig. 2) is based on cepstrum calculation (1). Spectrum calculation uses Radix-2 based FFT core [20]. Fast $\log_2$ operation is implemented using binary mask filters (Fig. 5). When incoming data match certain mask, LOG_2 module returns number of matched filters, which represent rounded integer value of $\log_2$.

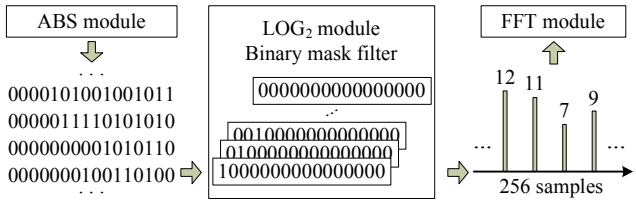

Fig. 5. Implementation of fast $\log_2$ operation.

Module 3. The big issue in hardware is the implementation of sequential algorithm, which requires accessing the data located in memory at variant addresses [21]. DTW algorithm is sequential by nature. The principle used to fill the error matrix is shown in Fig. 6. The features of test word are located on x axis, reference features are located on y axis.

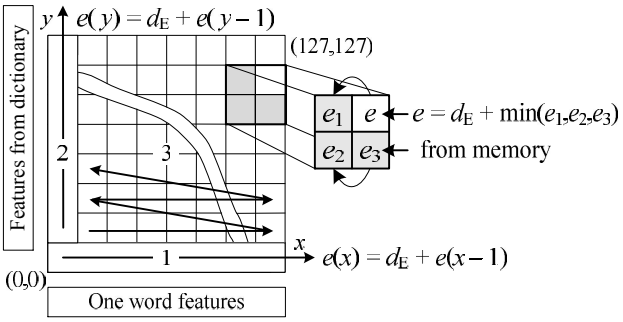

Fig. 6. The principle of filling the error matrix.

At the first step error values $e(x)$ are calculated for first line in positive x direction. At the second step errors $e(y)$ in first column are calculated. Finally the error in rest of the area is estimated using 2 × 2 size sliding filter (Fig. 6). Filter moves from left to right and from bottom to top until reaches last error matrix element located at address (127, 127). Using this filter the minimum value of 3 neighbours marked in grey is added to Euclidean distance for element e and saved in error matrix memory at e address. IP core implementation of this procedure is shown in Fig. 7.

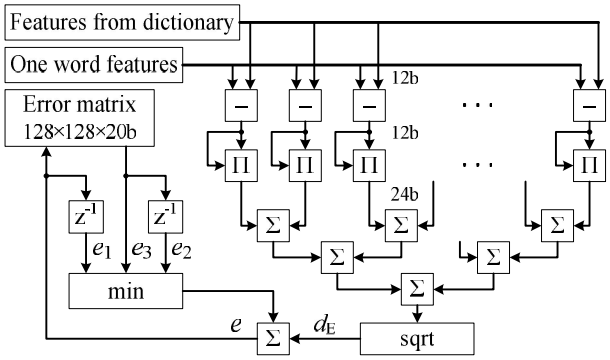

Fig. 7. Implementation of Euclidean distance and error matrix filling.

The block diagram presents fast Euclidean distance estimation using 11 summations, 12 subtractions, 12 multiplexers and one square root core. The square root core has pipelined implementation. The calculation of Euclidean distance d_E is done per one clock cycle.

The difference between vectors is estimated by

$$d_{\mathrm{E}} = \sqrt{\sum_{i=1}^{12} \left(c_{\mathrm{w}}(i) - c_{\mathrm{d}}(i)\right)^2}, \quad (2)$$

where d_{E} – Euclidean distance; $c_{\mathrm{w}}(i)$ – i-th LPC, LPCC or LFCC in one word feature memory; $c_{\mathrm{d}}(i)$ – i-th LPC, LPCC or LFCC in dictionary.

