ISSN 1392-2114 ULTRAGARSAS, Nr.1(34). 2000.

Architectures for ultrasonic delay time estimation tasks

E. Kazanavičius, R Venteris

Digital Signal Processing Laboratory, KUT

Studentų 50-214c, Kaunas LT3031, LITHUANIA.

Introduction

Ultrasonic pulse-echo time delay estimation (TDE) tasks are indispensable in the variety of ultrasonic measurement applications such as range metering, liquid and gas flow velocity metering, ultrasonic vision and tomography, NDT evaluations.

Although analog implementations exist for some TDE subtasks [4], recently these tasks also are being effectively solved by digital processing methods. Digital implementations are helpful due to perfect system repeatability, flexibility, robustness.

However, although the TDE concept and the constituent subroutines are more or less the same, a newly designed TDE system usually must have an individual architecture. The architecture should be designed according to specific application requirements such as signal lengths, required computation speed and precision, cost, available hardware resources. Consequently, searching for the optimal solution of this problem becomes the reason for extensive design period that also is subjected for reduction. Also certain system flexibility is needed at algorithm development and system prototyping phase.

Currently a variety of design resources are available. They are DSPs, general purpose processors (GPP), microcontrollers, hardware accelerators, application specific integrated circuits (ASIC), also field programmable gate arrays (FPGA) that latterly play important role for implementation of time and design area constrained tasks. These resources are well provided with the tools for hardware synthesis and software development. However, the problem exists – which platform to choose for a particular system design. The answer would be about the optimal mix of all of them. Nowadays system level hardware-software codesign methodologies [7] and tools are still being under research. They are subjected to reach such design parameters as minimal design area, cost, design period and power consumption by thus satisfying the requirement for execution speed.

Basically, the two objective functions are used:

1) Minimize architecture area S_h by retaining measuring cycle time constraint T_c not exceeded.

2) Minimize measuring cycle time t_p by retaining architecture area constraint S_c not violated.

Stringent T_c is the characteristic feature of applications where the investigated medium changes in time. For example, measuring cycle duration constraint is applied for mobile robot vision system in order to reach *real time* robot control [1], [6].

Therefore TDE applications fall into the domain of real time embedded and DSP systems. Following [7], we state TDE task being heterogeneous by *computational semantics* since can be described by control-flow and data-flow models and also by *implementation* since its individual nodes can be implemented in software either hardware.

The paper shows high variability of digital TDE architectures being restricted by T_c and S_c. We propose the general TDE task, present the analysis of its implementation and give several architecture examples.

Resources for heterogeneous system implementation

Heterogeneous system implementation may use a set of different purpose hardware resources. In our case we distinguish four resource groups:

1) *DSP processors* - suitable for low-data rate DSP, control and application specific algorithms. Most DSP processors have Harward architecture with a single fast multiplier-adder therefore at algorithm level performs sequential computations. This architecture is optimized for filters, Fourier transform, modulation by thus performing one multiplication-addition per cycle.

2*) FPGAs* – for high rate parallel DSP, bit level operations, system specific fast control and timing.

3) *microcontrollers* – for control and timing.

4) *communication devices and memory* – for internal and external communication, data buffering.

Recently FPGAs becoming capable for implementation of optional DSP processor or controller cores and cover needed control logics. Then the system could fit to one chip. However, logical distinguishing of processors and controllers *still makes sense* as a number of FPGA-ready DSP cores with plenty of software are available in VHDL. Meanwhile developing of a new processor architecture is rather complicated work.

Heterogeneous specification of the general TDE task

At the system level a task can be represented by *data and control flow graphs* (CDFGs). There was suggested a general CDFG covering the class of multi-channel TDE tasks. The CDFG represents nodes that indicate computation and control tasks, and edges indicate control, timing and data flows (Fig. 1). The subjects for the further analysis are nodes within coarse dashed rectangles.

