TESTING APPARATUSES, SERVERS AND METHODS FOR CONTROLLING A TESTING APPARATUS

Cross-reference to Related Applications

[0001] The present application claims the benefit of the US provisional patent application No. 61/678,701 filed on 2 August 2012, the entire contents of which are incorporated herein by reference for all purposes.

Technical Field

[0002] Embodiments relate generally to testing apparatuses and methods for controlling a testing apparatus.

Background

[0003] Finding the most similar matches to a query vector from a large database of vectors, also known as Nearest Neighbor (NN) search, is a well-known problem in audio, video and other information retrieval, particularly audio/video fingerprinting, which tries to identify a query audio/video clip from a database of reference audio/video content. Exact NN search is challenging when the vectors have high dimensions, where no indexing structure is known to be consistently faster than brute-force search. For approximate NN (ANN), commonly used methods such as Locality Sensitive Hashing (LSH) either become slow due to excessive number of hard disk seeks, or have to use an excessive amount of main memory for indexing, when the NN distance to query vector is

far and the database is large. Thus, there may be a need for more efficient methods and devices.

Summary

[0004] According to various embodiments, a testing apparatus may be provided. The testing apparatus may include: a plurality of cells, each cell including a plurality of terminals, the plurality of cells configured to define a string, wherein each cell of the plurality of cells includes a first terminal; a controller configured to control voltages to the respective first terminals of the plurality of terminals of each cell based on a query pattern, wherein a state of each cell of the plurality of cells is defined by the voltage supplied to the first terminal of the cell; and a determination circuit configured to determine whether the string corresponds to the query pattern based on the states of the plurality of cells.

[0005] According to various embodiments, a server may be provided. The server may include: a receiver configured to receive a query pattern from a client; a testing apparatus; and a transmitter configured to transmit a result determined by the determination circuit of the testing apparatus to the client.

[0006] A testing method may be provided. The testing method may include: controlling voltages to a respective first terminal of a plurality of terminals of each cell of a plurality of cells based on a query pattern, wherein the plurality of cells are configured to define a string, wherein a state of each cell of the plurality of cells is defined by the voltage supplied to the first terminal of the cell; and determining whether the string corresponds to the query pattern based on the states of the plurality of cells.

Brief Description of the Drawings

[0007] In the drawings, like reference characters generally refer to the same parts throughout the different views. The drawings are not necessarily to scale, emphasis instead generally being placed upon illustrating the principles of the invention. In the following description, various embodiments are described with reference to the following drawings, in which:

FIG. 1 shows an illustration of top-2 NNs (nearest neighbors);

FIG. 2 shows an illustration of random projection and quantization of a feature vector;

FIG. 3 shows an illustration of an example of LSH hash indexing and look-up;

FIG. 4A and FIG. 4B illustrate a difference between LSH and media fingerprinting;

FIG. 5 and FIG. 6 show illustrations of weak bits and their enumeration to improve recall and/or reduce storage requirement;

FIG. 7 shows an illustration of vote-count based media fingerprint database;

FIG. 8 shows an illustration of different distribution of vote count;

FIG. 9A shows testing apparatus according to various embodiments;

FIG. 9B shows a server according to various embodiments;

FIG. 9C shows a flow diagram illustrating a testing method;

FIG. 10 shows an illustration of a scheme, for example illustrating comparing multiple projections at a time per vote count update;

FIG. 11A and FIG. 11B show illustrations of the use of weak bits along with multiple projection comparison;

FIG. 12 shows an illustration of embedding linear time trend information into storage format of media fingerprints to filter out unrelated matches;

FIG. 13 shows an illustration illustrating a need for multiple input probes if using less storage space in media fingerprints;

FIG. 14 shows an illustration of avoiding the need for AND/intersection by using large enough samples so that one probe provides enough linear time trend information;

FIG. 15 shows an illustration of an alternative storage approach by enumerating non-zero query/reference time offsets;

FIG. 16 shows an illustration illustrating an effect of radius r on per-projection success rate (p_1) vs. collision rate (p_2) for the range quantizer;

FIG. 17 shows an illustration illustrating $p^*_{collision}$ as a function of range r;

FIG. 18 shows an illustration illustrating a performance comparison;

FIG. 19 shows an illustration of an implementation, with each reference vector corresponding to one column in a DRAM matrix array;

FIG. 20 shows an illustration of a working principle of floating gate transistors in Flash memories;

FIG. 21 shows an illustration of an example of a basic circuit layout of NAND Flash within one column;

FIG. 22A shows an illustration of different charging states;

FIG. 22B shows an illustration of an interlocked fGT pair representation enabling power-savings;

FIG. 23 shows an illustration of an example, where the series is 6 cells long, and query pattern is 10X;

FIG. 24 shows an illustration of an example of reference-side wild-card;

FIG. 25A, FIG. 25B, and FIG. 25C show illustrations, like stated in Table 2;

FIG. 26 shows an illustration of an example of range query;

FIG. 27 shows an illustration of an example of NAND Flash string in schematic and in manufactured integrated circuit;

FIG. 28A shows a schematic symbol of the Flash cell; and

FIG. 29B shows an equivalent circuit.

Description

[0008] Embodiments described below in context of the devices are analogously valid for the respective methods, and vice versa. Furthermore, it will be understood that the embodiments described below may be combined, for example, a part of one embodiment may be combined with a part of another embodiment.

[0009] In this context, the testing apparatus as described in this description may include a memory which is for example used in the processing carried out in the testing apparatus. A memory used in the embodiments may be a volatile memory, for example a DRAM (Dynamic Random Access Memory) or a non-volatile memory, for example a PROM (Programmable Read Only Memory), an EPROM (Erasable PROM), EEPROM (Electrically Erasable PROM), or a flash memory, e.g., a floating gate memory, a charge trapping memory, an MRAM (Magnetoresistive Random Access Memory) or a PCRAM (Phase Change Random Access Memory).

[0010] In an embodiment, a "circuit" may be understood as any kind of a logic implementing entity, which may be special purpose circuitry or a processor executing software stored in a memory, firmware, or any combination thereof. Thus, in an embodiment, a "circuit" may be a hard-wired logic circuit or a programmable logic circuit such as a programmable processor, e.g. a microprocessor (e.g. a Complex Instruction Set Computer (CISC) processor or a Reduced Instruction Set Computer (RISC) processor). A "circuit" may also be a processor executing software, e.g. any kind of computer program, e.g. a computer program using a virtual machine code such as e.g. Java. Any other kind of implementation of the respective functions which will be described in more detail below may also be understood as a "circuit" in accordance with an alternative embodiment.

[0011] Finding the most similar matches to a query vector from a large database of vectors, also known as Nearest Neighbor (NN) search, is a well-known problem in audio, video and other information retrieval, particularly audio/video fingerprinting, which tries to identify a query audio/video clip from a database of reference audio/video content. Exact NN search is challenging when the vectors have high dimensions, where no indexing structure is known to be consistently faster than brute-force search. For approximate NN (ANN), state-of-art algorithms such as Locality Sensitive Hashing (LSH) either become slow due to excessive number of hard disk seeks, or have to use an excessive amount of main memory for indexing, when the NN distance to query vector is far and the database is large.

[0012] Finding similar matches of a query in a database has many applications. For example, similar image search allows a user to find other related images based on a query

image that the user already has. The similarity measure may be defined as the distance between feature vectors of two images, where features with a predetermined definition (such as color, brightness, sharpness) are extracted from an image and form a feature vector. One simple example could be: dividing an image into a 10x10 grid of small blocks, and the average brightness in each block is one feature, and thus forming a feature vector of 100 dimensions. In this example the feature vector is essentially a coarse representation of the image, but other types of features may be used such that from the features alone it is impossible or unlikely to infer a coarse approximation of the image, e.g., to minimize copyright concerns. The distance metric may be defined as the Euclidean (i.e., L2 norm) distance, although other distance metrics such as L1 norm, may also be used. An Lp norm of a d-dimensional vector $\vec{v} = (\vec{v}_{(1)}, \vec{v}_{(2)}, \ldots, \vec{v}_{(d)})$ may be defined as $\|\vec{v}\|_p = (\sum_{i=1}^{d} |\vec{v}_{(i)}|^p)^{1/p}$, and the Lp norm distance between two vectors v and w is $\|\vec{v} - \vec{w}\|_p$. If p = 2, Lp norm is the L2 norm. For p = 1, Lp norm is the L1 norm, and it is to be noted that the absolute operator may be applied before the exponentiation, so that no negative term may be generated during summation. Another example of distance metric is Hamming distance, when the features are discrete (specifically, binary) instead of continuous, and may be defined as the number of differing bits between two such binary vectors. The Hamming distance may also be equivalent to the case of L1 norm distance when the features are binary (i.e., 0 or 1).

[0013] Given a definition of the feature vector and the distance metric, the top-k Nearest Neighbors (NNs) may be defined as the k feature vectors among a given database that are closest in distance to the query feature vector. If k is not mentioned, then the NN may be the top-1 NN.

[0014] FIG. 1 shows an illustration 100 of top-2 NNs. FIG. 1 illustrates top-2 NNs to a query point in a 2D Euclidean space. For example, A and B may be the top-2 NNs for Q, as no other points are closer to Q than A and B (in circle 102, there are only Q, A, and B); for example, C and D are further away from Q than A and B. Unless explicitly noted, the NN may refer to the exact NN, where the definition of closest distance may be exact. It may be found that no known indexing structure is consistently faster than brute-force search for exact NN for high dimensional data where dimension d > 10. Since many similarity search applications involve large d, approximate NN (ANN) is often used instead to reduce search complexity. Commonly used methods for ANN may be Locality Sensitive Hashing (LSH) and its variants.

[0015] FIG. 2 shows an illustration 200 of random projection and quantization of a feature vector 204. A quantization interval 202, a random projection vector 206 and a projected value 208 are shown. The symbol "∃" in FIG. 2 may mean "there exists". For example, "∃D such" may mean that there are D such random projection vectors, and random projection may be applied D times, once per projection vector.

[0016] LSH works by projecting a feature vector 204, quantizing the projected value (see FIG. 2), and then applying this step D times (usually D < d) and concatenating the D quantized values to form a codeword (also known as robust hash or signature) for index building or look-up. The projection is typically the dot-product of a feature vector $\vec{v}$ and a projection vector $\vec{a}$ of the same dimension d, i.e., $\vec{v}.\vec{a} = \sum_{i=1}^{d} \vec{v}_{(i)} \times \vec{a}_{(i)}$. The following chart shows a high level concept implementation of LSH.

