3. HARDWARE ARCHITECTURE

The architecture of the Ni1000 Recognition Accelerator consists of two main parts: a dedicated *classifier* engine and a general-purpose 16-bit *microcontroller*. The classifier implements the model described in the previous chapter, while the microcontroller code directs the passage of data through the classifier, implements the learning algorithm(s) and interacts with software running on the host.

Figure 3-1 is a block diagram of the internal hardware architecture. The upper part of Figure 3-1 shows the classifier, the middle part shows the microcontroller, and the lower part shows the interface to the host.

In an application environment, the classifier receives data from the host system through the bus interface, processes it, and sends the classification results back through the bus interface to the host. The classifier exploits both array and pipeline parallelism to provide the host with up to 33,000 classifications per second. The parallel hardware of the Distance Calculation Units and their tight coupling to the Prototype Array (PA) are responsible for much of this processing power. The Prototype Array holds 1024 (prototypes) x 256 (dimensions) x 5 (bits per dimension), for a total of 1.3 million non-volatile Flash storage bits. It can also hold prototypes for multiple problems with varying dimensionality—up to 8192 prototypes with 32 dimensions. Problems that need additional prototype storage can be solved with systems using multiple Ni1000 Accelerators. It is also possible to trade lower input-vector dimensionality for higher input-feature bit resolution.

Each of the 512 parallel Distance Calculation Units calculates the city-block distance (see Chapter 2) by summing the differences between each component (dimension) of the input vector and that of a prototype vector stored in the Prototype Array. The DCUs are multiplexed twice in time to achieve a sustainable processing rate of 16.5 billion operations per second and a bandwidth of up to 2.5 x 1010 bits per second. Parallel, absolute-value subtractors sequentially process the dimensions.

The classifier's Math Unit (MU) calculates probability density functions and class results concurrently. It processes floating-point data and computes the exponential and other mathematical functions that appear in equations (1) and (3) of Chapter 2. The MU uses a six-stage pipeline with a resolution of 16-bits for floating-point computations (10-bit mantissa and 6-bit exponent). It places results in one of two static RAMs. This double-buffering scheme allows the Math Unit to continue processing a second vector without interrupting the classification pipeline. The Prototype Parameter RAMs (PPRAMs) hold parameters like λ, *k,* and *C*, described in Chapter 2.

The middle part of Figure 3-1 shows the microcontroller. It is a fully custom, 16-bit, Harvard-architecture microcontroller that supervises learning, performs chip maintenance tasks, and maintains communication with the host. It can also exchange interrupts with the host. The 4k x 16-bit PGFLASH Flash memory stores the microcontroller programs. All memory devices are memory-mapped to the microcontroller's address space, with the exception of the PGFLASH, which is the microcontroller's program Flash memory. Other facilities available to the microcontroller include 256 words of general-purpose static RAM (GRAM) and a free-running

32-bit timer. Although not shown in this diagram, the microcontroller has access to virtually all of the memories in the classifier, although classification must stop while the microcontroller accesses these memories. A detailed description of the structure of the classifier and microcontroller follows in the remainder of this chapter.

The bottom part of Figure 3-1 shows the interface to the host, which consists of input buffers (IRAM), an output buffer (ORAM), and sixteen I/O control registers. The external data bus can be either 32 or 64 bits wide and will perform single-clock burst transfers. The input stage buffers two full-sized vectors. The outputs can be either in IEEE standard 32-bit floating-point format or the internal 16-bit floating-point format. Both the host and the Accelerator's on-chip microcontroller can access the sixteen 16-bit I/O control registers. The registers contain various control parameters for the Accelerator and provide a general channel for communication between the microcontroller and the host.

3.1 The Classifier

The classifier consists of the pipeline shown in Figure 3-2. While data is loaded into the double buffer at the input of the pipeline or read from the output of the pipeline, the classifier can compare a previously loaded input vector against the prototype vectors in the prototype array.

The classifier consists of the following units:

- *Input RAM (IRAM)*—a double buffer consisting of two 256 x 5 memories. Each memory can store one input vector. The IRAM is part of the bus interface unit, which is described in Section 3.2.
- *Prototype Array (PA)*—a flash memory that holds the prototype vectors allocated during learning, i.e. the coordinates of the Radial Basis Function (RBF) centers.
- *Distance Calculation Units (DCUs)*—a 512-processor array that performs the distance calculations between an input vector and each prototype vector in the PA.
- *Prototype Parameter RAMs (PPRAMs)*—a memory that holds all of the data that defines an RBF except its prototype vector (which is stored in the PA). This data includes the RBF radius, number of vectors it recognized during exposure to the training set, etc. Unlike the PA, the PPRAM is not flash memory; it is static RAM. Typically, it is loaded during power-on initialization from off chip or from a reserved section of the prototype array (PA).
- *Math Unit (MU)*—a six-stage pipelined processor that implements the decay function for evaluating probability. It also applies the threshold function, to decide whether an input vector falls within an RBF.
- *Math Unit RAMs (MURAMs)*—a set of memories that receives the class IDs that are classified as similar to the input vector. It also holds the accumulated probability value for each class.
- *Output RAM (ORAM)*—a buffer that receives the classification results for a vector and optionally reformats the probability values from the internal 16-bit floating-point format into a format compatible with the standard IEEE 32-bit floating-point format. The ORAM is also part of the bus interface unit, which is described in Section 3.2.