To ensure synchronous filling of error matrix the data from dictionary, one word buffer and error matrix are read out in parallel. Because the BRAM memory has one clock reading latency the valid address must be prepared before accessing the data. Therefore the address counter is implemented as separate process independent of the error calculation process. These two address and error estimation processes runs synchronous. The advantage of hardware implementation is the pipelined calculation of error matrix. At each rising edge of clock new error e value is calculated and stored in error matrix memory immediately.

The back-path searching algorithm uses the same 2×2 size sliding filter to find a path with minimum error from (127, 127) to (0, 0). The DTW algorithm is repeated for every reference feature sequence in the dictionary. The minimal error path value indicates the recognized word in the dictionary. The threshold can be applied additionally for the minimal error value in order to reject erroneously recognized word. The back-path searching area is constrained (Fig. 8).

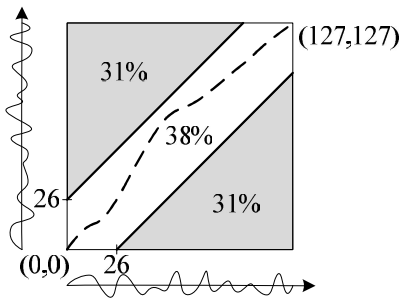

Fig. 8. The principle of the border constraints used in the error matrix.

The grey triangles mark the region where the error is not computed. Only 38 % of error matrix is used. Such restriction speeds-up the DTW calculation up to 2.6 times without negative influence on recognition accuracy. The acceleration value is obtained by comparing the recognition results for the DTW without and with border constraints.

IV. RESULTS

The implemented Lithuanian word recognition system was tested experimentally. Isolated words were given from computer as *.wav signals via memory card to FPGA board. All recognition experiments were speaker dependent. The records of isolated words with narrowband noise were recorded in office environment. 100 different words were pronounced 4 times by 5 male and 5 female speakers. The system was trained using the first session of records and the rest 3 sessions of records were used for testing. The robustness of feature systems was tested adding white noise to original records obtaining 30 dB and 15 dB signal-to-noise ratio (SNR) values. The recognition rates for all speakers at different SNR values are given in Table I. Each value in the table is obtained by averaging recognition rates for one (M_i – male, F_i – female) speaker over 3 sessions.

TABLE I. ISOLATED WORD RECOGNITION RATES (IN PERCENTS).

Features extraction method		**LFCC**	**LPCC**	**LPC**
Speaker	i	$M_i \mid F_i$	$M_i \mid F_i$	$M_i \mid F_i$
Original record with narrowband noise	1	99 | 95	99 | 97	91 | 93
	2	97 | 86	95 | 89	85 | 87
	3	91 | 95	90 | 98	73 | 92
	4	84 | 99	77 | 99	68 | 97
	5	89 | 95	79 | 92	57 | 82
Record with white noise, SNR = 30 dB	1	96 | 95	94 | 97	79 | 89
	2	94 | 85	91 | 85	82 | 81
	3	79 | 93	77 | 93	57 | 92
	4	83 | 98	84 | 99	66 | 95
	5	91 | 94	88 | 90	64 | 81
Record with white noise, SNR = 15 dB	1	91 | 67	93 | 87	82 | 72
	2	72 | 65	85 | 74	64 | 66
	3	65 | 72	72 | 83	48 | 73
	4	69 | 81	79 | 98	57 | 87
	5	75 | 67	77 | 89	56 | 66

The recognition rate highly depends on individual speaker pronunciation. Male M_1 has highest recognition rate for all tested male speaker over all used features extraction methods and SNR values. Female F_4 speaker has relatively highest rate over all tested female speakers. The recognition rate for speakers M_1 and F_4 is decreasing relatively slower in comparison with other speakers. This was influenced by articulation of separate word and by velocity of speaker pronunciation. The misrecognitions in most cases appear for words with similar constellation of phonemes: "dalis → šalis". "visas → viskas", "įmonė → priemonė", "kaip → kiek", "laikas → vaikas", "kartas → darbas", "tirti → dirbti". Therefore elimination of similar words from recognizer dictionary would give higher recognition rate in our case.

