An FPGA Implementation of Incremental Clustering for Radar Pulse Deinterleaving

Scott Bailie
bailie@ll.mit.edu
MIT Lincoln Laboratory, Lexington, MA

Miriam Leeser
mel@coe.neu.edu
Northeastern University, Boston, MA

Background

A military aircraft in a hostile environment may need to use radar jamming in order to avoid being detected or engaged by the enemy. Effective jamming can require knowledge of the number and type of enemy radars; however, the radar receiver on the aircraft will observe a single stream of pulses from all radar emitters combined. It is advantageous to separate this collection of pulses into individual streams each corresponding to a particular emitter in the environment; this process is known as pulse deinterleaving. Pulse deinterleaving is critical for effective electronic warfare (EW) signal processing such as electronic attack (EA) and electronic protection (EP) because it not only aids in the identification of enemy radars but also permits the intelligent allocation of processing resources.

Receiver front-ends perform numerous measurements on each input pulse including, but not limited to, time of arrival (TOA), pulse width (PW), and frequency (RF). The collection of these measurements forms a pulse descriptor word (PDW) describing the characteristics of the pulse. Using clustering, a pattern recognition and data mining technique that attempts to separate related data into groups called clusters, it is possible to deinterleave a single stream of radar PDWs and identify the number and characteristics of the radars in the operating environment.

The FPGA implementation of a novel incremental clustering algorithm for pulse deinterleaving is presented, including an architecture description and the resulting performance and area metrics. The FPGA implementation is 40 times faster than software and is capable of keeping up with input data in real time.

Algorithm Overview

Unlike non-incremental algorithms that require all data to be present prior to processing, incremental approaches cluster data in a streaming mode as it is received. Being a one-pass approach, incremental clustering is relatively fast, requires little long-term storage, and is suitable for data whose characteristics may evolve over time.

The algorithm developed for this implementation is called Incremental Clustering for Evolving Data, or ICED. ICED performs clustering using the PW and RF parameters which results in each input PDW being defined by a single point in the 2-dimensional PW / RF space. This is demonstrated in Figure 1 for three radar emitters with unique PW and RF characteristics. As inputs are received by the algorithm, the distance from the input to each existing cluster is calculated. If the distance to the nearest cluster is less than a specified threshold, then the input is assigned to that cluster and the cluster's coordinates are updated as a weighted average of the current cluster coordinates and the coordinates of the input. If the distance to the nearest cluster exceeds the specified threshold, a new cluster, with the coordinates of the input, is created.

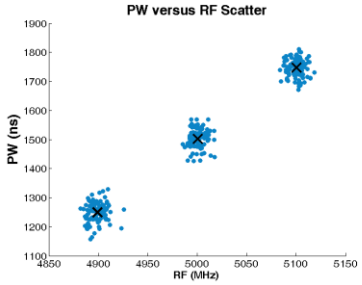

Figure 1 - Clustering of PW & RF Parameters

Every time an input is assigned to a cluster the weight of that cluster, representing its strength, is incremented. Since clusters may cease to be assigned new data if the characteristics of a particular radar changes or if that radar is no longer in range, ICED contains a fade mechanism to phase out stale clusters. As previously mentioned, cluster weights are incremented as new data is assigned to that cluster. Keeping track of time using the TOA parameter the fade operation periodically decreases cluster weights providing a higher dependence on new data and removing stale clusters.

Pulses from radars emitting at different pulse repetition intervals (PRIs) will eventually result in overlapping pulses at the receiver, as shown in Figure 2. The measured PW and RF will likely result in the creation of a cluster which does not correspond to an actual emitter in the environment. Based on the expected infrequent occurrence of these overlaps, the clusters formed by these cases will have low weights and the fade mechanism will act as a filter by deleting them quickly.

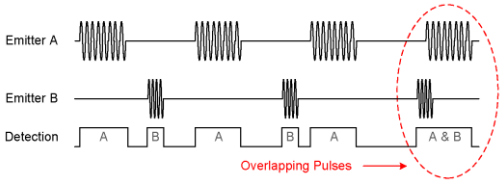

Figure 2 - Example of Overlapping Pulses

* This work is sponsored by the Department of the Air Force under Air Force Contract FA8721-05-C-0002. The opinions, interpretations, conclusions, and recommendations are those of the authors and are not necessarily endorsed by the United States Government.

Implementation

The ICED algorithm is implemented on an Innovative Integration X5-400M XMC module containing a Xilinx Virtex-5 SX95T FPGA. Emitter scenarios are developed in MATLAB and PDW inputs are created in software using a SIMULINK model of a pulse measurement module. The PW and RF inputs are represented by 16-bit unsigned integers while the TOA parameter is represented by a 32-bit unsigned integer. Cluster parameters such as coordinates and weight are available in real-time via software accessible FPGA registers.