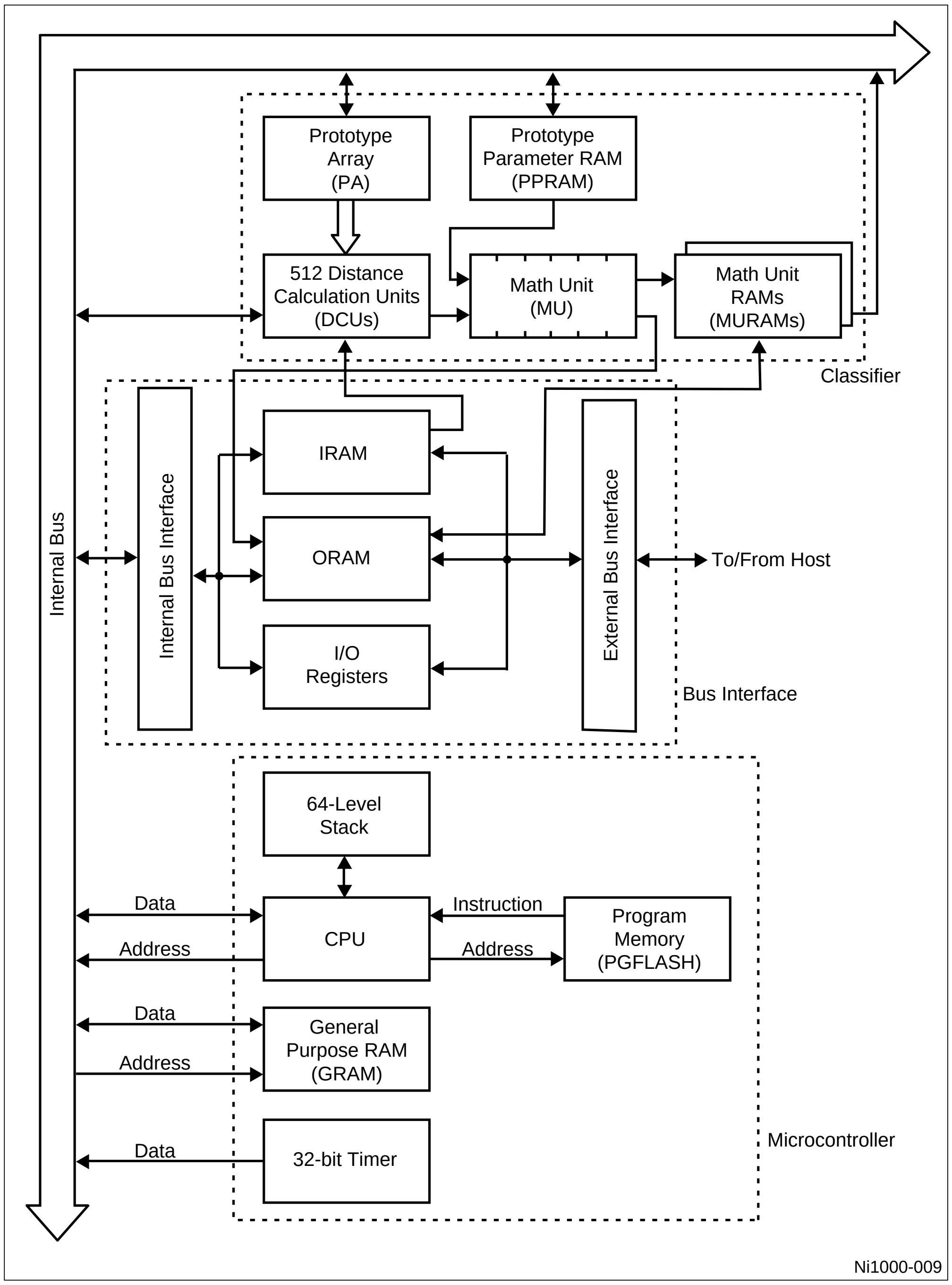

Figure 3-1. Internal-Architecture Block Diagram

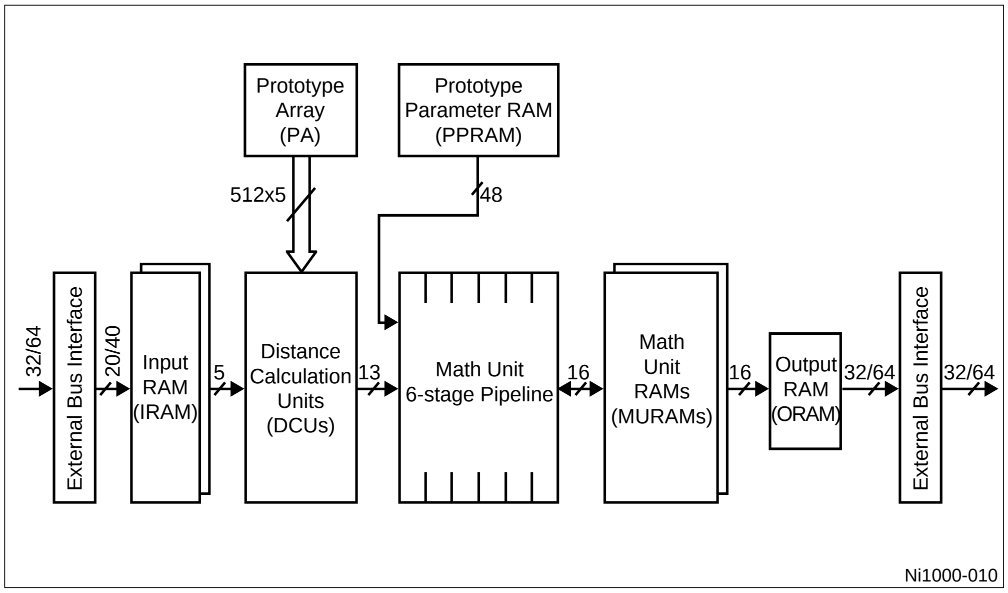

Figure 3-2. Classifier

3.1.1 Distance Calculation Units (DCUs)

Figure 3-3 shows an individual distance calculation unit (DCU) from the 512-unit array. The DCUs are statically associated with PA columns. Due to redundant array elements, only 500 DCUs are used for classification at any time. It computes the absolute value of the difference between an feature (dimension) of the input vector and the corresponding feature of a prototype. The DCU then accumulates that value into a running tally of the distance between the input vector and the prototype. Such distance is calculated between the input vector and each valid (i.e. not disabled) prototype.