The summarized recognition rates with confidence intervals (in brackets) for all feature systems and SNR values are given in Table II. The rates are averaged separately for 5 male and 5 female speakers. The confidence intervals were estimated with the confidence level of 95 %. The highest average recognition rate of 95 % is achieved for original records of female speaker using LPCC. The LFCC gives highest recognition rate of 92 % for male speaker while using original record with narrowband noise. Results are graphed in Fig. 9.

TABLE II. AVERAGE RECOGNITION RATES (IN PERCENTS).

Features extraction method	**LFCC** M, F	**LPCC** M, F	**LPC** M, F
Original record with narrowband noise	92.0 (–1.49 +1.32)	88.0 (–1.76 +1.61)	74.8 (–2.31 +2.20)
	94.0 (–1.32 +1.15)	95.0 (–1.23 +1.05)	90.2 (–1.62 +1.46)
With white noise, SNR = 30 dB	88.6 (–1.72 +1.57)	86.8 (–1.82 +1.68)	69.6 (–2.44 +2.35)
	93.0 (–1.41 +1.24)	92.8 (–1.43 +1.26)	87.6 (–1.78 +1.63)
With white noise, SNR = 15 dB	74.4 (–2.32 +2.21)	81.2 (–2.08 +1.96)	61.4 (–2.60 +2.54)
	70.4 (–2.42 +2.33)	86.2 (–1.86 +1.71)	72.8 (–2.36 +2.26)

The recognition rates hit in the overlapped confidence intervals while using LFCC and LPCC features for both

male and female speaker at original record and record with 30 dB SNR. The LPCC gave the highest robustness in comparison with LPC and LFCC features at 15 dB SNR. LPC demonstrated the lowest recognition accuracy during all experiments.

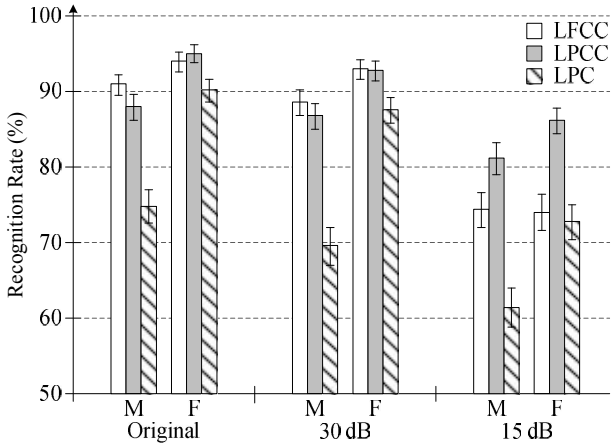

Fig. 9. The average recognition rate of word pronounced by male (M) and female (F) speaker at different SNR values using LFCC, LPCC and LPC as features extraction methods.

The FPGA resource utilization rate used in system is shown in Table III. LPC and LPCC feature extraction algorithms were implemented on soft-core processor. Therefore it uses 3 DSPs slices only for sequential signal processing. Soft-core uses 8 BRAMs to store instructions code and one BRAM to exchange data with the hardware part of system. Error matrix utilizes 18 BRAMs in DTW module. 100 BRAMs are used to store word feature vectors. 12 DSPs slices are used for implementation of fast Euclidean distance calculation.

TABLE III. FPGA UTILIZATION RATE.

Module	SLICEs	LUTs	BRAMs	DSPs
Soft-Core (LPC, LPCC)	1980 (13 %)	2503 (8 %)	9 (5 %)	3 (2 %)
LFCC	1572 (10 %)	2779 (9 %)	14 (7 %)	9 (5 %)
DTW	2567 (17 %)	4365 (14 %)	18 (9 %)	12 (6 %)
Others	1550 (10 %)	2808 (9 %)	100 (52 %)	0 (0 %)

The relative utilization of FPGA modules is shown in Fig. 10. Only 50 % of FPGA logic memory (SLICEs) is used. The remaining amount of the memory can be used to duplicate few DTW units with the aim to make the comparison process at least 3 times faster.

