Hardware synthesis Methodology of N-FINDR algorithm

Anabella Medina Machín, Gustavo Marrero Callicó, Sebastián López Suárez
Integrated System Design Division, IUMA
Las Palmas de Gran Canaria, Spain
allebana@gmail.com, gustavo@iuma.ulpgc.es, seblopez@iuma.ulpgc.es

***Abstract*—This paper presents a hardware synthesis methodology for the implementation of the N-FINDR algorithm using MATLAB. In order to implement hardware prototypes following the time to market philosophy, we have developed a hardware synthesis methodology for high-level languages. Using software like Embedded C and Fixed Point MATLAB Toolbox and Catapult C we have achieved good results in the synthesized hardware for FPGAs with reduced execution results. In this article we used the fixed point arithmetic as an alternative to floating point which is not suitable for electronic devices such as FPGAs.**

Keywords: hyperespectral, endmember, methodology, N-FINDR.

During the last years, several algorithms for endmember extraction have been published in scientific literature. Despite the different nature of these algorithms, they all demand a huge computational effort in order to extract the endmembers of a hyperspectral image. Combined with the high dimensionality of hyperspectral data, they cause serious complications in the use of these algorithms in application domains under real time constraints.

This paper proposes a fast implementation methodology of one of the most popular endmember extraction algorithm, the N-FINDR. This methodology is based in the synthesis of the N-FINDR MATLAB algorithm.

To achieve this, software tools have been used to manage embedded C code generation from MATLAB's original code. Having obtained the Embedded C code, we proceed to perform the hardware synthesization from high-level language. Before synthesizing the algorithm, we have to use fixed point arithmetic instead of the inefficient floating point.

The Embedded C program generated is able to reproduce the results of its original, while exhibiting a lower computational complexity in comparison of the Matlab algorithm. Using the proposed solution of fixed point for the embedded C code obtain good results from frequency and latency in the hardware synthesized by the example of Altera Stratix III FPGA or Xilinx-SPARTAN6.

The rest of this paper is organized as follows. Section I describes the N-FINDR algorithm, while Section II exposes the methodology developed for synthetized MATLAB code algorithm. Section III presents the most significant results obtained and, finally, Section IV outlines the conclusions extracted from this work.

I. THE N-FINDR ALGORITHM

The N-finder algorithm (N-FINDR) was developed by M. E. Winter [1]. It finds a simplex of the maximum volume with a given number of vertices, p. It is based on an assumption that for a given p-vertex simplex, the simplex that yields the largest volume will be the one whose p vertices are most likely specified by the purest pixels. The vertices of an N-FINDR found simplex are the desired set of endmembers.

Unfortunately, there are several disadvantages of implementing the N-FINDR. One is that there is no provided criterion to determine how many endmembers for the N FINDR to generate. Another is that the N-FINDR uses randomly generated vectors as initial endmembers, which is not an effective way to initialize the algorithm. It generally takes a long time to find a desired set of endmembers. Most importantly, due to the nature of random initial endmembers, the N-FINDR generally produces different sets of final endmembers at separate runs.

In this section, we summarize the steps to implement the N-FINDR according to [2].

1. Preprocessing:
   a) Let be the number of endmembers required for the N-FINDR to generate.
   b) Apply a DR transform such as MNF to reduce the data dimensionality from L to p where L is the total number of spectral bands.
2. Exhaustive search:

For an arbitrary set of data sample vectors e1, e2, ...,ep form a p-vertex simplex specified by S (e1, e2, ..., ep) and define its volume $V(e_1, e_2, \ldots, e_p)$ by

$$V(e_1, \ldots, e_p) = \frac{\det \begin{bmatrix} 1 & 1 & \ldots 1 \\ e_1 & e_2 & \ldots e_p \end{bmatrix}}{(p-1)!}$$

Find a set of data sample vectors in the data, denoted by $\{e^*_1, e^*_2, \ldots, e^*_p\}$ that form a p-vertex simplex to yield the maximum value of:

$$\{e^*_1, e^*_2, \ldots, e^*_p\} = \arg \{ \max_{\{e\}} V(e_1, e_2, \ldots, e_p)\}$$

The set of $\{e^*_1, e^*_2, \ldots, e^*_p\}$ is the desired set of endmembers needed to be found. To complete the above exhaustive

search in this step require $\binom{N}{p} = (N!)/(p!\,(N-p)!)$ operations.