The DCU accumulates a sum of the absolute differences, called *city-block distances,* between the input vector and each prototype vector in each dimension. The following equation expresses the city-block distance, *d,* between an input vector *U* with *i* dimensions and a prototype vector *P*.

$$d = | u_0 - p_0 | + | u_1 - p_1 | + ... | u_i - p_i |$$

The DCU has two accumulators and is used in a two-cycle mode, in which half of the prototypes in the PA are processed in one cycle, and the other half in the following cycle. When there are 500 prototypes or less, a one-cycle mode is used that does not require the second accumulator. When there are more than 500 prototypes, the two-cycle mode is used, with the other accumulator being used during the second cycle. Section 3.6 describes the timing of the DCUs and the classification pipeline.

At the end of a classification pass through the prototype array, the values in the accumulators represent the city-block distance between the input vector and each valid prototype vector stored in the PA. This is accessed by the MU pipeline for evaluating the probability and threshold functions.

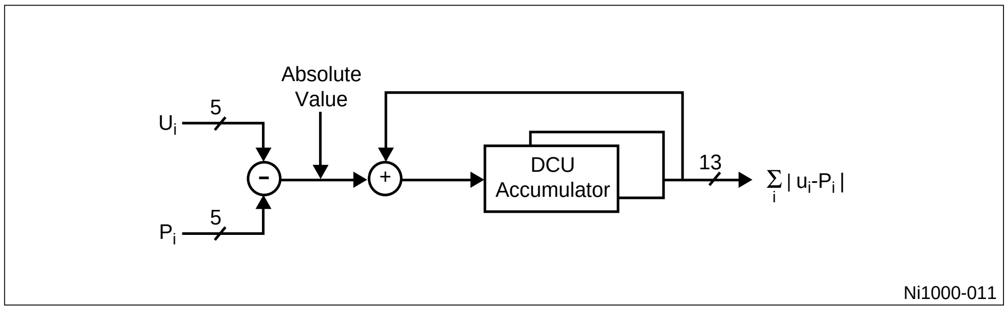

Figure 3-3. Distance Calculation Unit (DCU)

3.1.2 Prototype Array (PA)

The PA is a flash memory holding up to 256 five-bit features for each of up to 1024 prototype vectors (or 8192, for problems with 32 dimensions or less). Flash memory is a form of non-volatile electrically-erasable memory. See Section 3.2.4 for a description of writing to the PA.

Figure 3-4 shows a conceptual view of the PA flash memory being accessed by the DCUs. An input vector is presented as a stream of 5-bit integers. These are the individual features of the input vector. The 500 DCUs operate one feature of the input vector against the corresponding feature in up to 500 prototype vectors simultaneously. After the last prototype has been processed, the DCUs pass their accumulated city-block distances to the next stage of processing, the math unit (MU) pipeline.

PA flash memory can only be programmed by the microcontroller. Programming PA requires a 12V voltage applied to the Vpp pin.

Two address ranges in the microcontroller's address space are used to read the PA. An address in the range from B000h to B3FFh is used to specify one of the 1K prototype vectors to read. Note that, due to redundancy and remapping, a column address and prototype number may not match. Another range of 256 addresses from B800h to B8FFh specifies which features of the vector are to be read. Reading is a two-step process in which a first read
specifies either the vector or the feature, and a second read specifies the remaining quantity. Either the vector or the feature can be specified first. Valid data is returned on the second read. The upper six bits of the data are undefined. The lower ten bits are the value of one 5-bit feature in both true and complement form. Figure 3-5 shows the alignment of a 5-bit feature, p[0:4]. Complemented values are indicated by a bar over the bit name.

Figure 3-6 shows the architecture of the PA during programming. There are four registers used to control the PA during programming: the control and status registers, *CSA* and *CSB*, and the hardware mode setting registers, *AUX* and *MODE*. See Section 5.1.6 for details.