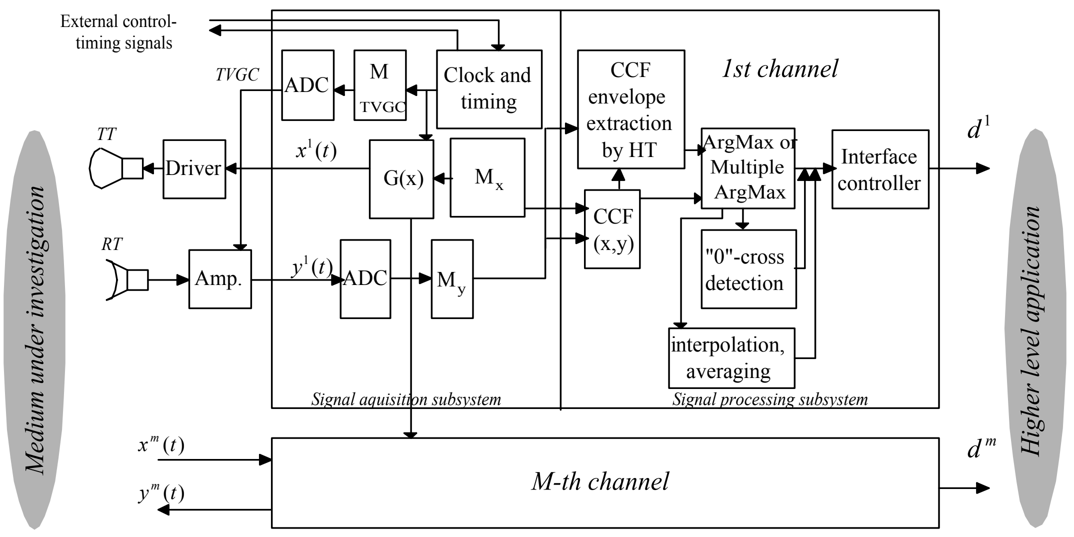

Fig. 1. **CDFG of the general TDE task.**

The CDFG consists of M independent measuring channels. For each channel the generator G feeds the transmitting transducer TT with the reference signal x from the memory M_x. The echo or transient signal y is accepted by the receiving transducer RT, digitized and stored in the memory M_y. $TVGC$ is time varying gain control performed by DAC and gain memory M_{TVGC}. Several channels can have the same clock and timing in order to work simultaneously. Also some external control-timing signals are provided that could be applied e.g. to drive some analog circuits.

After processing period T_{dsp} the computed vectors of time delays d^i are dumped out to the external system. The common signal processing nodes are cross-correlation function (CCF), CCF envelope extraction by use of Hilbert transform (HT), detection the maximal value argument of a signal (ArgMax), zero-cross detection. CCF is computed in order to get the signal peak at the signal occurrence moment. In some applications it is sufficient to find the time-delay of this peak by ArgMax [1], [6]. When $\mathbf{y}$ is noisy and more accurate delay should be measured then HT is used to extract CCF envelope and zero-cross detection to the derivative function of the envelope is applied. Also averaging, interpolation [5] can be used for better accuracy.

Certainly, for the particular application not every described node or edge is necessary but an optional combination of the nodes. The optional edges are drawn by fine dashed line.

Actually, the two logical independent subsystems in each channel can be distinguished:

Signal emission-acquisition subsystem (SEAS) which is related with real time ultrasonic scanning process and always characterized by fixed and stringent scan timing constraints T_{scan}. This subsystem has no DSP operations and control-timing is dominant in it.

Signal processing subsystem (SPS) is different from the SEAS since performs mainly data processing and much less control functions. For the SPS time constraint T_{dsp} is set. For the system which performs scanning and sequential data processing:

$$T_c = T_{scan} + T_{dsp}$$

Here we come to the conclusion that T_c depends on T_{dsp} and *time sheduling* of nodes in the SPS.

Varying S_h versus t_p

For a particular DSP node the relation between node execution time t_{pN} and node area S_{hN} exists as presented in Fig. 2 [7].