$S = \{\vec{v}_i\}_{i=1}^{N}$ // database of reference vectors
Hash_map table; // hash index for database

```
{a_j}_{j=0}^{D-1}                    // D projection vectors
LSH_build()
{
 for (i = 1; i <= N; i++) {
  cw = compute_codeword(v_i);
  table.add(cw, v_i);    // add v_i to the hash bucket under index value cw
 }
}

LSH_lookup(q)
{
 cw = compute_codeword(q);
 S_candidates = table.lookup(cw);  // set of reference vectors with same indexing
codeword
 return v s.t. min_v ||v - q|| where v ∈ S_candidates
}

compute_codeword(v)
{
 cw = "";        // initially empty
 for (j = 0; j < D; j++)
  cw = cw + quantize(v.a_j); // + for concatenating quantized projected values
 return cw;
}
```

Chart 0. Example pseudo-code for LSH

[0017] The feature vector may optionally be converted to a vector of different (usually lower) dimensionality, for example by a number D of random projections where D is the newly desired dimensionality. A projection may be defined as the dot-product of a feature vector $\vec{v}$ and a projection vector $\vec{r}$ of the same dimension d, i.e., $\vec{v}.\vec{r} = \sum_{i=1}^{d} \vec{v}_{(i)} \times \vec{r}_{(i)}$. A dot-product implements a linear projection, and non-linear projections may also be used as projections, depending on user requirement. A projection generates a scalar value, and applying the projection with D (preferably different) projection vectors generates D scalar values and thus forms a D-dimensional vector. If $\vec{a}$'s coefficients follow a random distribution, it is called a random projection. If the distribution is

Gaussian N(0,1), then $|\vec{v}.\vec{a} - \vec{w}.\vec{a}|$ will be distributed as $N(0, \sigma^2)$ where $\sigma = \|\vec{v} - \vec{w}\|_2$. Since a query vector $\vec{q}$ is closer to its NN (denoted $\vec{v}_{NN}$) than to other reference vectors (denoted $\vec{v}_{ot}$), we have Pr[quantize($\vec{v}_{NN}.\vec{a}$) == quantize($\vec{q}.\vec{a}$)] > Pr[quantize($\vec{v}_{ot}.\vec{a}$) == quantize($\vec{q}.\vec{a}$)] $\Rightarrow$ Pr[compute_codeword($\vec{v}_{NN}$) == compute_codeword($\vec{a}$)] > Pr[compute_codeword($\vec{v}_{ot}$) == compute_codeword($\vec{a}$)]. It will be understood that Pr stands for probability. That is, the query codeword is more likely to match NN's than unrelated vectors'. Either the converted vector or the feature vector is then quantized (if the vector elements are already and always discrete, then the quantization step may be optional) on one dimension at a time (see FIG. 2 for example of converted case where random projection is used), and then with the quantized values concatenated to produce a discrete value known as a robust hash, signature, or codeword. A quantizer converts a scalar, continuous value into a discrete value depending on which quantization interval/bin the scalar value falls into, and the same quantizer may be used for all dimensions, or different quantizers may be used for certain dimensions. A c-bit quantizer generally describes c-bit discrete values and can support up to 2^c quantization intervals/bins. Quantization may also be performed after a projection without combining multiple projection values to form the converted vector, since the combining step and the converted vector is not strictly needed.

[0018] Feature vectors from both the database (also referred to as reference feature vectors, or reference vector in short) and the query are applied with the same projection, quantization and concatenation process (same sets of projection vectors are applied to both reference and query feature vectors, etc.), to generate the query feature vector's codeword, which may then be used as indexing value for look-up of a feature vector in a

database. If hash indexing is used, as is the case with LSH, the codeword of the reference vector is usually used to build the hash index, and the codeword of the query feature vector (or query vector in short) is usually used as the hash key for look-up, and all collision items in the hash bucket matching the hash key may be considered candidates for full matching. The full matching is usually the full (e.g., Euclidean) distance check between candidate feature vectors corresponding to the collision items and the query feature vector.

[0019] FIG. 3 shows an illustration 300 of an example of LSH hash indexing and look-up. In the illustration 300, LSH hash indexing with D projections per hash table 304 is shown. Quantized query projections 302 and a HDD (hard disk drive) 306 are shown.

[0020] Typically with projections, if two feature vectors are close in high dimensional space using the given distance metric, then their projected values will also tend to be close with a relatively high probability. So if a query vector is much closer to its NN than to other reference vectors, the probability of the NN quantizing to the same value as the query vector would generally be larger than that of other reference vectors. This means the concatenated key (i.e., the codeword) will be more likely to match between the NN and query vector than between other reference vectors and query vector, as noted in left side of FIG. 3.

[0021] Another application of similarity search is media fingerprinting, whose goal is to identify a query media clip from a database of reference media content. If the media is audio, it is called audio fingerprinting. Audio features such as frequency components may be extracted from consecutive sampling of the audio signals (for both query and reference audio) over some predetermined audio buffer size, say at every 20ms stepping and 300ms

audio buffer, which means the buffer slides 20ms forward in time on each step (hence consecutively) and generates a feature vector per step. A sequence of consecutive codewords, typically as a pre-determined number of codewords, may be called a media fingerprint, and in case of audio signals, an audio fingerprint. The goal of an audio fingerprint search engine is to identify the query audio clip based on similarity of audio signals, which in turn is typically based on similarity of audio fingerprints and/or feature vectors. To reduce complexity, some similarity measure of audio fingerprints, such as Hamming distance (number of differing bits) between audio fingerprints, is typically used instead of full (e.g., Euclidean) distance comparison on feature vectors

[0022] There may be some differences between media fingerprinting and LSH, though they share many common traits. One subtle but important distinction of fingerprinting is the time sequence relationships between successive codewords in media fingerprinting. In case of similar image search, successive images in a database may be completely different and bear no particular relationship to each other, and only one image, namely the query image is being compared against the database on one search. In contrast, an audio fingerprint is a sequence of codewords (where each codeword corresponds to 300ms of audio with 20ms stepping in the previous example), and this entire sequence or at least a significant portion of it has to match well in order to be included in the search results. Both LSH and media fingerprinting may use the same type of hash tables for indexing look-up. Although one could take a media fingerprint of say n-codeword length and its corresponding n d-dimensional feature vectors and concatenate the vectors into a "super-long" n×d dimensional feature vector and apply LSH directly on this super-long vector to implement the equivalent of media fingerprint search, this will generally

increase search complexity and/or storage requirement and is typically not used. The second difference is the storage requirement, which will be described next.

[0023] In the following, challenges in commonly used approaches will be described.

[0024] Similarity search based on ANN is challenging when the feature vector is high dimensional ($d \gg 1$) and the NN reference vector in the database has a large distance to the query vector such that the distances from non-NN vectors to the query vector are not significantly higher than the NN distance ($\|v_{NN} - \vec{q}\| \approx \|v_{ot} - \vec{q}\|$). This is especially relevant when the top-k NN(s) are supposed to have a semantic correspondence to the query, and for example, may happen in case of audio fingerprinting, if the query audio clip is a heavily modified version of an original audio clip. One example of such heavy modification is microphone capture of loudspeaker playback of an audio clip in a noisy environment. This may result in significant distortion in features between query vector and corresponding original vector, and such distortion may significantly increase quantization errors in the generation of codewords, because on some dimension(s), the original vector (or its converted vector if projections are used) may quantize to one value and the query vector (or its converted vector if projections are used) may quantize to a different value, causing the concatenated value (i.e., the codeword) to be different. Since the codeword is usually used for database look-up (see FIG. 3), a different codeword would cause the look-up to miss the hash bucket containing the NN, unless there are other robustness mechanism(s) built into the indexing/search algorithm (example to be described in more detail below with reference to FIG. 5 and FIG. 6). At the bit level, the quantization error may be expressed in terms of the bit error rate (BER), which is the ratio of number of differing bits between a query and a reference codeword to total

number of bits in a codeword. In case of media fingerprinting, this ratio may also be defined between fingerprints instead of codewords. If multi-bit instead of 1-bit quantization is used, then the more general symbol error rate (SER), which is otherwise defined same as BER except the unit of difference is a symbol (representing the quantized value) instead of 1-bit, is usually used. In case of 1-bit quantization, BER is equivalent to SER.

[0025] As BER increases, it tends to be increasingly more difficult to find the NN with both high recall and precision. Recall in ANN is generally defined as the probability that the search results returned from a query contain the desired information, and in case of ANN, the desired information is the true (i.e., exact) top-k NNs (k = 1 for top-1 NN). For an exact NN search algorithm, its recall is 1.0 because it has to return only exact NNs by definition, therefore recall is usually relevant to ANN instead. The precision may be defined as the ratio of desired information to some top portion of the search results after the results have been potentially sorted according to likelihood of being desired information. In case of ANN, a full (e.g., Euclidean) distance check is generally performed to sort candidate matches and only top-k matches with the smallest distances to the query are returned, hence its precision would be the same as its recall and becomes redundant. Therefore, in this disclosure, we define ANN's precision alternatively in terms of collision count: that is, the number of candidate matches. The smaller the collision count is, the higher precision the ANN algorithm is, and any mapping function from collision count to precision that satisfies this behavior may be used. This measure is useful, because having a small collision count reduces the number of full distance checks that have to be performed and thus reduces search time. If the feature vectors are stored

on hard disks, reducing collision count usually means reducing number of disk seeks, and disk seek time is usually a major bottleneck in information retrieval applications.

[0026] With high BER, LSH suffers from at least one of: low recall, slow search time, large storage requirement. The following table illustrates an example, where chance of finding the NN on 1 look-up is only 0.00124, and to achieve a recall of 0.95, L=2418 independent hash tables are needed, so that if 1 table finds the NN, it is success. This requires a huge amount of storage, particularly DRAM if fast search time is needed.

Pr[success on 1 look-up]	$p_s = (1\text{-SER})^D = (1\text{-BER})^D = 0.8^{30} \approx 0.00124$ (1-bit quantization)
Pr[final success after L look-ups]	$p_{final} = 1-(1-p_s)^L$
No. of hash tables	$L \approx 3/p_s = 2418$
Storage requirement	Index $\approx L \times 2^D \times 8B = 19344GB$
	Reference vectors $\approx N \times 100B = 100GB$
Search time (all in DRAM)	< 1ms (uses 19444GB DRAM)
(vectors on HDD)	24.18sec (uses 19344GB DRAM, 10ms HDD seek time)
(all on HDD)	48.36sec (≈0GB DRAM)

Table 1. Inefficiency of LSH. BER = 0.2, N = 10^9, D = 30, p_{final} >= 0.95, 100B per vector, 8B indexing overhead per vector

[0027] As an example, if 1-bit quantization is used, and the true NN exhibits a BER of 0.2, and the database has 1 billion vectors, then LSH may need D = 30 projections and

hence a 30-bit codeword (with 1-bit quantization), in order to limit average hash collision count to 1 vector per hash look-up, since $10^9/2^{30} \approx 1$. However, this makes success probability on one look-up to be $p_s = (1\text{-BER})^D = 0.8^{30} \approx 0.00124$ (for multi-bit quantization, SER should generally be used and $p_s = (1\text{-SER})^D$, but it is also possible to use $p_s = (1\text{-BER})^{D \times c}$ where c is the average number of bits per quantizer), and to obtain a higher final recall (p_{final}), L (preferably independent) hash tables are created so that if one out of L tables finds the true NN, it is success. The needed value of L can be derived from the equation $p_{final} = 1-(1-p_s)^L$. For $p_{final} >= 0.95$ and BER of 0.2, $L \approx 3/p_s \approx 2418$ tables are at least needed. Assume 8 bytes overhead per vector for hash indexing, L tables of D-bit index would require $L \times 2^D \times 8B$ = 19344GB of DRAM, because storing hash index tables on hard disk would be too slow to use. If a feature vector needs 100B to store, then 100GB is needed to store all feature vectors of this database. Since 19344GB is much larger than 100GB, it would be more effective to store the feature vectors also in DRAM rather than on hard disks, in order to reduce full (e.g., Euclidean) distance check time, which requires random access to the candidate feature vector. This results in 19444GB of DRAM usage for 1 billion vectors, or close to 20kB DRAM per vector, making it a very expensive solution. Note if hard disk drives (HDDs) are used to store feature vectors, then it saves 100GB DRAM but on average L disk seeks are needed per LSH ANN search, translating to a very slow 24.18sec per search at 10ms disk seek time if using only one HDD. If most of the hash tables are also stored on HDDs, then most DRAM can be saved but the number of disk seeks will at least double, leading to an even slower >= 48.36sec per search.