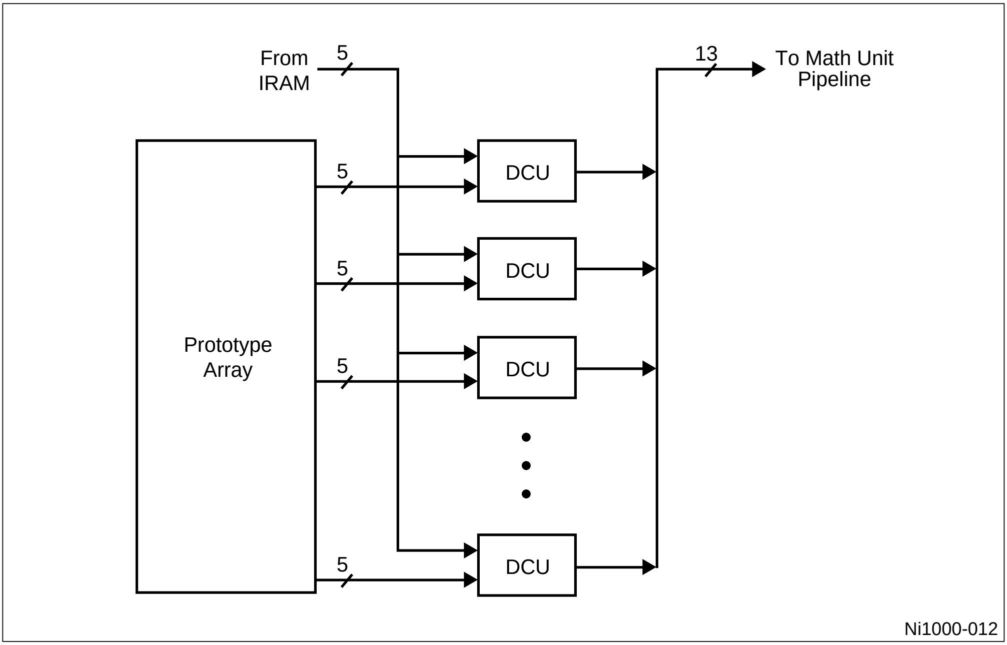

Figure 3-4. Conceptual View of Prototype Array (PA) and DCUs

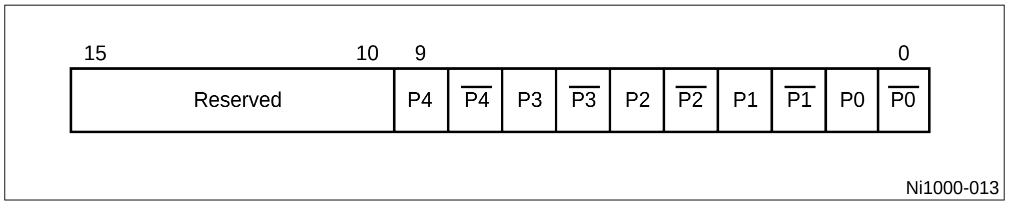

Figure 3-5. PA Data Format

3.1.3 Prototype Parameter RAMs (PPRAMs)

As each 13-bit city-block distance enters the MU, it is accompanied by 48 bits of parameters for its RBF, which come from the PPRAMs. These parameters include several fields, such as the RBF radius. The PPRAMs can only be written by the microcontroller, and appears as three 16-bit banks in the microcontroller's address space. Figure 3-7 shows the PPRAMs.

The addresses of the three banks, PPRAM1, PPRAM2, and PPRAM3, are given in Chapter 5. The fields of the 48-bit word passed to the MU pipeline, broken down by bank, are given in Figure 3-8.

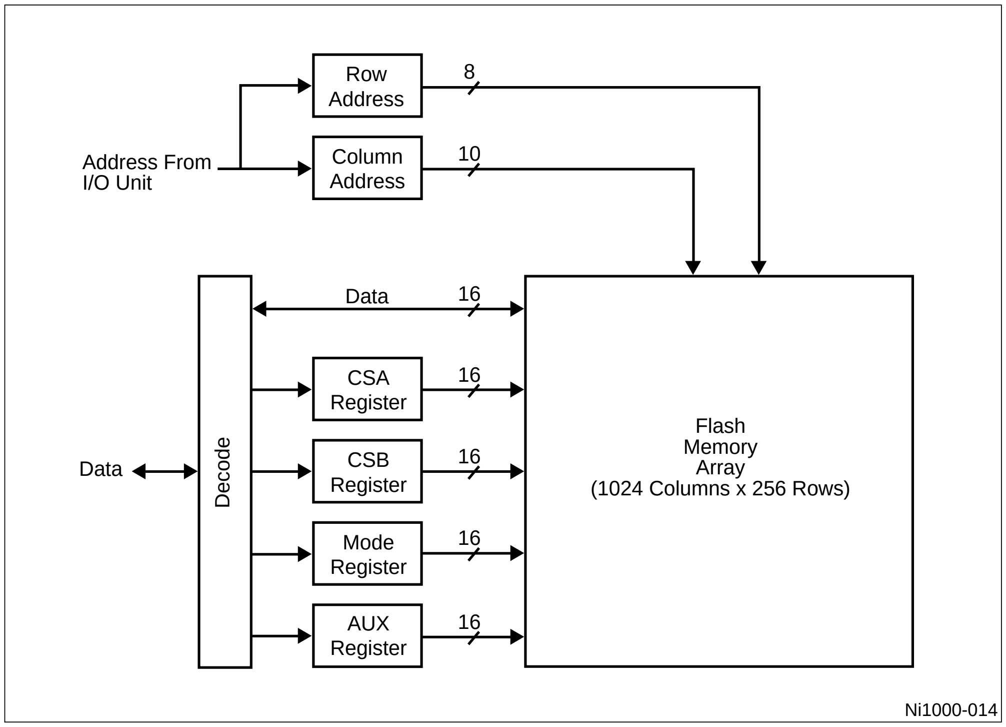

Figure 3-6. PA During Programming

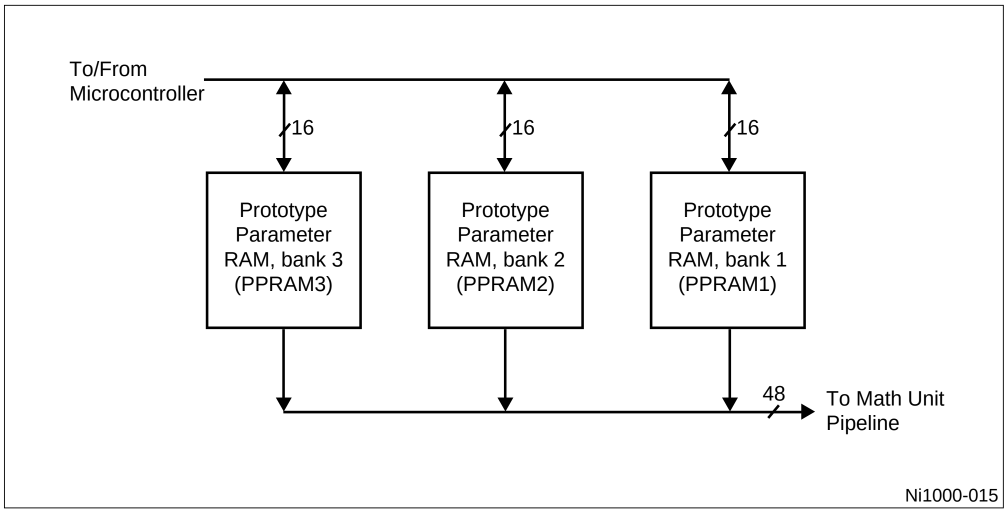

Figure 3-7. Prototype Parameter RAMs (PPRAMs)