For each node a set of implementations with different t_{pN} and S_{hN} is available. The set of implementations lies around the hyperbola having the number of operations

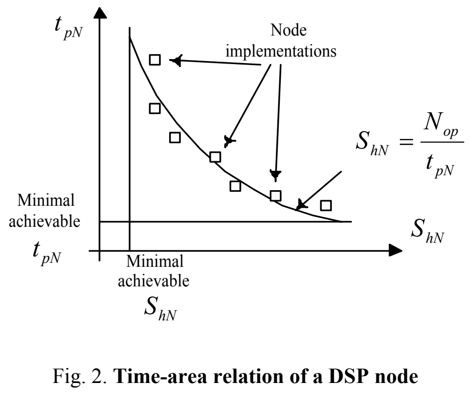

Fig. 2. **Time-area relation of a DSP node**

N_{op} (products and additions) in its numerator.

In the case of minimal S_{hN} the operations are being performed in sequence within extended t_{pN}.

Alternatively, for shorter t_{pN} the node must be mapped to parallel architecture which obviously has larger

S_{hN}. The extreme achievable limits of S_{hN} and t_{pN} exist. Stepping over the extreme achievable t_{pN} the possible solution would be to use two identical systems working in parallel with mutual time shift $\frac{t_{pN}}{2}$ and producing the results every $\frac{t_{pN}}{2}$ period, as it will be shown in Fig.6.

As the rule, software systems have sequential execution and reduced S_h meanwhile FPGA implementations are used for parallel architectures with short t_p and extended S_h.

The way of varying S_{hN} and t_{pN} within the node and node sheduling is to use the four classical architectures:

1) "*Single instruction, single data"* (SISD) which represents sequential Neuman architecture.

2) "*Multiple instruction, single data"* (MISD) which represents data pipeline.

3) "*Single-instruction, multiple data"* (SIMD) which represents several parallel data paths processed by single control path.

4) "*Multiple instruction, multiple data"* (MIMD) which represents pipelined data processing in several data paths.

The conclusion is to choose the node algorithm with minimal N_{op} and to make it sufficiently parallel to satisfy S_{cN} and T_{cN}.

Analysis and implementation of TDE nodes

Further we make analysis of the nodes, their algorithms, N_{op}, and reasonability to map them to a particular resource.

The functioning of SEAS is described by the following parameters:

a) time events in SEAS (e.g. window functions)

b) scan cycle duration T_{scan},

c*)* ***x*** and ***y*** lengths n,

d) ***x*** and ***y*** sampling rates,

e*) TVGC* sampling rate and amplitude precision,

f***) y*** sampling precision (number of ADC bits).

Basically, the alternatives of SEAS implementation are related mainly with *control and timing* node. Other SEAS nodes are assumed to be defined by the application requirements and unchangeable. Note that several functions should being done simultaneously – x emission, y input, *TVGC* execution, clock and timing. For applications where control of these functions is performed with frequency of several megahertzs the implementation can be made by software in DSP or controller. In fast SEASs where timing-control events and sampling rates reach tens of megahertzs the hardware implementation on FPGA is more appropriate.

Cross-correlation function:

a) Direct CCF [8]:

$$R_{xy}(k) = \frac{1}{M}\sum_{i=0}^{M-1} x(i+k)\cdot y(i),\quad k=0..N-1$$

The algorithm has a simple control structure; however, its N_{op} increases by n^2 where n is the number of samples in ***x*** and ***y*** arrays. DSPs can be used for implementation of this algorithm but if T_c is not fulfilled the faster CCF algorithms should be chosen. Conventional DSP FIR filter software can be used to compute CCF. FPGA is helpful due to the opportunity to implement parallel architecture.

b) Fast CCF by FFT [8].

$$R_{xy} = FFT^{-1}\left[FFT(x)\cdot FFT^{*}(y)\right]$$

where – 1 means inverse FFT and * means complex conjugate array of FFT.