[0028] FIG. 4A and FIG. 4B illustrate a difference between LSH and media fingerprinting. FIG. 4A shows an illustration 400 of a query codeword 402 for a 1st hast table 406, a query codeword 404 for a second hash table 408, and a HDD 410. FIG. 4A illustrates search steps of LSH, adapted from FIG. 3 with L = 2 (i.e., 2 hash tables). FIG. 4B shows an illustration 412 of a plurality of codewords 414, a hash table 416, and a HDD 418. FIG. 4B illustrates search steps of media fingerprinting, using n look-ups and fingerprint Hamming distance.

[0029] If the above 1 billion vector database is for media fingerprinting, storage requirement may be reduced, which is the second main difference between media fingerprinting and LSH. If a media fingerprint consists of n codewords, then the look-up miss by one codeword may be mitigated by trying look-ups on all n codewords (see FIG. 4A and FIG. 4B), and as long as at least one codeword look-up is successful, it would find the true NN and lead to the correct original audio clip. If look-up with i-th codeword, the candidate codeword's corresponding fingerprint (the n-codeword reference sequence whose i-th element is the said candidate) is generally compared against the query fingerprint for Hamming distance (Euclidean distance check at codeword or fingerprint level may be used but it is uncommon). FIG. 4A and FIG. 4B illustrate this difference between LSH and media fingerprinting. Instead of requiring L hash tables as in LSH, media fingerprinting exploits the semantic redundancy among n codewords in a fingerprint: as long as one out of n codewords leads to true NN, the search is considered a success. Such semantic redundancy is generally not applicable in case of similar image search, where there is no presumed time sequence information between successive images. Of course, media fingerprinting may also use multiple hash tables to further

improve recall, but it usually needs fewer hash tables than LSH for the same p_{final}, due to this inherent redundancy.

[0030] FIG. 5 and FIG. 6 show illustrations of weak bits and their enumeration to improve recall and/or reduce storage requirement. FIG. 5 shows an illustration 500 illustrating a quantization interval 502, a query vector 504, a reference vector 506, a random projection vector 508, a projected value 510 of the query, and a projected value 512 of the reference vector. FIG. 5 illustrates a cause of an issue: the query or reference vector's projected value lying too close to quantization boundary. FIG. 6 shows an illustration 600 of an initial query codeword 602, a codeword with weak bits 604, an illustration 606 of values (or variations) the codeword with weak bits 604 may take, and an illustration of the number 608 of variations of the codeword with weak bits 604. FIG. 6 illustrates that identifying and enumerating (on a query-side) weak bits may improve recall w/o (without) many hash tables.

[0031] For both LSH and media fingerprinting, within one codeword look-up, the miss may be mitigated by enumerating "weak bits" in a codeword , i.e., the bits that are most likely to have flipped from original to query, generally due to their pre-quantization value (which is the projected value when projection is used or the feature value otherwise) lying too closely to the quantization boundary, as illustrated in FIG. 5. In case of 1-bit quantization, a weak bit is enumerated for 0 and 1, so enumerating w weak bits means generating 2^w variations of the query codeword for look-up. If multi-bit quantization is used, a weak bit may be more precisely referred to as a "weak projection", and a small range covering the quantized value is used for each such projection during enumeration. Usually the "weak bits" are determined on the query codeword because then the weak bit

positions are fixed at query time, whereas weak bit positions in the reference codeword are often different across the database, making it impractical to enumerate all the reference side weak bits (with an exception for the invention in this disclosure, to be described in more detail below). FIG. 6 illustrates weak-bit enumeration for a 6-bit long codeword, although in practice a codeword is usually much longer.

[0032] Weak-bit enumeration may be used to reduce L and thus storage requirement, because by enumerating variations of weak bits, firstly the effective length of a codeword is shorter, and secondly the remaining non-enumerated bits are expected to have a lower BER compared to the average BER of the entire codeword (SER in case of multi-bit quantization), thus improving p_s and reducing L. Commonly used methods suggest the prospect of reducing L by 10x to 20x on some difficult data sets. If we assume L is reduced 20x in the above example, the indexing would still require 967GB, and feature vector still 100GB, with total of 1067GB DRAM. This corresponds to about 1kB of DRAM per vector, and if the database has 1 trillion vectors, which may be expected with rapid growth of Internet information, it would require roughly 1000TB of DRAM, which implies that thousands of high-end PCs (roughly 1 large data-center) would be needed to meet the storage and computing requirements to achieve 0.95 recall on a 1 trillion vector database. Thousands of PCs are expensive to purchase, and also take up huge amount of space and consume enormous amount of electricity.

[0033] Thus, what is needed is a solution that can provide high recall and high precision, as well as with fast search time, at low cost, and perform efficiently on large-scale, difficult data sets. According to various embodiments, a novel algorithm may be provided that can attain these goals.

[0034] A vote-count method may provide high recall and precision on relatively difficult data sets, with an implementation that achieves relatively fast search time and low cost, thus making it potentially useful for large-scale data sets. However, the vote-count method may still have fairly high collision count if the data set is significantly correlated and not random enough. High collision count translates to longer search time due to the need for full distance check on all candidate matches.

[0035] The vote-count method may also use random projections as LSH does, but instead of concatenating D quantized projections to form a codeword as in LSH, vote-count performs L projections (where this L may be different from LSH's L in value) and quantization for both reference and query vectors, and increment a vote counter for each reference vector if the query and reference matches on a projection after quantization. Below is the illustrative pseudo code, with O.A[j] and Q.A[j] denoting the quantized projected value of reference vector O and query vector Q, respectively, on j-th projection. It is to be noted that if projections are not used, O.A[j] and Q.A[j] may denote the quantized value of the feature vector on j-th dimension. If the feature vectors' dimension d > L, then a subset of those d dimensions may be chosen that the new feature vectors have dimension L. If d < L, then a smaller L may instead be chosen.

```
for (j = 0; j < L; j++)
    ∀reference object O, if (O.A[j] == Q.A[j]) then O.counter++;
∀O, return O as candidate if O.counter >= threshold T*L
```

Chart 1. Basic vote-count process flow

[0036] FIG. 7 shows an illustration of vote-count based media fingerprint database (for example with F codewords/sec, L projections, 3-sec long). A quantized projected value of a query frame I 702 and data for a plurality of frames 704 are shown. Each column corresponds to a reference vector, with its rectangle on j-th row denoting its reference quantized projection value on j-th projection (here j = 0). The arrow to the right of the circle denotes equality comparison.

[0037] After comparing L projections, the reference vectors whose vote count exceed T*L are chosen as candidate matches and then checked for full distance, and those with the smallest distances are returned as the top-k NN(s). The database may be stored according to FIG. 7, and the update of vote counters for each projection can be efficiently implemented using the approach as described in more detail below.

[0038] The probability distribution of the vote count value after all L projections are compared can be approximated as binomial distributions. For true NN, under 1-bit quantization, Pr[quantize($\vec{v}_{NN}.\vec{a}$) == quantize($\vec{q}.\vec{a}$)] = probability for one projection to match between query and true NN reference := p_1 = 1-BER (or more generally p_1 = 1-SER for multi-bit quantization), and Pr[quantize($\vec{v}_{ot}.\vec{a}$) == quantize($\vec{q}.\vec{a}$)] = probability for one projection to match between query and unrelated reference := p_2 = 0.5 (or more generally $p_2 = 0.5^c$ for c-bit quantization if each bin has equal probability mass). Therefore, the vote count would have two binomial distributions, one with a peak at p_1*L for true NN, another with a peak at p_2*L for unrelated reference vectors. Suitably chosen L and threshold factor T can make the probability of unrelated vector's vote count exceeding T*L so low that very few of unrelated vectors become candidate matches, i.e.,

having very low collision count (thus high precision and faster search time). This may be illustrated in FIG. 8.

[0039] FIG. 8 shows an illustration 800 of different distribution of vote count between true NN 804 and unrelated vectors 802; for 1-bit quantization, p_1 = 1-BER, p_2 = 0.5.

[0040] On large, difficult data sets, vote count's two binomial distributions generally may filter out unrelated reference vectors much more effectively than LSH's hash buckets.

[0041] However, if reference vectors in the data set are highly correlated, e.g., due to heavy clustering, then a query vector may have an entire cluster of reference vectors close to it, causing high collision count, and make the basic vote-count algorithm perform inefficiently. In case of audio fingerprinting, this can happen if the feature extraction algorithm makes the feature too robust but not unique (e.g., random) enough.

[0042] Furthermore, the commonly used method may update the vote count one projection at a time, thus requiring L rounds of update. If this number of rounds can be reduced, it may lead to a reduction in search time.

[0043] The method may always update the vote count at each projection, but due to the presence of weak bits or weak projections, some projections may be better left out in the update of the vote count.

[0044] The power consumption of the corresponding hardware implementing the vote-count algorithm where the hardware may be Dynamic Random Access Memory (DRAM), NAND Flash or NOR Flash, as described in more detail below. The method may be desired to be implemented in hardware because updating N vote counters on an

N-vector database would be very slow if done in software for large N. However, such hardware may consume a lot of power. On a one-trillion vector dataset, if implemented on NAND Flash memory, and each column consumes 1μA and supply voltage V_{DD} is 3V, then power consumption during vote-count could be as high as $P_{max} = 10^{12}*1\mu A*3V =$ 3MW. Due to NAND Flash memory's characteristics, as will be described in more detail below, the average power may be 0.5*3MW = 1.5MW. Such high power may be both a tough challenge for cooling and a serious concern for fire hazard if the semiconductor transistors do not lose integrity and melt first, and cooling itself may consume even more power. If implemented on NOR Flash or DRAM, search time may be faster but NOR Flash and DRAM are much more expensive, and power consumption may likely to be even higher.