A top-level block diagram is shown in Figure 3. Inputs are first buffered in a FIFO permitting asynchronous operation from the pulse measurement module. The first step is to normalize the PW and RF values ensuring an equal weighting in each dimension. Simultaneously, the **Fade** module calculates the cluster weight decrement based on the elapsed time since the previous input. Next, the normalized PW and RF parameters and the fade value are passed to the **Clusters** module. Here, an array of ***n*** cluster center modules, $\mathbf{C_0 - C_{n-1}}$, simultaneously calculate their respective distances to the input. The resulting distances as well as current cluster weights are supplied to the **Min_d** and **Min_w** modules. The **Min_d** and **Min_w** modules are responsible for finding the clusters with the minimum distance to the input and minimum weight, respectively.

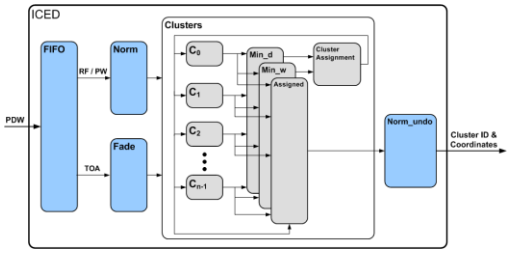

Figure 3 - Top-Level Block Diagram

The job of the **Cluster Assignment** module is to pick the winning cluster based on the minimum distance and weight calculation. The cluster with the minimum distance is chosen if its distance satisfies the specified threshold. Otherwise, the lightest cluster is selected as a replacement for creating a new cluster. When resources are available, the choice will result in the selection of an unused cluster (weight = 0) and when all resources are consumed it will result in the selection of the "weakest" cluster (lightest weight), possibly representing noise or an overlap condition. In a hardware implementation with fixed resources, such a decision is necessary.

The assignment decision is fed back to the cluster center modules allowing the wining cluster to update its coordinates. To decrease overall latency, every cluster center assumes it will be the winner and begins the coordinate update calculation in parallel with the distance calculation and determination of the wining cluster, this is shown in Figure 4.

Once the winner is chosen, the corresponding cluster center updates its coordinates with the calculated value while all other centers remain unchanged. Finally, the **Assigned** module selects the coordinates of the winning cluster, which are then output after being converted to the native input units.

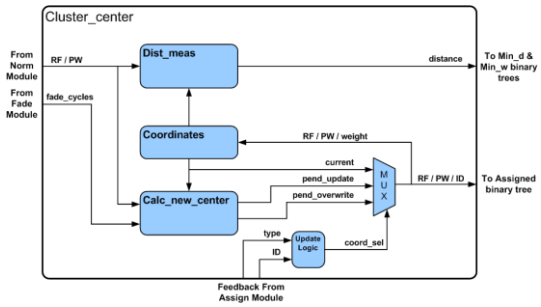

Figure 4 - Cluster Center Module

Results

The area consumption based on the number of implemented clusters is shown in Figure 5. The current implementation is for 16 cluster centers and consumes approximately 70% of the SX95T FPGA.

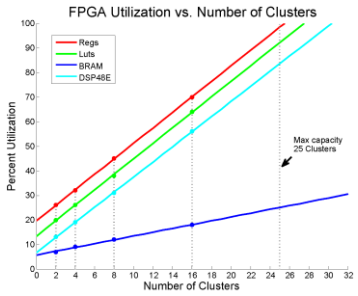

Figure 5 - FPGA Utilization vs. Number of Clusters

The resulting processing latency is 84 clock cycles and is shown in Figure 6. This corresponds to 420ns with the current clock frequency of 200MHz and results in a 39x speedup over a C implementation running on a 3.00 GHz Intel Xeon 5160.

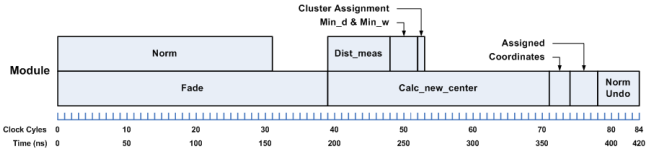

Figure 6 - Processing Latency for Major Modules

In an actual scenario, the deinterleaver must be able to keep up with the flow of input pulses, which is a function of the number of emitters and their pulse repetition frequency (PRF). Using a typical high PRF value of 200 KHz as an example, the current FPGA implementation could support nearly 12 such emitters, which would represent an extremely dense environment.

Conclusions

Effective EA and EP require pulse deinterleaving to enable proper identification of received radar waveforms and in turn intelligently allocate processing resources in real-time. A parallelized 16 cluster implementation of the ICED algorithm consumes 70% of a Xilinx Virtex-5 FPGA and results in a 39x speedup over software. This performance is more than adequate for anticipated environments and future work might target time sharing of resources between cluster modules, in turn decreasing parallelism and allowing other system modules to be co-located on the same FPGA.