This algorithm has smaller N_{op} for *N>128* but complicated control path and significantly increased hardware resources. Parallel hardware implementation is unreasonable because of more effective implementation of the previous algorithm. The reason to use this algorithm is due to reducing t_p of CCF in DSPs.

c) Often the above two methods are straigtforward and redundant. The signal certain properties aren't appretiated with the aim to reduce N_{op}. An example of reduced N_{op} is computing the CCF of binary phase-modulated, e.g. Barker coded sequences [3]. This algorithm assumes reference signal x to be the composition of a periodic symbol $c(k)$ with k samples and Barker l-length code $b(l)$. The CCF can be computed by using double convolution:

$$R_{xy} = y(t) * c(t) * b(t)$$

Supposing y length to be $n >> l,k$ the calculation period becomes significantly less since:

$N_{op} = n\cdot(l+k)$ instead of $N_{op} = n\cdot l\cdot k$

Extraction the envelope $E_{xy}(k)$ is described by the following formulas:

$$E_{xy}(k) = \sqrt{R_{xy}(k)^2 + \widetilde{R}_{xy}(k)^2},\quad k=0..n-1,$$

where $\widetilde{R}_{xy}(k)$ is Hilbert transform of $R_{xy}(k)$:

$$\widetilde{R}_{xy}(k) = \sum_{m=o}^{M-1}\widetilde{R}_{xy}(k-m)\cdot h(m)$$

The coefficients *h(m)* are chosen to be of the Hilbert transformer [8]. In direct case HT is calculated by using FIR filter approach. Like all FIR filters it can be effectively implemented in hardware. The software implementation also can be reasonable. For example, when an approximate maximum of CCF have been found, it is sufficient to calculate several HT values within its neighbourhood to find the fine maximum.

Another way to compute HT is using FFT:

$$HT(R_{xy}) = FFT^{-1}\left[i\cdot(\mathrm{Re}(RFT) - \mathrm{Im}(RFT)\right],$$

where

$$RFT = FFT(x) \cdot FFT^{*}(y)$$

Here HT of entire CCF array is found. This algorithm is applicable when multiple maxima are searched and CCF is being computed by fast algorithm, therefore, intermediate results from the CCF algorithm can be used. Due to the reason of algorithm complexity software implementation should be used in the latter case.

For the purpose to find the accurate maximum peak between samples also various ***interpolation*** techinques are used. They can be DSP interpolation approaches e.g. linear and optimal filtering or polynomial interpolation [5].

The other nodes are the "threshold" from the processed digital signals to the estimated time delays. ***ArgMax detection*** algorithm can vary for different applications. Finding absolute maximum of the array is trivial while in the case of multiple maxima the more sophisticated algorithms may be applied [3]. Actually, the *Argmax* is much application specific, therefore, all possible cases aren't analysed in advance. ***Zero-cross detection*** and ***averaging*** also are simple subroutines with non-intensive calculations. Generally, the latter three algorithms and polynomial interpolation quite easy realizable in software and parallel their implementation in hardware would be complicated and not effective.

Interface controller is specified by the communication speed and data protocol. Usually DSPs and controllers provide on-chip serial and parallel interfaces. Alternative hardware implementation on FPGA also is possible. Note, that data stream is not intensive as delay vectors d may contain from several to several tens of data values.

TDE architectures for diverse time constraints

According the above reasoning several architectures with different T_c are suggested. We start at moderate values of T_c.

1) Single channel TDE system:

a) Unrestricted t_p, moderate T_{scan}. The system architecture can be minimized to a single DSP with very

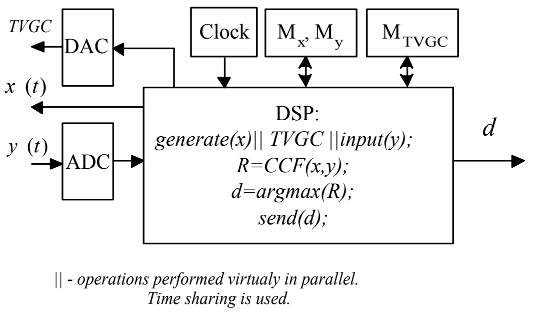

Fig.3. **TDE architecture with moderate T_{scan} and unrestricted t_p.**

few of glue logics and necessary external devices (SISD architecture).

b) Let us assume an example with more stringent T_{scan} and timing event constraints. DSP is not able to perform scan control. SEAS is implemented in hardware. Processing subroutines are left for the DSP.