[0045] According to various embodiments, an ANN search method and its supporting hardware design (in other words: ANN search devices) may be provided that can achieve fast search time, high recall and precision, and save orders of magnitude in power and energy consumption, on large and difficult data sets, compared to state-of-art approaches.

[0046] According to various embodiments, devices and methods may be provided for high performance search methods for large-scale information retrieval.

[0047] FIG. 9A shows a testing apparatus 900 according to various embodiments. The testing apparatus 900 may include a plurality of cells 902. Each cell of the plurality of cells 902 may include a plurality of terminals. The plurality of cells 902 may be configured to define a string. Each cell of the plurality of cells 902 may include a first terminal. The testing apparatus 900 may further include a controller 904 configured to control voltages to the respective first terminals of the plurality of terminals of each cell

of the plurality of cells based on a query pattern. A state of each cell of the plurality of cells 902 may be defined by the voltage supplied to the first terminal of the cell. The testing apparatus 900 may further include a determination circuit 906 configured to determine (in other words: to test) whether the string corresponds to the query pattern based on the states of the plurality of cells. The plurality of cells 902 and the controller 904 may be coupled with each other, like indicated by a first line 908, for example electrically coupled, for example using a line or a cable, and/ or mechanically coupled. The plurality of cells 902 and the determination circuit 906 may be coupled with each other, like indicated by a second line 909, for example electrically coupled, for example using a line or a cable, and/ or mechanically coupled. The first line 908 from the controller 904 to the plurality of cells 902 may represent the control voltages to the first terminals (for example G-terminals). The second line 909 from the determination circuit 906 to the plurality of cells 902 may represent the serial circuit (e.g., the bit-line) being sensed like will be described in more detail below. According to various embodiments, the per column serial circuit, e.g., the bit-line, may be sensed (either in voltage or current) to determine whether the pattern matches, like will be described in more detail below. According to various embodiments, the state of a cell may refer to a conductibility state, for example indicating whether the cell is conductive under a probing voltage.

[0048] According to various embodiments, the plurality of cells 902 may be connected in series to define the string.

[0049] According to various embodiments, states of each cell of the plurality of cells may include a conductable state and a non-conductable state.

[0050] According to various embodiments, each cell of the plurality of cells may be configured to be in the conductable state if the voltage supplied to the first terminal of the cell is in a pre-determined range.

[0051] According to various embodiments, the determination circuit 906 may be configured to determine that the string corresponds to the query pattern if the states of all cells of the plurality of cells 902 is the conductable state.

[0052] According to various embodiments, each cell of the plurality of cells 902 may be a transistor.

[0053] According to various embodiments, the first terminal of each cell of the plurality of cells 902 may include or may be a gate terminal.

[0054] According to various embodiments, two cells of the plurality of cells 902 define one bit of the string.

[0055] According to various embodiments, a bit of the string may have a value selected from a list of values consisting of: low; high; and don't care.

[0056] According to various embodiments, the testing apparatus 900 may be configured to provide a test for at least one of: audio fingerprint; video fingerprinting; content identification; anti-piracy; similar image search; example-based image processing; super-resolution; de-noising; image compression; bioinformatics; DNA pattern matching; biometrics security; voiceprint; or faceprint.

[0057] FIG. 9B shows a server 910 according to various embodiments. The server 910 may include a receiver 912 configured to receive a query pattern from a client (not shown). The server 910 may further include a testing apparatus 914 according to various embodiments, for example the testing apparatus 900 of FIG. 9A. The server 900 may

further include a transmitter 916 configured to transmit a result determined by the determination circuit of the testing apparatus 914 to the client. The receiver 912, the testing apparatus 914, and the transmitter 916 may be coupled with each other, like indicated by lines 918, for example electrically coupled, for example using a line or a cable, and/ or mechanically coupled. In other words, according to various embodiments, a network service may be provided where a query request is sent from a client to a server, processed according to various embodiments as described herein, and the query result returned to the client.

[0058] FIG. 9C shows a flow diagram 920 illustrating a testing method. In 922, voltages to a respective first terminal of a plurality of terminals of each cell of a plurality of cells may be controlled based on a query pattern. The plurality of cells may be configured to define a string. A state of each cell of the plurality of cells may be defined by the voltage supplied to the first terminal of the cell. In 924, it may be determined (in other words: it may be tested) whether the string corresponds to the query pattern based on the states of the plurality of cells.

[0059] According to various embodiments, the cells of the plurality of cells may be connected in series to define the string.

[0060] According to various embodiments, states of each cell of the plurality of cells may include or may be a conductable state and a non-conductable state.

[0061] According to various embodiments, each cell of the plurality of cells may be in the conductable state if the voltage supplied to the first terminal of the cell is in a pre-determined range.

[0062] According to various embodiments, the testing method may further include: determining that the string corresponds to the query pattern if the states of all cells of the plurality of cells is the conductable state.

[0063] According to various embodiments, each cell of the plurality of cells may include or may be a transistor.

[0064] According to various embodiments, the first terminal may include or may be a gate terminal.

[0065] According to various embodiments, two cells of the plurality of cells may define one bit of the string.

[0066] According to various embodiments, a bit of the string may have a value selected from a list of values consisting of: low; high; and don't care.

[0067] According to various embodiments, the testing method may further include: providing a test for at least one of: audio fingerprint; video fingerprinting; content identification; anti-piracy; similar image search; example-based image processing; super-resolution; de-noising; image compression; bioinformatics; DNA pattern matching; biometrics security; voiceprint; or faceprint.

[0068] An improved vote-count algorithm, which will also be referred to as enhanced vote-count or vote-count++ and to distinguish from the (basic) vote-count, may be provided which may addresses these issues. In addition, a power-efficient hardware design with preferred embodiments by modifying NAND Flash, which is also an integral part of enhanced vote-count, may be provided, like will be described in more detail below.

[0069] In the following, a description of comparing multiple projections per vote count update according to various embodiments will be provided.

[0070] As will be described in more detail below, an implementation may be provided that can efficiently update the vote counters in parallel. If each projection takes time τ to update all vote counters, then it will take $L*\tau$ plus some overhead to perform one ANN search.

[0071] If m projections can be compared at a time for each update, and the vote counter is incremented only if all said m projections compare equal between query and reference, then search time may be about $L/m*\tau$. This may assume m projections can be compared within time τ, which is validated as will be described in more detail below.

[0072] FIG. 10 shows an illustration 1000 of this scheme, for example illustrating comparing multiple projections at a time per vote count update, m = 3 (here j = 0). It is to be noted that in contrast to FIG. 7, 3 projections 1002 instead of 1 are compared at a time. For example two frames 1004 and 1008 are shown, but further frames may be present, like indicated by dots 1008. For example, L rows (or projections) may be present, like indicated by 1010.

[0073] The following pseudo-code shows an example process flow of how such multiple projection comparison works. ceil() is the ceiling (rounding up) function. It is to be noted that O.A[j..j+m-1] == Q.A[j..j+m-1] is a short-hand for (O.A[j] == Q.A[j]) && (O.A[j+1] == Q.A[j+1]) && ... && (O.A[j+m-1] == Q.A[j+m-1]).

```
for (j = 0; j < L; j += m)
    ∀reference object O, if (O.A[j..j+m-1] == Q.A[j..j+m-1])
        then O.counter++;
∀O, return O as candidate if O.counter >= threshold T*ceil(L/m)
```

Chart 2. Example process flow of multiple projection comparison

[0074] If L is not divisible (or dividable) by *m*, then at a certain loop above, say the last loop, the number of multiple projections being compared may be less than m accordingly, for example this number may be L%m, i.e., the remainder of L/m. The above example assumes an equal number of projections (*m*) compared at a time, but more generally it is also acceptable to change this number m at some loops or at each loop. Even more generally, the said multiple projections do not need to form a partition of all L projections and could be overlapping across loops. What this means is, some projection(s) may be reused across loops, which may save on time computing projections and/or storage capacity, but potentially at the expense of more correlation in vote count updates across loops. It is to be noted that in the above pseudo-code, the counter is incremented (by one) per *m* projections, instead of per projection, upon an overall equality match on the *m* projections. If the counter is incremented by *m* per *m* projections and ceil(L/m) is replaced by L, then it is functionally equivalent to increment by one per *m* projections. Of course like mentioned in Sec. II, if projections are not used, O.A[j..j+m-1] may denote *m* quantized values of the feature vector on j-th to (j+m-1)-th dimensions. When L is divisible by m, L/m is also referred to as L'.

[0075] As will be described in more detail below, an efficient implementation that can perform equality comparison on all m projections with time τ may be provided, thus reducing search time to roughly $L/m*\tau$. This may be achieved without requiring m times the hardware resources during implementation. This implementation may also reduce power consumption exponentially, as will be described in more detail below, by ensuring non-negligible power is consumed for a column only when all *m* projections of this

column match with those of the query vector. The expected power during vote-count is roughly $p_2^m \times P_{max}$, where P_{max} is the maximum possible power consumption when all columns consume non-negligible power. As has been described above, an example of P_{max} is 3MW for a one-trillion vector database implemented on NAND Flash. With m = 8 and $p_2 = 0.5$, for example, expected power will be roughly $1/256 \times P_{max}$, or about 11.7kW. A higher m will lead to lower power consumption. However, for m > 1, the binomial distribution curves in FIG. 8 will shift and generally L needs to be increased in order to maintain the same recall and same low collision rate. An increased L, as will be described in more detail below, may translate to more hardware resources and thus higher cost needed during implementation. Nevertheless, if m is carefully chosen, the benefits of power-saving, which can be several orders of magnitude, far outweigh the additional cost in hardware implementation. Furthermore, as will be described further below, an example configuration may achieve high recall, low collision, fast search time, as well as very low power consumption on a one-trillion vector database with a BER $\approx$ 0.2.

[0076] In the following, the introduction of weak bit into vote-count according to various embodiments will be described.

[0077] If a query or reference bit (or more generally a projection for multi-bit quantization) is weak, then its value is unreliable, so its corresponding equality comparison result may also tend to be unreliable, hence it may be better to remove such weak bits or projections from being participating in the vote-count process. Since the query side weak bits or projections are fixed given a query vector but reference side weak bits or projections can vary across reference vectors, one way of introducing weak bit into vote-count is by disregarding the number of weak bits (or more generally, weak

projections), say w such projections, from L, and the vote counters are updated only if a query side projection is not weak. At the end of the loop, the vote counters are compared to T*(L-w) instead of T*L. The pseudo code below illustrates an example of how this works.

```
w = 0;
for (j = 0; j < L; j++)
    if (Q.A[j] is weak) w++;
    else
      ∀reference object O, if (O.A[j] == Q.A[j]) then O.counter++;
∀O, return O as candidate if O.counter >= threshold T*(L-w)
```

Chart 3. Example process flow of weak-bit with vote-count

[0078] In the following, range query, for example using a generalized weak bit for vote-count, according to various embodiments will be described.