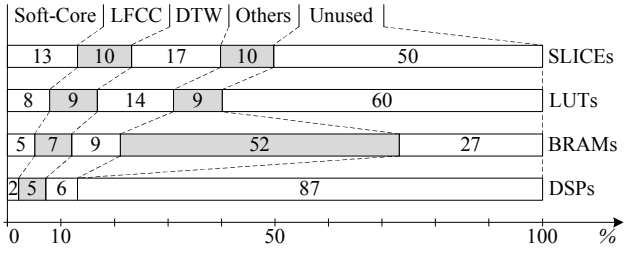

Fig. 10. The relative utilization of FPGA modules.

The duration of signal processing in main modules is shown in Table IV. In order to run recognition in real-time the feature extraction delay must be less than 11.61 ms. All feature extraction modules enable real-time operation. Duration of LPCC calculation is more than 50 times longer in comparison with LFCC calculation time. The reason is the linear prediction implementation on soft-core processor. The autocorrelation, LFCC and DTW modules run on 50 MHz, soft processor core runs on 100 MHz clock frequencies.

TABLE IV. SIGNAL PROCESSING DURATION.

Module	Pulses	Clock, MHz	Delay, µs	11610/Delay
LFCC	3320	50	66.4	174.85
Autocorrelation	270		5.4	2150.00
Levinson-Durbin	166207	100	1662.1	6.99
LPC	166477		1667.5	6.96
LPC → LPCC	168201		1682.0	6.90
LPCC	334678		3349.5	3.46
One DTW	16650	50	333.0	34.86
One DTW_c	6404		128.1	90.63

The time-line diagram of feature extraction and comparison stages is shown in Fig. 11. The gray and white rows mark the amount of time needed for FPGA-based and CPU-based calculations respectively (Matlab was used as software running on personal computer with 50 % usage of 3 GHz CPU). CPU-based LPCC and LPC extraction is more than 10 and 5 times faster respectively. The higher calculation speed is strongly influenced by 30 times higher CPU clock frequency. The advantage of FPGA against CPU is observed in LFCC and DTW calculations. FPGA-based LFCC runs 1.5 times faster than CPU-based because of simplifications in logarithm and FFT core employment. FPGA-based DTW and DTW_c calculation is speed-up more than 280 times in comparison with same algorithm implemented in Matlab.

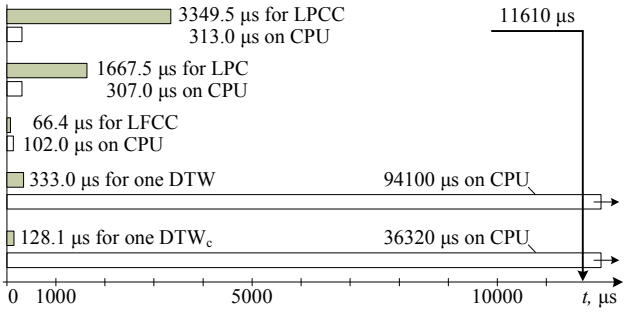

Fig. 11. The time-line diagram of feature extraction and comparison stages based on FPGA and CPU.

The delay of signal processing is very convenient to present in time-line, as shown in Fig. 12.

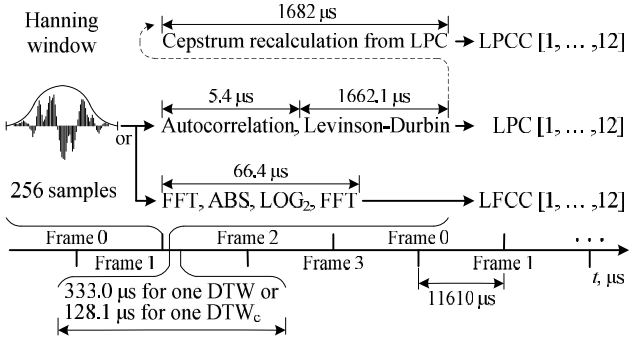

Fig. 12. Time-line of LFCC, LPC, LPCC and DTW calculation.

Calculation of LFCC features for one word takes 66.4 µs.