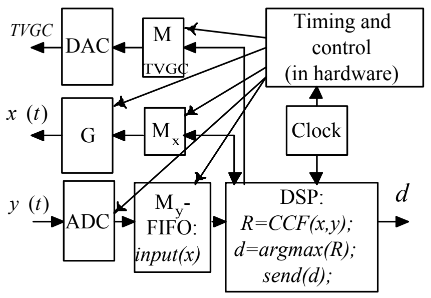
TVGC
DAC
M TVGC
Timing and control (in hardware)
x (t)
G
Mx
Clock
y (t)
ADC
My-FIFO: input(x)
DSP: R=CCF(x,y); d=argmax(R); send(d);
d

Fig.4. **Hardware accelerated TDE architecture for increased time constraints.**

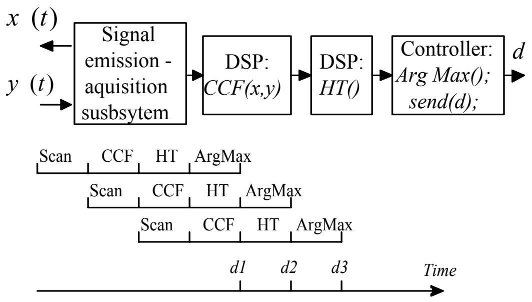

Fig.5. **Pipelined TDE architecture.**

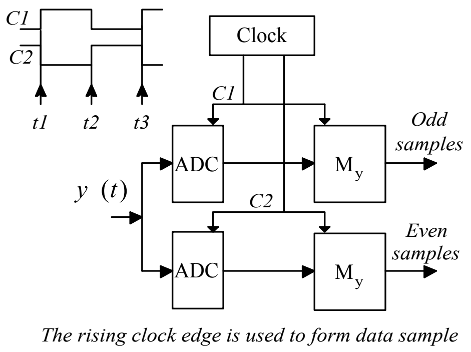

Fig.6. **Signal acquisition over extreme time constraints.**

c) architecture with the pipelined processing nodes (MISD). In this case d appearance period is equal to the longest execution time on a particular processor.

d) When extreme achievable t_p is exceeded the single system is not able to handle the required data throughput. Then two or more parallel systems should be working with a shift in time. The fast module for signal input consisting of 2 slower input blocks is shown in Fig.6.

2) *Multiple channel TDE system:*

a) One solution is to compose a multichannel system from single channel TDE architectures. All architecture cases described for a single channel TDE can applied in this way for a multichannel application.

b) If time is moderately constrained, then the time-multiplexed architecture with single DSP can be applied.

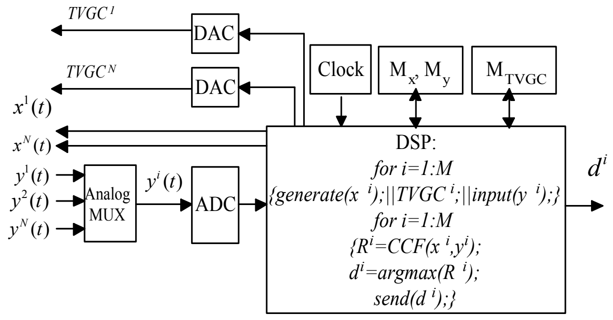

Fig.7. **Multichannel-multiplexed TDE architecture**.

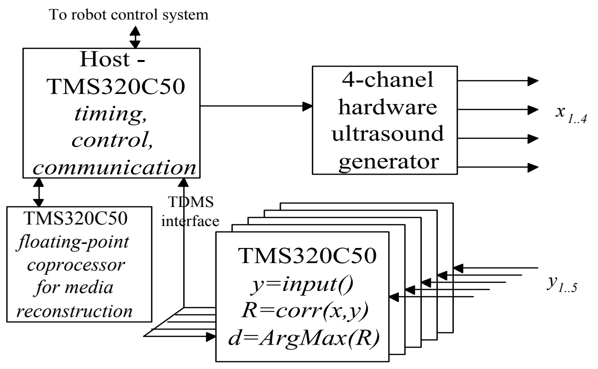
To robot control system
Host - TMS320C50 timing, control, communication
4-chanel hardware ultrasound generator
x 1..4
TDMS interface
TMS320C50 floating-point coprocessor for media reconstruction
TMS320C50 y=input() R=corr(x,y) d=ArgMax(R)
y 1..5

Fig.8 **Architecture of ultrasonic robot vision system**.