[0079] When multi-bit quantization is used, a weak bit is more precisely referred to as a weak projection. For a query side projection, instead of using a single discrete quantized value x_1=quantize($\vec{q}.\vec{a}$), a (preferably small) range covering x_1, such as [x,y](denoting a range of discrete numbers from x to y), may be used. If a = quantize($\vec{v}.\vec{a}$), i.e., reference vector $\vec{v}$'s corresponding quantized projected value is a, then instead of testing (x_1 == a) during vote-count's equality comparison, a range query test of (a ∈ [x,y]) may be used, which will have a higher chance of success in equality comparison. The following (Chart 4) may be an example pseudo-code using query side

range query, and note here instead of excluding weak projections from vote-count process, they are used toward vote-count update, which is another way of using weak bits and projections. In Chart 4 O.A[j] refers to a single quantized value and Q.A[j] a quantized value range, and the $\in$ operator tests whether O.A[j] falls within the range represented by Q.A[j].

```
for (j = 0; j < L; j++)
    ∀reference object O, if (O.A[j] ∈ Q.A[j]) then O.counter++;
∀O, return O as candidate if O.counter >= threshold T*L
```

Chart 4. Example process flow of query-side range query with vote-count

[0080] Such range query test capability can be implemented efficiently like will be described in more detail below. In addition, this implementation also allows reference-side range query: if a reference vector $\vec{v}$'s projection is best described using a range [a,b], and query's quantized projected value is x=quantize($\vec{q}.\vec{a}$), then the test of (x $\in$ [a,b]) can also be implemented efficiently like will be described in more detail below. If reference-side range query is used, the if-test (O.A[j] $\in$ Q.A[j]) in Chart 4 would be changed to (Q.A[j] $\in$ O.A[j]). Furthermore, this implementation allows range queries from both reference side (say [a,b]) and query side (say [x,y]) to be used together, which would test whether [x,y] $\cap$ [a,b] is empty. If it is not empty (with the if-test (O.A[j] $\in$ Q.A[j]) in Chart 4 changed to (Q.A[j] $\cap$= O.A[j]), where the $\cap$= operator tests whether the intersection of two ranges represented by left and right operands is empty), the comparison is successful and the vote counter for that reference vector is incremented. If

a single value say x instead of a range is used, then this value is regarded as a trivial range of [x,x] with only one element in the range, and this provides valid definition whether only either query or reference side (non-trivial) range query is used, or both are used, or none are used. If a query side range used in range query encompasses all allowed numbers in the corresponding quantizer, then this range will always match any validly defined reference side range, and becomes a "always-match" or "don't care" match pattern. Conversely, the same can happen for reference side range if it encompasses all allowed numbers in the corresponding quantizer, so such "always-match" or "don't care" pattern can occur at and be used for both reference and query side. In case of 1-bit quantization, such "always-match" or "don't care" pattern degenerates to and is a weak bit.

[0081] In the following, intelligently coupling multiple projection comparison with range query according to various embodiments will be described.

[0082] If multiple projection comparison is properly coupled with range query or weak bit, it can further improve the performance of vote-count. An implementation according to various embodiments that couples these two capabilities, and allowing reference and/or query side range query, will be described in more detail below, and still only takes time τ for each vote count update. In one extreme configuration, the entire codeword may be compared at a time (i.e., L = m) with range query in some projections of the codeword (reference and/or query side), and the threshold factor T should be chosen such that $0 < T*\text{ceil}(L/m) <= 1$, meaning that as long as the vote count (which could be only as large as 1) is non-zero (i.e., 1), the reference vector is chosen as candidate, and the whole comparison can be done in time τ. The following is the example

pseudo code, with the $\cap=$ operator denoting range query comparison as previously described, with one operand being the query side range, and another operand being the reference side range, and returns true if and only if the two operand ranges have non-empty intersection. If the operands use the notation [j..j+m-1], as is shown in Chart 5, it means m projections are tested for $\cap=$ at a time and all m of them must test true for the if-test to be true.

```
for (j = 0; j < L; j += m)
    ∀reference object O, if (O.A[j..j+m-1] ∩= Q.A[j..j+m-1])
                then O.counter++;
∀O, return O as candidate if O.counter >= threshold T*ceil(L/m)
```

Chart 5. Example process flow of weak-bit with vote-count, m <= L

[0083] FIG. 11A shows an illustration 1100 of the use of weak bits along with multiple projection comparison, m = 3 (here j = 0). Various elements shown in FIG. 11A may be similar to elements shown in FIG. 10, so that the same reference signs may be used and duplicate description may be omitted. A x denotes a weak bit, and here query side has a weak bit 1102 for projection 1, and reference side's frame 1 vector has a weak bit 1104 for projection 2. FIG. 11A shows use of weak bits, which is the special case of range query, on multiple projections. It is to be noted that both query and reference side weak bits are shown to illustrate the flexibility of mixing and matching various capabilities according to various embodiments.

[0084] In FIG 11A, the value of query projection 0 and 2, and various reference projection values (except the wild-card) are not shown. In FIG 11B, specific values are shown for both query and reference, so query 1X0 will match frame 1, but not frame 0.

[0085] In the following, reducing collision count for media fingerprinting by intelligently exploiting linear time trend according to various embodiments will be described.

[0086] In media fingerprinting, feature extraction may be designed to emphasize robustness as opposed to uniqueness/differentiation (where differentiation can mean distance separation between unrelated feature vectors), in order to cope with heavy modification in the query clip. However, this may increase collision count in candidate matches and sometimes even break the semantic meaning of NN matches: that is, the actual corresponding original content's feature vector may even become 2^{nd} or 3^{rd} NNs etc., instead of being the top-1 NN, and this would reduce recall unless top-k ($k > 1$) matches are returned as opposed to top-1, but at the cost of more unrelated matches in the search result. Whether manifested as higher collision count or weakening of NN semantics, such reduced differentiation would increase unrelated matches and result in longer search time. Here a match can refer to either a candidate match or a match that already passed full (e.g., Euclidean) distance check to be among top-k in a search result.

[0087] Media fingerprinting has the inherent time sequence semantics, so one method to filter out the number of unrelated matches (candidate or top-k) is to perform several searches at query timestamp t, t+1, t+2, etc., and then perform a linear trend detection over time-axes (x-axis being query timestamp, y-axis being timestamp of matched content) on the search results after initially sorting the search results by matched content

ID. That is, the matches under the same content ID are tested for presence of linear trend over time, and only matches that exhibit linear trend may be returned as final matches. This exploits linear time trend characteristics inherent in the query and reference content, and is generally checked for all matched content IDs, and usually the matches exhibiting the strong linear trend are considered the real matches. However, if the collision count and/or the k in top-k NNs is high in each search, the method just described may run slowly.

[0088] To deal with the limitation of the above filtering method, according to various embodiments a method may be provided next that can reduce collision count in candidate matches by applying linear time trend constraint during the vote-count steps. Instead of updating the vote counters for one query timestamp at a time, the vote counters may be updated for quantized projected values for multiple query timestamps at a time. For example, query timestamps t, t+1, and t+2 may be checked simultaneously, by placing the quantized values of a reference feature vector from a set of projections and from t_r, t_r+1, t_r+2, within the same column of storage, as illustrated in the pseudo-code of Chart 6 and in FIG. 12, as opposed to FIG. 7. Here O.A_0[i], O.A_F[j], O.A_2F[j] denote quantized values on j-th projection for 3 reference vectors with frame or codeword offset 0, F, and 2F, respectively, Q.A_0[i] etc. are similarly defined. If an unrelated match at reference timestamp t_r happens to be a top-k NN (whether exact or ANN) at query timestamp t, it is very unlikely for its successor vector at t_r+1 to also be a top-k NN (whether exact or ANN) at query timestamp t+1, and similarly for t_r+2 and t+2, etc. A sampling interval other than 1 sec may be used. At least 2 samples (i.e., 2 query timestamps) are needed to verify the presence of linear trend. A linear slope of 1.0 is

assumed, because query and reference content are generally extracted at the same rate and are expected to have the same playback speed. If the extraction rate and/or playback speed is different for query and reference, the assumed/expected slope will be different from 1.0 but still fixed and known as long as the extraction rate and playback speed are fixed and known for query and reference. If the playback speed of query clip is changed relative to reference by a slight but unknown ratio, then several possible speeds may be enumerated, e.g., at 0.9, 1.0, 1.1 (relative to reference playback speed), and for each speed, Q.A_F should come from the query frame that corresponds closest in time to 1-sec. If guessed query playback speed is 0.9, then query frame i+F/0.9 (with rounding) should be used instead of i+F for deriving Q.A_F, and similarly for Q.A_2F.

```
for (j = 0; j < L; j++)
    ∀reference object O, if ((O.A_0[j] == Q.A_0[j])
                            && (O.A_F[j] == Q.A_F[j])
                            && (O.A_2F[j] == Q.A_2F[j]))
                        then O.counter++;
∀O, return O as candidate if O.counter >= threshold T*L
```

Chart 6. Example process flow of weak-bit with vote-count

[0089] FIG. 12 shows an illustration 1200 of embedding linear time trend information into storage format of media fingerprints to filter out unrelated matches, reducing collision count and search time. 1-bit quantization and L projections are used as example, feature vectors extracted at F frames/sec (F fps). 1 rectangle denotes 1 projection, 1 circle

denotes 1 quantized projected value of query vector. Notations and settings may be the same as in FIG. 7 otherwise. For example a query frame 1202 and a plurality of frames 1204 are shown.

[0090] FIG. 12 (c.f. FIG. 7) shows an example of efficiently storing quantized values from t_r, t_r+1,t_r+2 and querying with quantized values from t, t+1,t+2 (time=t at i-th query frame) simultaneously, while FIG. 7 shows a conventional approach to storing and querying media fingerprints. In FIG. 12, only 1 projection from a given query timestamp is compared at a time, but it is possible to add multiple projection comparison in FIG. 10 (which is across projections) on top of time trend information (which is across time), and may even combine reference and/or query side range query capabilities. In FIG. 12, if m projections from a given query timestamp is compared at a time, then each circle should be expanded to m circles, to represent the m projections. Weak bits and range queries such as in FIG. 11A may also be used together with time trend information.

[0091] Alternatively, instead of using the same j-th projection such as for frames with time offset 0, F and 2F in Chart 6 and FIG. 12, different projections may be used such as across these 3 types of frames, as long as the choice of projections is same between query and reference side.

[0092] It is to be noted that in FIG. 12, immediately after the column storing frame F-1,2F-1,3F-1, the next column stores frame 3F,4F,5F, because the quantized projections for frames F+1 to 3F-1 are already stored in preceding columns. This saves storage space, but would require more query probing (where a probing is an equality comparison over one or more projections), for example, as illustrated in FIG. 13, like will be described below. When combining linear time trend information from 3 samples, then 3 sub-

searches instead of 1 are necessary to cover 3 possible time offsets (e.g., 0, 1, and 2 seconds) between query and reference, for storage format of FIG. 12. These are called sub-searches because they are not an entire ANN search in itself, but instead portions of the entire ANN search.