This is 174.8 times faster than real-time recognition requires. LPCC feature extraction takes 3349.5 μs and it is 3.46 times faster that real-time operation requirement. Therefore one soft-core processor can be used for LPCC analysis of signals from 3 different data channels simultaneously. The delays for one DTW and DTW_c process are 333.0 μs and 128.1 μs respectively. In non real-time mode the whole set of reference words are sequentially analyzed by DTW (or DTW_c) algorithm and it takes 33.30 ms (12.81 ms in DTW_c case). This time-span does not enable real-time recognition because only 34 DTW (or 90 DTW_s) comparisons will be performed in 11.61 ms. In order to make system more applicable and usable with larger dictionaries it is proposed to use activation word to initiate the recognition process. Recognition of this activator will require only one DTW comparison in real-time. When activator is recognized the system captures pronounced utterance and recognizes it in non-real-time mode.

The fastest version of software-based comparison process [18] is speeded-up 403 times using DTW algorithm and 1049 times using DTW_c algorithm. LFCC feature extraction is speeded-up 160 times. The average recognition rate is improved by 1 % (up to 94 %) using LFCC and by 2 % (up to 95 %) using LPCC features. Other researchers declare similar recognition rates of 90–93 % on hardware-based speech recognizers [3], [4], [10], [14], [22].

V. CONCLUSIONS

FPGA implementation of isolated word recognition system was upgraded for real-time operation. Comparing with previous recognition system implementation feature extraction process is speeded-up 160 times, word comparison process – 403 times. The comparison process is accelerated 2.6 times additionally by employing constraints in calculated error matrix thus giving 1049 times faster comparison process. Further acceleration is achieved implementing square root operation per one clock cycle. The final achieved speed of word comparison is 7800 words per second.

The implemented system was tested using Lithuanian language words. The system attained 95 % accuracy using linear predictive coding cepstral analysis. It was more robust to white noise and outperformed linear frequency cepstral coefficients and linear predictive coding analysis giving 86.2 % recognition rate at 15 dB signal-to-noise ratio. Higher recognition accuracy can be expected by establishing optimal analysis parameters, optimizing dictionary structure.

The dynamic time warping with constrains module can be additionally accelerated by improving hardware architecture with the aim to increase clock frequency. The future work is to increase the dictionary size and to duplicate few dynamic time warping cores with the aim to take advantage of speed.

REFERENCES

[1] R. Lileikyte, L. Telksnys, "Quality estimation methodology of speech recognition features", *Elektronika ir elektrotechnika* (*Electronics and Electrical Engineering*), vol. 110, no. 4, 2011, pp. 113–116.

[2] G. Pyz, V. Simonyte, V. Slivinskas, "Lithuanian speech synthesis by computer using additive synthesis", *Elektronika ir elektrotechnika* (*Electronics and Electrical Engineering*), vol. 18, no. 8, 2012, pp. 77–80.

[3] J. Choi, K. You, W. Sung, "An FPGA implementation of speech recognition with weighted finite state transducers", in *2010 IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, 2010, pp. 1602–1605. [Online]. Available: http://dx.doi.org/10.1109/ICASSP.2010.5495538

[4] R. Veitch, L. M. Aubert, R. Woods, S. Fischaber, "Acceleration of HMM-based speech recognition system by parallel FPGA Gaussian calculation", in *2010 VI Southern Programmable Logic Conf.*, Mar. 2010, pp. 197–200. [Online]. Available: http://dx.doi.org/10.1109/SPL.2010.5483010

[5] S. Martincic-Ipsic, M. Pobar, I. Ipsic, "Croatian large vocabulary automatic speech recognition", *Automatika*, vol. 52, no. 2, 2011, pp. 147–157.

[6] P. Sojka, I. Kopecek, K. Pala, "Large vocabulary continuous speech recognition for Estonian using morphemes and classes", *Lecture Notes in Computer Science*, Berlin: Springer, 2004, pp. 245–252.

[7] T. Hirsimaki, M. Kurimo, "Decoder issues in unlimited Finnish speech recognition", in *Proc. NORSIG'04*, 2004, pp. 320–323.