Example of architecture implementation

There is the implementation example of ultrasonic robot vision system that was described with more details in [6].

System is based on three different resource types – hardware as the ultrasound generator, fixed point DSP TMS320C50 for signal input, CCF processing, distance extraction; also TMS320C50 for overall system control and floating point coprocessor TMS320C31 – as GPP for surrounding media reconstruction. Although the two latter DSPs perform non-DSP functions they were selected due to convenient fast interfacing with no additional communication hardware and unified software development tools.

Conclusions and further research

The analysis of general TDE task was presented. Two independent subsystems in TDE data and control flow graph can be distinguished - SEAS and SPS. SPS nodes also can be differentiated into signal processing and delay extraction ones. For SEAS and digital signal processing both hardware and software implementations are reasonable depending on application specific time constraints. Nodes such as *ArgMax*, *Zero-cross* are calculation non intensive to compare with DSP algorithms CCF and HT, therefore, software approach is considered.

Further research could be directed towards creating the automated high level synthesis tool for implementation TDE tasks in large scale FPGAs.

Acknowledgements

This project was carried out as a part of INCO-COPERNICUS project No. ERB IC15-CT96-0726.

References

1. **Kažys R.** Smart systems for robot vision. – MMAR'98, pp. 827-832, 1998.
2. **Kažys R.** Delay time estimation using the Hilbert transform, Matavimai, KUT, Vol. 3, pp. 101-106, 1996.
3. **Audenaert K. et al.** Accurate Ranging of Multiple Objects using Ultrasonic Sensors. IEEE Robotics and Automation, 1992, pp.1733-1738.
4. Spread Spectrum Ultrasonic Evaluation, Final Report, EECE, Iowa University, 1994.
5. **Lai X., Torp H.** Interpolation Methods for Time-Delay Estimation Using Cross-Correlation Method for Blood Velocity Measurement. IEEE transactions on Ultrasound, Ferroelectric and Frequency Control, Vol.46,1999.
6. **Kazanavičius E., Venteris R.** Implementation of DSP algorithms for Ultrasonic Measurement Applications. Ultragarsas Nr.2(32), KTU, Kaunas,1999.
7. **Kalavade A.** System-level codesign of mixed hardware-software systems, Ph.D. Dissertation, University of California, Berkeley, Sept.,1995.
8. **Oppenheim A. V., Shaffer R. W.** Discrete-time signal processing, Prentice Hall, 1989.

E. Kazanavičius, R. Venteris

Ultragarso signalo lėkio trukmės įvertinimo uždavinių architektūros

Reziumė

Ultragarso signalo lėkio trukmės (USLT) matavimas yra pagrindinis daugelio taikomųjų atstumo matavimo, ultragarsinės regos, tomografijos, skysčių ir dujų srauto greičio matavimo, nedestruktyvios kontrolės uždavinių algoritmas. Šiems uždaviniams spręsti efektyviai naudojami skaitmeninio signalų apdorojimo algoritmai ir atitinkamos kompiuterinės architektūros. Nors įvairiems USLT matavimo uždaviniams taikomi tie patys apdorojimo metodai, tačiau kiekvienam taikomajam uždaviniui keliami individualūs reikalavimai ir apribojimai. Todėl turi būti parenkamas individualus algoritmas ir architektūra. Pagrindiniai apribojimai architektūrai yra maksimali leistina algoritmo vykdymo trukmė ir architektūros plotas. Straipsnyje analizuojami USLT matavimo klasės uždaviniai, jų pagrindiniai paprogramiai bei galima jų realizacija programinėmis ir aparatinėmis priemonėmis. Pateikiami keli architektūrų pavyzdžiai, esant skirtingiems vykdymo laiko apribojimams.

Pateikta spaudai 2000 04 17