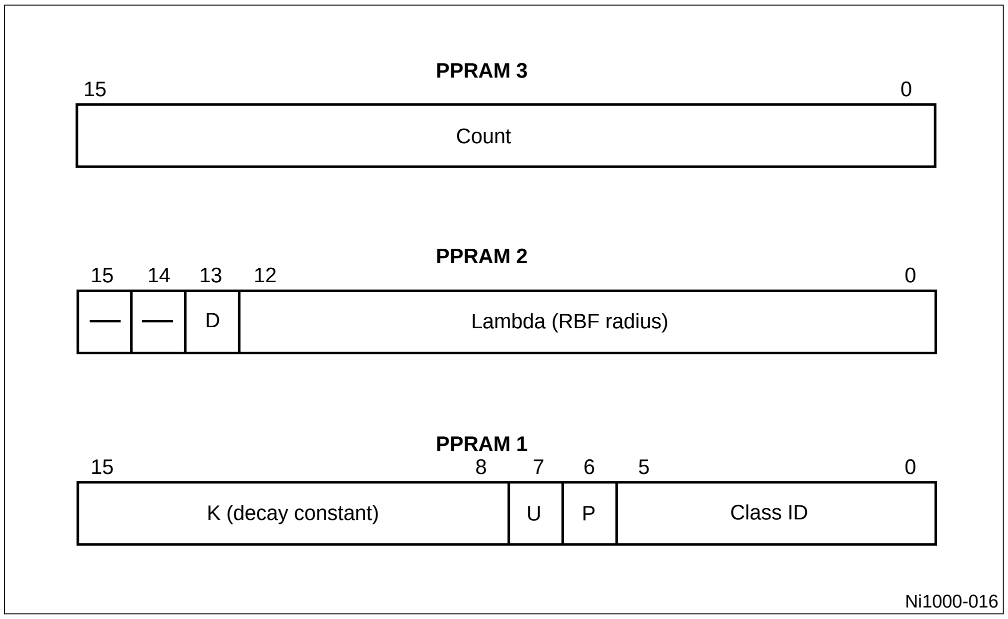

Figure 3-8. PPRAM Word Format

The fields of a PPRAM word are:

- *Count—C[0:15]*—the number of training vectors that fall within this RBF during learning; used as a factor during classification when calculating probability density.
- *Disable Flag—D*—set to disable this prototype.
- *Lambda—L[1:12]*—the RBF threshold distance.
- *Decay-Constant Mantissa—*K_m*[0:3]*—unsigned mantissa of the decay constant of the exponential function.
- *Decay-Constant Exponent—*K_e*[4:7]*—signed exponent of the decay constant of the exponential function.
- "*Used" Flag—U*—set when the PPRAM word is loaded with a valid prototype.
- *Probabilistic—P*—indicates that the RBF threshold distance for this prototype is the minimum allowed threshold distance. This bit is passed through to the class identification result to indicate that only probabilistic, not deterministic, classification is possible with this prototype.
- *Class—S[0:5]*—the class ID of the prototype.

All prototypes that have their *Used* flag set to 1 will be processed by the classifier. To avoid the possibility of processing spurious data, all locations in the PPRAM should be written whenever the chip is loaded with a new set of prototypes, and unused prototypes should have their *Used* flag cleared to 0.

3.1.4 Math Unit (MU)

The inputs to the first stage of the MU pipeline are the fields described above for the PPRAMs, accompanied by D, the 13-bit city-block distance calculated between the input vector and the prototype by the DCUs. The MU pipeline has two functions: it checks whether an input vector falls within an RBF and indicates when that happens, and it calculates the value of the RBF's probability density function at the point in feature space described by the input vector. Figure 3-9 shows the architecture of the MU pipeline. The Math Unit transfer function is described in Section 5.1.8.

The MU pipeline and the next functional block of the classifier, the math unit RAMs (MURAMs) are closely tied together. Several stages of the pipeline access data in the MURAMs. The ports shown in Figure 3-9 are connected to the MURAMs, as shown in Figure 3-10. The first stage of the MU pipeline writes to the *Flag MURAM*. The second stage writes to the *Class List MURAMs*. The fifth stage reads from the *Probability MURAMs*, and a result is accumulated which is written back to the *Probability MURAMs* following the sixth pipeline stage.

Recognition of whether an input vector falls within an RBF is indicated in the first stage of the MU pipeline by subtracting the RBF radius from the city-block distance and making a decision based on the sign of the result. This indication is used to update the 1-bit flag MURAM, which indicates whether that given output class is similar to the input vector. If the RBF is the first of its class to indicate recognition of the vector, a counter is incremented, the address it issues is used to allocate the next entry in the class-list MURAM and the RBF *P* bit is copied into the result for that class. The latter is a list of classes indicating recognition of the input vector.

Also in the first stage of the pipeline, the decay constant and the city-block distance are multiplied, and this floating-point product is split into three components for separate processing. This split is done for efficiency; each component is handled by a circuit that is easy to implement in hardware, then the output components are recombined.