[8] V. Prochazka, P. Pollak, J. Zdansky, J. Nouza, "Performance of Czech speech recognition with language models created from public resources", *Radioengineering*, vol. 20, pp. 1002–1008, Dec. 2011.

[9] R. Maskeliunas, A. Esposito, "Multilingual Italian – Lithuanian small vocabulary speech recognition via selection of phonetic transcriptions", *Elektronika ir elektrotechnika* (*Electronics and Electrical Engineering*), vol. 121, no. 5, 2012, pp. 85–88.

[10] G. Zhang, J. Yin, Q. Liu, Ch. Yang, "A real-time speech recognition system based on the implementation of FPGA", in *Proc. Cross Strait Quad-Regional Radio Science and Wireless Technology Conf.*, July 2011, pp. 1375–1378. [Online]. Available: http://dx.doi.org/10.1109/CSQRWC.2011.6037220

[11] L. Stasionis, A. Serackis, "Selection of an optimal adaptive filter for speech signal noise cancellation using C6455 DSP", *Elektronika ir elektrotechnika* (*Electronics and Electrical Engineering*), vol. 115, no. 9, 2011, pp. 101–104.

[12] A. Chakravarty, *Speech recognition toolkit for the Arduino*, 2013. [Online]. Available: http://arjo129.github.com/uSpeech/

[13] *Natural Language Processor*, Sensory, 2010. [Online]. Available: http://www.sensoryinc.com/products/NLP-5x.html

[14] S. T. Pan, C. C. Lai, B. Y. Tsai, "The implementation of speech recognition systems on FPGA-based embedded systems with SoC architecture", *Int. Journal of Innovative Computing, Information and Control*, vol. 7, no. 11, 2011, pp. 6161–6175.

[15] D. Sart, A. Mueen, W. Najjar, V. Niennattrakul, E. Keogh, "Accelerating dynamic time warping subsequence search with GPUs and FPGAs", in *2010 IEEE 10th Int. Conf. on Data Mining*, Dec. 2010, pp. 1375–1378.

[16] Y. Zhang, K. Adl, J. Glass, "Fast spoken query detection using lower-bound dynamic time warping on graphical processing units", in *2012 IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, March 2012, pp. 5173–5176. [Online]. Available: http://dx.doi.org/10.1109/ICASSP.2012.6289085

[17] V. Arminas, G. Tamulevicius, D. Navakauskas, E. Ivanovas, "Acceleration of feature extraction for FPGA based speech recognition", in *Proc. SPIE 7745, Photonics Applications in Astronomy, Communications, Industry, and High-Energy Physics Experiments 2010*, pp. 774511–774511-6. [Online]. Available: http://dx.doi.org/10.1117/12.872081

[18] G. Tamulevičius, V. Arminas, E. Ivanovas, D. Navakauskas, "Hardware accelerated FPGA implementation of Lithuanian isolated word recognition system", *Elektronika ir elektrotechnika* (*Electronics and Electrical Engineering*), vol. 99, no. 3, 2010, pp. 57–62.

[19] E. Ivanovas, "Development and implementation of means for word duration signal processing", Ph.D. dissertation, Dept. of Electronic Systems., Vilnius Gediminas Technical Univ., Vilnius, 2012.

[20] *Fast Fourier Transform Logicore*, Xilinx, 2011. [Online]. Available: http://www.xilinx.com/support/documentation/ip_documentation/xfft_ds260.pdf

[21] D. Mihhailov, A. Sudnitson, V. Sklyarov, I. Skliarova, "Acceleration of recursive data sorting over tree-based structures", *Elektronika ir elektrotechnika* (*Electronics and Electrical Engineering*), vol. 113, no. 7, 2011, pp. 51–56.

[22] O. Cheng, W. Abdulla, Z. Salcic, "Hardware-Software Codesign of Automatic Speech Recognition System for Embedded Real-Time Applications", *IEEE Trans. on Industrial Electronics*, 2011, pp. 850–859. [Online]. Available: http://dx.doi.org/10.1109/TIE.2009.2022520