[0093] FIG. 13 shows an illustration 1300 illustrating a need for multiple input probes if using less storage space in media fingerprints. Data for a first sub-search 1302, data for a second sub-search 1304, and data for a third sub-search 1306 are shown. A circle in FIG. 13 may have same meaning as in FIG. 12, and q_0 may denote the projection is from query frame i, q_F from frame i+F, q_{2F} from frame i+2F, and the × denotes an always-match pattern (wild-card).

[0094] The 1st sub-search in FIG. 13 may be self-explanatory, but the 2nd sub-search may be more complicated, because effectively two probes are needed, since the query clip's q_0 may correspond to reference content's frame F, so q_0 need to be shifted downward by 1 row during matching, and the top row should be ignored (by using a "always-match" range query pattern). However, q_{2F} still need to be matched, as top row in another probe, with bottom 2 rows ignored (using "always-match"), and the two probes' result should be intersected (ANDed). This is similar for 3rd sub-search. Implementing the intersection between probes is expensive. Alternatively, we can choose a larger number of samples in linear time trend, but use a subset of projections that is guaranteed to fit within the same column.

[0095] FIG. 14 shows an illustration 1400 of avoiding the need for AND/intersection by using large enough samples so that one probe provides enough linear time trend

information. Here 5 samples are shifted in round-robin manner, and at least 3 samples are enforced within one probe.

[0096] For example, 5 samples may be used simultaneously for some time offset such as in 1st sub-search of FIG. 14, but 3 samples may also be used simultaneously without having to apply intersection and yet still capture enough linear time trend information, as illustrated in 3rd and 4th sub-searches of FIG. 14.

[0097] FIG. 14 shows how AND/intersection can be avoided. Using 5 samples (4 seconds apart), one probe can always contain at least 3 samples (if consecutively sampled, at least 2 seconds apart), so that unlike FIG. 13, the other probe is not necessary. In this example, the 1st sub-search has 5 samples, but one may use 3 or 4 samples as well, at a slight reduction in filtering capability but may improve pattern match probability slightly. FIG. 14 gives an intuitive order of search (from 1st to 5th), which corresponds to the round-robin shift of the input pattern, but in practice any order of search may be applied. Non-consecutive sampling, e.g., q_0 x q_{2F} x q_{4F}, may also be used. Based on the implementation like will be described in more detail below, one may run all 1st sub-searches (over all projections) on say in this example a 4-sec span of query clip, update the vote counters, and generate 1st ANN sub-results (e.g., generate candidate matches by applying >= T*L test on vote counters); then all 2nd sub-searches are run to generate 2nd ANN sub-results, etc. All these ANN sub-results can then be combined to generate the final ANN search results (with full distance check if necessary). If using the pseudo-code in Chart 6, either 5 modified copies of the code are needed with each having the corresponding if-test match pattern for 1st to 5th sub-searches, or one modified copy of the

code where the caller can supply the if-test match pattern for 1st to 5th sub-searches as an input parameter to the said code.

[0098] Alternatively, if storage space is not a concern, the quantized projections may be stored more than once, by enumerating non-zero time offsets that the query clip may have with respect to its corresponding reference content.

[0099] FIG. 15 shows an illustration 1500 of an alternative storage approach by enumerating non-zero query/reference time offsets. For brevity, the frame numbers are displayed instead of the rectangles in FIG. 12.

[00100] FIG. 15 shows an example, where all such non-zero time offsets are enumerated, and since 3 samples are used for linear time trend, the storage space usage is 3 times as that in FIG. 12. However, the approach in FIG. 15 has one advantage over FIG. 12: it only needs 1 sub-search (and with no AND/intersection required) instead of 3 sub-searches as in FIG. 13, because the extra storage already represents all possible time offsets between query and reference content. Therefore, this approach saves search time. Also, with the approach in FIG. 15, the pseudo-code in Chart 6 can be used as-is without modification.

[00101] To generalize the storage layout in FIG. 12, if the number of unique relative snapshot offsets stored on the said string is U (=3 in FIG. 12), where a snapshot is a timepoint (e.g., a frame) for feature extraction, and column 0 corresponds to time 0 and frame 0 of the reference content, then at column i, where n = int(i/F), i.e., n is the integer quotient of i/F, and j = i%F and % is the integer modulo operator, quantized project values derived from frame n*U*F+j, n*U*F+j+F,..., n*U*F+j+(U-1)*F are stored.

[00102] In the following, a range quantizer for reducing collision rate while retaining recall according to various embodiments will be described.

[00103] In the vote-count method, the feature vectors and the query vector (or their converted vectors if projections are used) are quantized, and the decision of the vote count updating is based on comparison result on quantized vectors. The quantizer used in the vote-count method thus determines the granularity of the comparison and hence it plays an important role in determining the overall performance of the vote count algorithm. For example, it may be desirable that the quantizer provides a granularity that is sufficiently fine to avoid unnecessary collision. On the other hand, it also may be desired to provide sufficient error tolerance in order to maintain a desirable level of recall rate. The quantizer used in vote-count algorithm should thus be designed based on these two different and in general contradicted optimization criteria in order to deliver the optimal query quality.

[00104] In the following, a quantizer according to various embodiments will be described, which may improve the performance of the vote-count method. The quantizer may use a three-step quantization interval with an adjustable "match-zone" centered at the projected value of the feature vectors, and the vote count is incremented only when the projected value of the query vector falls into this "match-zone". The width, or "range" of the match-zone is determined as a solution of a constraint optimization problem to provide the desirable trade-off between recall rate and collision rate of the final vote-count algorithm based on the statistical models of both feature vectors. According to various embodiments, this quantizer or this quantizer design may be referred to as range quantizer. Experimental results show that compared to the fixed quantizer described

above, the range quantizer better captures the statistical property of feature vectors and hence it can dramatically improve the performance of the vote-count method.

[00105] It is to be noted that the original implementation described below designed for the original vote-count algorithm is not capable of supporting the quantizer design. However, the range query feature as described above may make it possible to implement the new range quantizer based method. As will be described below, an actual implementation may extend discrete-valued range query as described above to continuous-valued. In the following, a the theoretical derivation and numerical performance evaluation will briefly be described.

[00106] Denote $S = \{\vec{v}_i\}_{i=1}^{N}$ the set of all feature vectors extracted from a media database. It may be assumed that the feature vectors $\vec{v}_i$, $i = 1,\ldots,N$ are instances of a random vector $\vec{\mathbf{v}}$, all with d dimensions. Furthermore, it may be assumed that elements $\vec{\mathbf{v}}_{(i)}$, $i = 1,\ldots,d$ of random vector $\vec{\mathbf{v}}$ are identically and independently distributed (i.i.d.) Gaussian random variables with zero mean and variance $\sigma_v \triangleq E\left[\vec{\mathbf{v}}_{(i)}^{\,2}\right]$. It may be seen that this seemingly over-restricted assumption may be realistic in many use cases where the elements of feature vectors are indeed approximately Gaussian distributed due to central limit theory (e.g., due to each feature being derived/combined from many sub-features), and the i.i.d. requirement may be easily satisfied by performing proper linear transform and scaling on the feature vector space.

[00107] It is to be noted that a query vector $\vec{q}$, which is extracted from a query media file using the same method as that for generating the feature vectors in S, can be represented as

$$\vec{q} = \vec{v} + \vec{n},$$

where $\vec{v} \in S$ is the true matching (original) feature vector of the query media clip, and $\vec{n} \triangleq \vec{q} - \vec{v}$ is the distortion or noise vector which models the deviation of the query media to its matching one due to media processing such as editing, filtering, cropping, or compression. Similarly, the noise vector may be modeled with a random vector $\vec{\mathbf{n}}$, which may be assumed to be Gaussian distributed with zero mean and a diagonal covariance matrix $\sigma_n^2\mathbf{I}$ where $\mathbf{I}$ is the identity matrix.

[00108] In vote-count algorithm implementing range quantizer, the vote count of a feature vector may be increased by one only when it is sufficiently close to the query vector after linear projection, or equivalently, when the projected value of the query vector falls into a certain range of the projected value of feature vector, on a given projection. More specifically, denoting by r the radius of the range, the vote-count algorithm with range quantizer can be expressed as follows:

$$c_i \leftarrow \begin{cases} c_i + 1, & |\vec{a}.\vec{q} - \vec{a}.\vec{v}_i| < r, \\ c_i, & \text{otherwise,} \end{cases} \qquad (1)$$

where $\vec{a} \in R^d$ may be the random projection vector and c_i may be the vote count corresponding to feature vector $\vec{v}_i \in S$, and $\vec{a}.\vec{q}$ represents the dot-product (i.e., linear projection) of $\vec{a}$ and $\vec{q}$.

[00109] It can be seen the probability of incrementing the vote count for feature vector $\vec{v}_i$, after linear projection $\vec{a} \in R^d$, depends on whether it is the true matching vector of the query vector $\vec{v}$, which is given by:

$$\Pr(|\vec{a}.\vec{q} - \vec{a}.\vec{v}_i| < r) = \begin{cases} p_1, & \vec{v} = \vec{v}_i \\ p_2, & \text{otherwise} \end{cases}$$

[00110] Here, $p_1 \triangleq \Pr(|\vec{a}.\vec{n}| < r)$ may be the probability of incrementing the vote count corresponding to feature vector $\vec{v}_i$ if $\vec{v} = \vec{v}_i$, i.e., the per-projection success rate

under linear projection $\vec{a}$; and $p_2 \triangleq \Pr(|\vec{a}.(\vec{n}+\vec{v}-\vec{v}_i| < r)$ is the probability of incrementing vote count while $\vec{v}_i$ is not the matching vector for $\vec{v}$, i.e., the per-projection collision rate under linear projection $\vec{a}$.

[00111] FIG. 16 shows an illustration 1600 illustrating an effect of radius r on per-projection success rate (p_1) vs. collision rate (p_2) for the range quantizer. The range 1604 may be centered at $\vec{a}.\vec{v}$ with width $2r$, but it is also possible to make its center off $\vec{a}.\vec{v}$ for more flexibility in optimization while at the cost of much more complex optimization procedure. Curve 1602 is the probability distribution of what a reference projected value would lie at (when considering all reference vectors), after being projected with projection $\vec{a}$. Curve 1604 is the probability distribution of what the query $\vec{q}$ after projecting with $\vec{a}$ would lie at (when considering all possible noise/distortions in $\vec{q}$).