After the first stage of the MU pipeline (See Figure 3-7), the four-bit exponent is processed by aligning its bits to the final result. The six most-significant bits (MSBs) of the mantissa are processed by a ROM lookup of the reciprocal of the exponential. The six least-significant bits are multiplied by a constant factor, the natural logarithm of 2 (i.e. ln 2). The computation of the exponential decay function occurs in the second and third stages, with recombination occurring in the third stage.

If an output class was indicated in the first stage as being recognized for the first time, its class ID is written to the class-list MURAM one cycle later. The address for this cycle comes from one of two class counters, and the data comes from the latch for the class ID in the second stage of the pipeline. Two counters are provided so that when the probability and class-list MURAMs swap (they are both double buffers), the counters can also swap, giving the circuits that control the ORAM a value for the number of classes in the class list.

The third stage of the pipeline produces the floating-point value of the probability density function for an RBF at the point in feature space described by the input vector.

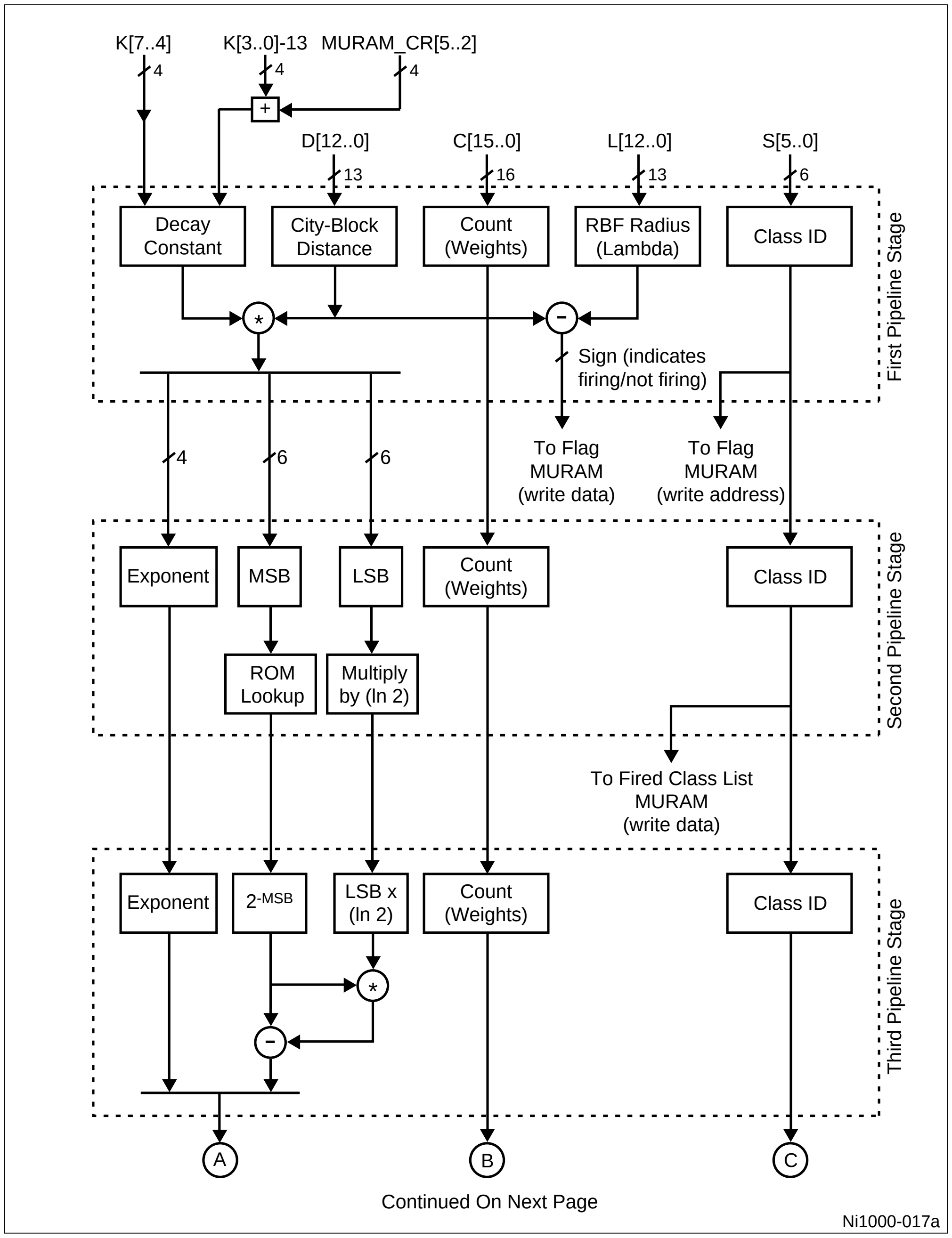

Figure 3-9. Math Unit (MU) Pipeline

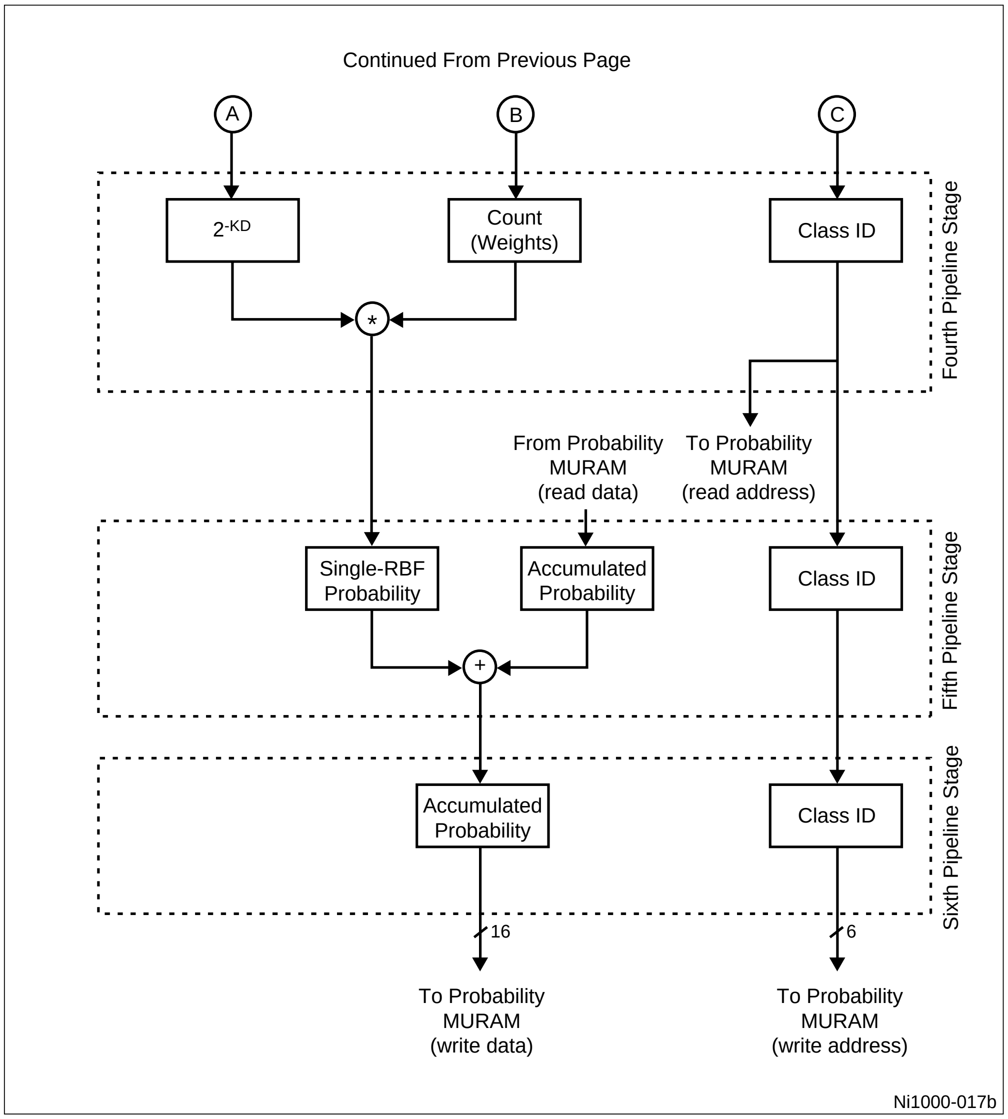

Figure 3-9a. Math Unit (MU) Pipeline (continued)

In the fourth stage of the pipeline, the probability value generated in the previous stage is scaled by the RBF's *Count*, which is typically a count of the number of times a vector in the training set fell within the radius of that RBF, although some algorithms may establish this factor using other means. The class ID from this stage is presented to the probability MURAM for a read cycle.

In the fifth stage of the pipeline, the scaled value for the probability density function of a single RBF is added to the accumulated value for all RBFs of the same class. This floating-point sum is written to the probability MURAM after the sixth stage of the pipeline, addressed by the sixth-stage class ID.

Once the last probability calculation has been performed for the input vector, the double-buffered MURAMs reverse roles, so that the classification results for the previous vector can be uploaded to the host through ORAM while the next vector is processed. Both the class list and probability density values are computed simultaneously, so either or both can be uploaded to the host without re-running the classification.