II. MATLAB CODE METHODOLOGY SYNTHESIZATION

Programming in high level languages allows quick development of algorithms. New programming tools like Embedded and Catapult C allow fast hardware synthesis from MATLAB code. For the best results of the synthesis it is necessary to study the algorithm to simplify and adapt it to the requirements of Embedded C. Furthermore, we have tested it for the use of fixed point arithmetic and integer arithmetic to improve the performance of the NFIND-R that runs on devices without Floating Point Units. The proposed methodology for the algorithm aims to simplify the process of hardware synthesizing other algorithms developed in MATLAB. The presented method selects the use of integer arithmetic, fixed point or floating depending on the algorithm, the restrictions imposed by Embedded MATLAB and taking into account the possible generation of overflow in math calculations.

III. RESULTS

In this section, the performances of the different versions of N-FINDR algorithms are compared.

Artificial hyperspectral images represent an excellent test bench for the purpose of comparing the diferent algorithms. In particular, the hyperspectral images used in this work were generated by the *demo* software tool available at [3].

In order to evaluate the accuracy of the algorithms, the Spectral Information Divergence (SID) measures the difference between the extracted endmember $\hat{m}_i$ and its correspondent real endmember signature m_i is calculated as follows:

$$SID_{m_i,\hat{m}_i} \equiv D(m_i|\hat{m}_i) + D(\hat{m}_i|m_i)$$

$$D(m_i|\hat{m}_i) \equiv \sum_{j=1}^{L} p_j \log\left(\frac{p_j}{q_j}\right)$$

$$p_j = \frac{m_{ij}}{\sum_{k=1}^{L} m_{ik}} \text{ y } q_j = \frac{\hat{m}_{ij}}{\sum_{k=1}^{L} \hat{m}_{ik}}.$$

TABLE I
N-FINDR SID ERROR

	5 Endmembers	10 Endmembers	15 Endmembers
N-FINDR original	0,00241	0,0545	0,0153
N-FINDR Fixed Point	0,00261	0,0545	0,0153
N-FINDR C++ fixed	0,00227	0,0169	0,018
N-FINDR integer arithmetic	0,00479	0,0545	0,0153

TABLE II
N-FINDR FRECUENCY

	Freq. MHz
N-FINDR original	37,164
N-FINDR Fixed Point	48,783
N-FINDR C++ fixed	33,568
N-FINDR integer arithmetic	32,624

TABLE III
N-FINDR LATENCY CYCLES

	Latency cycles
N-FINDR original	3.932.839
N-FINDR Fixed Point	2.492.406
N-FINDR C++ fixed	1.311.286
N-FINDR integer arithmetic	1611

IV. CONCLUSION

As seen from both tables, the SID error results are very similar between de different implementations. On the other hand, the frequently results are better for the N-FINDR Fixed Point implementation. However, that information could not be good at all, because it can refers to the frequently of internal operations that not necessary produce a data out. So, we decide that latency cycles could be better to perform the comparative algorithm. So, the N-FINDR integer arithmetic present good result for the latency cycles data.

V. REFERENCES

[1] Winter, M. E. "N-finder: An algorithm for fast autonomous spectral endmember determination in hyperspectral data." (Image Spectrometry V, Proc. SPIE) 3753, no. 266-277 (1999).

[2] Chein-I Chang, Fellow, IEEE, Chao-Cheng Wu, Member, IEEE, and Ching-Tsorng Tsai. "Random N-Finder (N-FINDR) Endmember Extraction Algorithms for Hyperspectral Imagery." *IEEE TRANSACTIONS ON IMAGE PROCESSING* 20, no. 3 (2011).

[3] *Bioucas.* http://www.lx.it.pt/~bioucas/code.htm. .