[00112] Since both the distortion vector $\vec{n}$ and feature vector $\vec{v}$ are Gaussian, their linear projections are Gaussian distributed as well. Without losing generality, assuming $\|\vec{a}\|_2 = 1$, and $\vec{\mathrm{v}}$ and $\vec{\mathbf{n}}$ are statistically independent, it can be shown that p_1 and p_2 is given, respectively, by

$$p_1 = \int_{\vec{a}.\vec{v}-r}^{\vec{a}.\vec{v}+r} pdf_{\vec{a}.\vec{q}}(x)dx = 2F\left(\frac{r}{\sigma_n}\right) - 1,$$

and

$$p_2 = \int_{\vec{a}.\vec{v}-r}^{\vec{a}.\vec{v}+r} pdf_{\vec{a}.\vec{v}_i}(x)dx = 2F\left(\frac{r}{\sqrt{2\sigma_v^2+\sigma_n^2}}\right) - 1,$$

where the integrations above are the more general formulas without Gaussian assumption and can be illustrated from FIG. 16; $F(x)$ is the cumulative distribution function of standard Gaussian distribution as follows:

$$F(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-\frac{t^2}{2}} dt.$$

[00113] If the vote-count method has been repeated for L random projection vectors, $\vec{a}_i, i = 1,\ldots,L$, the final values of vote count $c_i, i = 1,\ldots,N$ are, in fact, results of L Bernoulli trials with probability of a success p_1 if the feature vector under test is a true match, or p_2 if it is not. The probability mass function of c_i, $i = 1,\ldots,N$ is thus binominal, which is given by:

$$p_c(c) \triangleq \Pr(c_i = c) = \begin{cases} p_{c,1}(c), & \vec{v}_i = \vec{v} \\ p_{c,2}(c), & \text{otherwise} \end{cases}$$

where

$$p_{c,1}(c) = \binom{L}{c} p_1^c (1-p_1)^{L-c},$$

and,

$$p_{c,2}(c) = \binom{L}{c} p_2^c (1-p_2)^{L-c},$$

[00114] The recall and collision rates of vote-count algorithm may be calculated based on range quantizer, which may be given as follows:

$$p_{final} = p_{recall} = \sum_{c=l}^{L} p_{c,1}(c), \text{ and}$$

$$p_{collision} = \sum_{c=l}^{L} p_{c,2}(c).$$

[00115] Here $l \leq L$ is the threshold of vote-count algorithm (i.e., the T^*L). In a practical application it is certainly desirable to have a query algorithm such that the recall rate is maximized and the collision rate is minimized. One should see that these two requirements are, naturally, contradicted to each other since both p_{recall} and $p_{collision}$ are both monotonically increasing function with respect to the quantization range r. As a result, certain trade-off has to be made when r is selected. One trade-off that is

particularly useful for real-life applications is to find the quantization range r such that the collision rate is minimized while maintaining a certainly level of recall rate. Note that for a database with N feature vectors, $p_{collision} \times N$ gives the expected collision count on one vote-count ANN search. This leads to the following optimization problem (given L):

$$\text{P1}: \min_{r}(p_{collision}), \text{ s.t. } p_{recall} \geq \alpha,$$

where $0 < \alpha < 1$ is the minimum acceptable recall rate. It is, however, possible to optimize the quantization range r based on a different while equally important problem in real-life applications, which is maximizing the recall rate under the constraint of certain collision rate level, i.e.,

$$\text{P2}: \max_{r}(p_{recall}), \text{ s.t. } p_{collision} \leq \beta,$$

where $0 < \beta < 1$ is the maximum tolerable collision rate. According to various embodiments, devices and methods may be provided for optimizing problem P1 while the devices and methods according to various embodiments may similarly be applied to optimization problem P2 as well.

[00116] In the following, a solution of problem P1 will be described, which in its native format is not trivial. Note that for sufficiently large L, the binominal distribution is well approximated by Gaussian distribution with mean Lp and variance $Lp(1-p)$ where p is the probability of success. Therefore, for sufficiently large L, both the recall rate and collision rate can be approximated by:

$$p_{recall} \approx 1 - F\left(\frac{l - Lp_1}{\sqrt{Lp_1(1-p_1)}}\right),$$

and

$$p_{collision} \approx 1 - F\left(\frac{l - Lp_2}{\sqrt{Lp_2(1-p_2)}}\right),$$

where again l is the threshold for vote count (i.e., T^*L). Following this approximation, P1 can be rewritten as follows:

$$\text{P3: } \min_r \left[1 - F\left(\frac{l - Lp_2}{\sqrt{Lp_2(1-p_2)}}\right)\right], \text{ s.t. } 1 - F\left(\frac{l - Lp_1}{\sqrt{Lp_1(1-p_1)}}\right) \geq \alpha.$$

[00117] Since both functions are monotonically increasing function with respect to the optimization variable r, the inequality condition is active which can be changed to equality condition. Denote l^* as the solution of l that achieves the equality constraint, i.e.,

$$1 - F\left(\frac{l^* - Lp_1}{\sqrt{Lp_1(1-p_1)}}\right) = \alpha,$$

or

$$l^* = F^{-1}(1-\alpha)\sqrt{Lp_1(1-p_1)} + Lp_1.$$

[00118] Replace the above result into P3 yield the following unconstraint optimization problem

$$P4: \min_r p^*_{collision}$$

where

$$p^*_{collision} \triangleq 1 - F\left(\frac{l^* - Lp_2}{\sqrt{Lp_2(1-p_2)}}\right)$$

$$= 1 - F\left(\frac{F^{-1}(1-\alpha)\sqrt{p_1(1-p_1)} + (p_1 - p_2)\sqrt{L}}{\sqrt{p_2(1-p_2)}}\right),$$

or equivalently,

$$P4^*: \max_r g$$

where

$$g \triangleq \frac{F^{-1}(1-\alpha)\sqrt{p_1(1-p_1)} + (p_1 - p_2)\sqrt{L}}{\sqrt{p_2(1-p_2)}}$$

since $F(x)$ is a monotonically increasing function. P1 through P4* assume a given, fixed L. Since L is the number of the projections and controls the amount of computation required for projections (for both reference and query vectors) and as will be described in more detail below is also related to the system cost of implementation, yet another optimization formulation could be to minimize L while ensuring a minimum recall and maximum collision rate limit, as in the problem P5 below, where p_{recall} and $p_{collision}$ may be defined using the binomial distribution or Gaussian approximation just described:

$$\text{P5: } \min_r L, \text{ s.t. } p_{recall} \geq \alpha,\ p_{collision} \leq \beta$$

[00119] Furthermore, if a given cost factor $\eta > 0$ can be associated with collision rate, then it is possible to optimize the combination of recall and collision (given L), as the following problem P6:

$$\text{P6: } \min_r (1 - p_{recall} + \eta \cdot p_{collision})$$

[00120] While P5 minimizes system cost, an interesting goal may be to maximize search throughput per dollar, i.e., the number of searches per sec divided by the system cost while still meeting a given recall and collision rate limit. Since the collision count affects vote-count ANN search time, so if its impact on search time is correctly factored in (e.g., assuming a certain HDD or Solid State Disk/SSD seek time during full distance check), assuming database size N (number of feature vectors) is known, the collision rate limit may not even be explicitly necessary, since if collision rate (and hence collision count) were too high, search throughput will decrease and result in potentially lower the throughput to system cost ratio. This may be formulated as problem P7 as follows:

$$\text{P7: } \max_{r} \frac{searches\ per\ sec}{system\ cost}, \text{ s.t. } p_{recall} \geq \alpha, \text{ optionally } p_{collision} \leq \beta$$

[00121] Furthermore, in addition to keeping collision rate low, the testing of m projections at a time (like described above) may be combined with range quantizer to both save power and achieve high recall at low collision. This may be formulated as problem P8 as follows. Here $f_{overall}()$ may be any function that gives a reasonable overall score based on the expected peak power (which may be defined as $p_2{}^m \times P_{max}$), the system cost and the search throughput, where the score is higher if expected peak power is lower, and/or if system cost is lower, and/or if search throughput is faster:

$$\text{P8: } \max_{r} f_{overall}(expected\ peak\ power, system\ cost, searches\ per\ sec), \text{ s.t. } p_{recall} \geq \alpha,\ p_{collision} \leq \beta$$

[00122] In the following, performance evaluation will be described.

[00123] FIG. 17 shows an illustration 1700 illustrating $p^*_{collision}$ as a function of range r. The minimum collision rate is achieved at $r = 1.13$. ($\sigma_v^2 = 1$, $\sigma_n^2 = 0.5$, $\alpha = 0.98$, $L = 50$).

[00124] FIG. 18 shows an illustration 1800 illustrating a performance comparison of devices and methods according to various embodiments vs. fixed binary quantization when varying recall rate ($\sigma_v^2 = 1$,$\sigma_n^2 = 0.5$, $L = 50$).

[00125] FIG. 17 shows how a choice of r can greatly affect collision rate. When held at a recall of 0.98 and L of 50, assuming a signal-to-noise ratio (SNR) of 1:0.5, or ~3dB, the range quantizer can have $p^*_{collision}$ (calculated using Gaussian approximation of binomial distribution) as low as 3.3×10^{-6}, while FIG. 18 shows a conventional 1-bit quantizer achieves $p^*_{collision}$ of approximately 2.8×10^{-4}, almost 100x higher than range

quantizer. This 100x difference is also largely maintained when varying recall from 0.8 to 0.98, as shown in FIG. 18.

[00126] In the simulation for FIG. 17 and FIG. 18, under commonly used 1-bit quantization, one can derive that $p_1 \approx 0.804$ and $p_2 = 0.5$, that is, the BER is $1-p_1 \approx 0.196 \approx 0.2$ under ~3dB SNR condition. If the range quantizer is used instead and $r = 1.7$, one can derive that $p_1 \approx 0.9832$ and $p_2 \approx 0.7149$, and if one chooses m=32 (testing 32 projections at a time) and L' = 12 (for a total of L = L'×m = 384 projections), using binomial distribution function (because L' is too small and p_1, p_2 are too biased toward 1 for the Gaussian approximation) one can derive that when T*L' = 4, a recall ≈ 0.98 and $p_{collision} \approx 7 \times 10^{-16}$ can be achieved while expected peak power for a one-trillion vector database is only $p_2^m \times P_{max} \approx 0.7149^{12} \times 3MW \approx 65W$. This is more than 4 orders of magnitude in power saving than the design in original vote-count if both are implemented on NAND Flash, and is also about more than 3 orders of magnitude in power saving compared to other ANN implementations such as LSH with weak-bits. Its search time on projection comparison and vote counter update is roughly $L' \times \tau \approx 600\mu s$ (assuming $\tau = 50\mu s$, a fairly feasible value), if another 100μs is used to access (for example, using a fast conventional Flash SSD) the candidate feature vector for full Euclidean distance check, then about 700μs is sufficient to perform one ANN search under a very noisy condition of ~3dB SNR on the query vector. This is just one out of many possible configurations that can provide such high performance. Other choices of r, m, L', T*L' may also provide very high performance under adverse SNR conditions.

[00127] In the following, a design for devices according to various embodiments (for example hardware design according to various embodiments) will be described.

[00128] In the following, parallel update of vote counters according to various embodiments will be described.