3.1.5 Math Unit RAMs (MURAMs)

The output of the MU pipeline is loaded into the MURAMs. The architecture of the MURAMs, shown in Figure 3-10, is intimately tied to the pipeline. Four of the six stages of the pipeline shown in Figure 3-9 either provide an MURAM address, read data from an MURAM, or write data to an MURAM. The MURAM memories include:

- *Flag MURAM*—a 1 x 64 memory used as a table of classes that have already recognized the input vector being presented. Each entry corresponds to one of the 64 possible classes.
- *Class List MURAMs*—an 8 x 64 x 2 double buffer holding a list of the class IDs of recognized classes. A new byte is allocated every time a new class is encountered. (Unlike the other MURAM memories, the memories in this buffer are not indexed by class ID; they are addressed by counters, so they grow up from address zero.)
- *Probability MURAMs*—a 16 x 64 x 2 double-buffer that accumulates the probability value of the input vector for each class. As with the flag MURAM, each MURAM address corresponds to one of the 64 classes.

The MU pipeline sends its data to the set of class-list and probability MURAMs currently waiting to receive input (while the other set is available to unload data into the ORAM, discussed later). One set of MURAMs may be written with the data for the input vector being presented, while the other set passes data, if available, for the previous input vector to the ORAM or the microcontroller. After the input vectors have been processed, the MURAMs change roles.

The flag MURAM is not accessible to the ORAM, so it is re-used every cycle. It is indexed by the class ID. When a class is recognized, the bit addressed by the class ID is set. If the bit previously was clear, that indicates the class had not yet been recognized during the processing of the input vector. This causes the class counter to be incremented and allocates a word in the class-list MURAM. The counter keeps a running tally of the number of classes, which is used to address the class-list MURAM when a new word is allocated.

The class-list MURAMs are 8 bits wide, consisting of a six-bit class ID, a seventh bit to indicate that an RBF with minimal radius is recognizing the input vector (i.e. the first RBF to recognize this vector is a probabilistic RBF), and an eighth bit to indicate validity. Figure 3-11 shows the format of a byte in the class-list MURAMs.

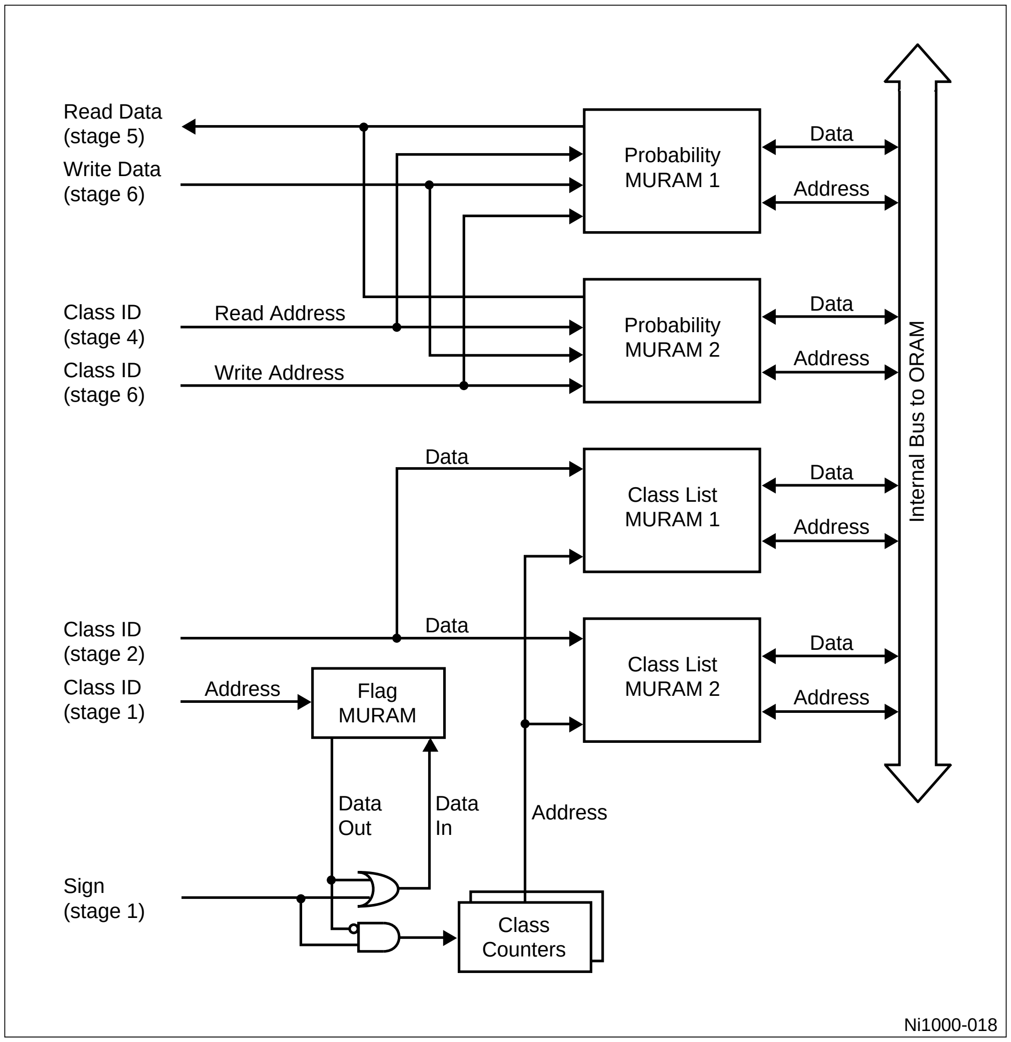

Figure 3-10. Math Unit RAMs (MURAMs)

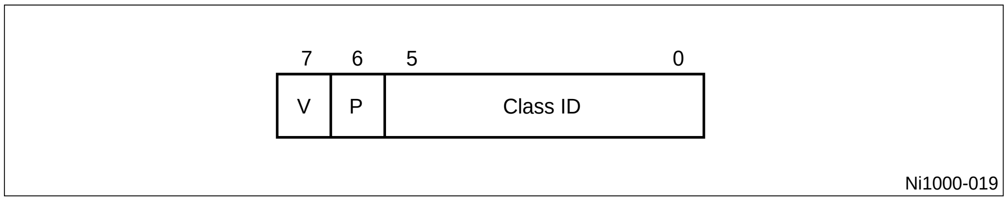

Figure 3-11. Class-List MURAM Word

The fields of a class-list MURAM word are:

- *Class ID—S[0:5]*—Class ID of a class that includes the input vector.
- *Probabilistic—P*—an RBF with the minimum radius recognized the input vector. This indicates that the first RBF to classify this vector was a probabilistic RBF.
- *Valid—V*—this word has been written since reset initialization.

The class-list MURAMs are addressed by a counter. The counter begins at zero and increments as new classes are encountered.

The probability MURAMs consist of a 16-bit floating-point accumulator in the internal format of the Ni1000 Accelerator. The internal format is provided to the outside world in its internal format or translated into an IEEE-compatible format as it passes out the ORAM. Figure 3-12 shows the internal format of a word in the probability MURAMs.

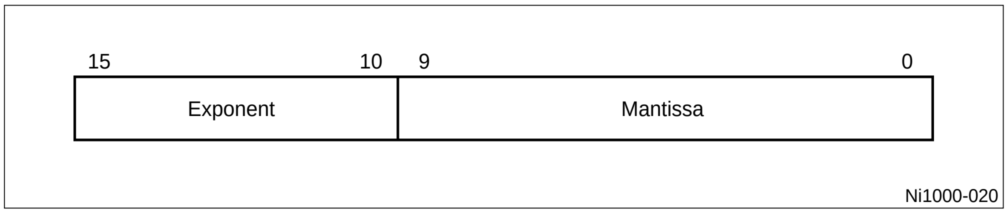

Figure 3-12. Probability MURAM Word

The fields of a probability MURAM word are:

- *Exponent*—six-bit 2's-complement exponent.
- *Mantissa*—10-bit fractional mantissa (i.e. 0 <= mantissa < 1).