[00129] With N reference vectors in a database, performing the equality comparison of quantized projected values between query and reference along with subsequent counter update would take O(N) memory cycles in software with single-threaded implementation. If N = 1 billion, and memory cycle is 1ns, this would take on the order of 10^9*1ns = 1 sec per projection, making the search too slow. Commonly used hardware implementation may perform the equality comparison and counter update very fast, using Dynamic Random Access Memory (DRAM) or any storage device with a matrix layout (e.g., Flash memory).

[00130] FIG. 19 shows an illustration 1900 of an implementation, with each reference vector corresponding to one column in a DRAM matrix array. The entire column of cells stores the quantized values from L projections, and at the end of each column an equality comparator and vote counter are added to provide parallel comparisons and counter update. In the illustration 1900, a basic vote-count hardware architecture for fast ANN, in DRAM, is shown.

[00131] Using binary/1-bit quantization as an example, in FIG. 19, at i-th projection, the entire i-th row of the memory cells in the storage array are read out and compared against the query vector's corresponding quantized projected value, and a column-wise vote counter (initialized to be zero upon a search) is incremented if and only if the comparison succeeds (i.e., the quantized values are the same between query and reference). After examining L projections, all vote counters' values are compared against a threshold T*L, and a column (and its corresponding reference vector) is selected as a

candidate if and only if its vote counter's value exceeds T*L. An appropriately chosen T and L may achieve high recall and high precision, even in large-scale, difficult data sets. If the database has 1 billion vectors, the storage device then needs to have 1 billion columns, which may be implemented by having multiple storage arrays of a fixed number of columns, with total number of columns equal to 1 billion (e.g., 100,000 arrays with 10,000 columns each), and activating the comparison on all involved storage arrays in parallel. A key characteristic of most matrix-layout storage arrays is that whenever a row is accessed for read, the stored data in the entire row in that array becomes available. Vote-count exploits this characteristic to update all the vote counters in parallel, instead of having to increment them one by one, for example such as in software.

[00132] The basic vote-count implementation, besides its weakness on highly correlated dataset, has a drawback in power consumption. This is because each update of vote counters requires data readout for all storage cells in the row being examined. This may consume a huge amount of power. In comparison, when a CPU (central processing unit) reads data from DRAM, usually only one or very few DRAM storage arrays are actively read out simultaneously, whereas in vote-count, all DRAM storage arrays on all vote-count DRAM chips (if implemented using DRAM) holding the database of quantized projected values must be actively read out and compared simultaneously.

[00133] In contrast to DRAM, Flash memories also have a matrix layout, but are non-volatile and do not require power to retain its stored data. NAND Flash, the kind used in digital cameras, USB (universal serial bus) thumb drives, is the cheapest type of Flash memories, and may serve as a base form from which power-efficient designs may be

applied. Note here the term cells and transistors (including the floating gate transistors to be described) may be used interchangeably, unless explicitly noted.

[00134] In the following, hardware designs according to various embodiments will be described that can efficiently achieve various advanced search features described above, including multiple projection comparison, range query, and the combination of the two. The said design can also significantly reduce power consumption.

[00135] In the following, basics of Flash memories will be described.

[00136] FIG. 20 shows an illustration 2000 of a working principle of floating gate transistors in Flash memories.

[00137] NAND Flash may be constructed as series of floating gate transistors (fGTs). An fGT (assuming n-channel type), in the case of single level cell (SLC), has two states, erased (encodes a "1"), or programmed (encodes a "0") Note the encoding-to-state correspondence is by convention but an opposite correspondence may also be used. In erased state, the fGT has a low threshold voltage (V_{th}) of say 0.7V, whereas its V_{th} is much higher in programmed state, say 3V, as illustrated for the working principle of fGTs in FIG. 20. The meaning of threshold voltage for fGT may be similar to that of a MOSFET: if the voltage from Control Gate to Source is above V_{th}, a conducting channel will form between Drain and Source and allow currents to flow (note that if the input voltage is below V_{th}, a very small current may still flow, but it will be almost negligible) through if $V_{DS} > 0$. In case of integrated circuit (IC) design, with concatenated transistors, the voltage from Control Gate to Bulk (which is typically connected to Ground) may be used to define V_{th} and its characteristics in lieu of the voltage from Control Gate to Source. If Bulk is not connected to Ground, the voltage from Control Gate to the Source

terminal of the bottom fGT (i.e., closest to Ground) in the series of concatenated fGTs may be used instead. In FIG. 20, the leftmost icon is the commonly used symbol denoting fGT, the middle diagram illustrates an fGT in erased state, and the rightmost diagram illustrates an fGT in programmed state. Note that for convenience of drawing, the ellipse representing the (negative) charge is placed in between the floating gate and the body electrode (the biggest blue unfilled rectangle), although in reality the charge is stored on the floating gate itself.

[00138] FIG. 21 shows an illustration 2100 of an example of a basic circuit layout of NAND Flash within one column.

[00139] In case of NAND Flash, within a column, a series (referred to as a "string" in NAND Flash terminology) of fGTs are concatenated together, with one fGT's Source connecting to the next fGT's Drain. This is illustrated in FIG. 21, where String select is a MOSFET that controls whether this series/string is selected, and GND select is another MOSFET that controls whether the string's bottom is connected to Ground. In comparison, within a column, a NOR Flash fGT's Drain is always connected to the same bit-line, and Source is always connected to the Source line. Such a series connection simplifies physical layout and reduces manufacturing complexity, thus leading to lower cost for NAND Flash in contrast to NOR Flash. It also means the fGTs will conduct only if all fGTs in the series form a conducting channel. Therefore, forming a conducting channel in one fGT is only a pre-requisite for a NAND Flash series/string to conduct, so that by using different wording it may be distinguished between "forming/causing a conduct channel, the fGT conducts" and "the series conducts or draws current". Because of this property, NAND Flash may read its data by probing all wordlines (a wordline is a

row input line) in the series with high voltage that is greater than V_{th} of the erased state (such as 3.5V which is greater than V_{th} of 3V illustrated in right part of FIG. 20, so that they always conduct) except for the row it intends to read. For the to-be-read row, the probing voltage is in between V_{th} of erased state and V_{th} of programmed state, such as 1V which is greater than V_{th} of 0.7V illustrated in right part of FIG. 20. These two voltages are denoted here symbolically as hi and mid, respectively. If the fGT is programmed, it won't conduct, else it will conduct. The detection of conduction and/or current in the series allows the detection of fGT state, either by sensing the current, or by sensing the voltage of the bit-line, hence the readout of stored data (0 or 1). This is illustrated in FIG. 21, where the 2^{nd} wordline input is at mid and all other wordlines at hi, and thus allows the readout of stored bit value in the fGT on the 2^{nd} row. A column has one or more series/strings whose ends (outer side of GND select MOSFET) are connected in parallel instead of in series. A typical string may have 8-128 fGT cells. A "page" is an entire row of data in a NAND Flash array, and a "block" is all pages of data from all rows within a string. In NAND Flash, typically the smallest unit of programming is 1 page, and smallest unit of erase is 1 block, and after a page is programmed, it has to be erased (along with the entire block, which is slow) before it can be programmed again.

[00140] The series nature of the NAND Flash circuit provides a possibility of power efficiency. If the circuits can be modified such that a column (which may correspond to a feature vector) conducts only if the whole query probing pattern matches with what's stored on the fGTs on the same rows as the query probe, then (non-negligible) power is consumed only if the pattern matches on that column, instead of consuming power for every column. However, a trivial design where probing voltages are equal to voltages

required to just cause an fGT having a particular test bit value from the query pattern to conduct, does not work. For example, to test for the 5-bit pattern == 10111 (where 1 means erased), if we probe with (mid hi mid mid mid), the series will conduct if the stored bits are 11111 or 10111. The reason is because fGT implements >= logic, whereas == logic is desired during the above pattern match. Here ">= logic" may mean whenever input voltage (to G-terminal) > (larger than) some other voltage (e.g. Vth), the cell is conductive (between D and S terminal). "== logic" may mean that the cell behavior is desired such that it is conductive only if its input voltage (to G-terminal) is == (equal to) some predefined voltage (which may correspond to its stored value).

[00141] In the following, an interlocked design of floating gate transistors as building block according to various embodiments will be described.

[00142] FIG. 22A shows an illustration 2200 of different charging states. A (mid,hi) probe won't make a "0" conduct.

[00143] FIG. 22B shows an illustration 2202 of an interlocked fGT pair representation enabling power-savings, assuming (hi,hi) is not allowed. (mid,mid) and (hi,hi) probes may not allowed by default. As illustrated in FIG 22A and 22B, in a Flash cell, if it (i.e., its floating gate) stores no charge, a V_G >= mid is sufficient to make cell conduct (from D to S); but if it stores full (negative) charge, a V_G >= hi is required to make cell conduct. Therefore, it may be deduced that a (mid,hi) input voltage pair will make an encoded "1" in FIG 22B conduct, but not make an encoded "0" conduct, and vice versa.

[00144] A hardware design according to various embodiments may implement == logic from fGT's >= logic, by using two in-series fGTs per represented bit, with interlocked probing voltages between these two fGTs, as illustrated in FIG. 22B. To

represent a 1, the top fGT should be in erased state and bottom fGT in programmed state; to test == 1, the top and bottom probing voltages should be (mid,hi), respectively. Similarly, to represent a 0, the top fGT should be in programmed state and bottom fGT in erased state; to test == 0, the voltages should be (hi,mid), respectively. The two other possible input probe combinations are not allowed here (but may be allowed in special cases to be described later). A close inspection will reveal that, a probe of (mid,hi) will cause the series to conduct only if the two fGTs represents a 1, because the top mid is not high enough voltage to make top fGT (programmed) on the right of FIG. 22B conduct, even though bottom hi is high enough, and both need to conduct for the series to conduct. Similarly, a probe of (hi,mid) will cause the series to conduct only if the two fGTs represents a 0. The reason this design works is because the stored data is conformed to an interlocked manner: one component of the probe pattern is sufficient to cause conduction when matched, but is also guaranteed to be insufficient for conduction when not matched. Alternatively, the representations in FIG. 22B may be swapped, with the left diagram encoding a 0 and right diagram encoding a 1.

[00145] If a series/string of 16 (i.e., 8 pairs of) fGT cells is creating using such interlocked design, assuming the database is relatively random, then on average approximately only $p_2^m = 1/2^8 = 1/256$ columns may conduct and consume (non-negligible) power, hence peak power is reduced to approximately 1/256 of P_{max} of the design in basic vote-count. In case the dataset is strongly biased that many more than p_2^m ratio of columns conduct, if the goal is to limit peak power, a current limiter may be used to cap peak power at a certain level, with the trade off that the match may take somewhat longer than τ to complete due to the weaker than nominal (e.g., 1μA) current per

conducted column. The vote counter is incremented if and only if the probed series conduct. Instead of a maximum count of L, with a 8-pair series the maximum count would be L'=L/8. This may be an example of multiple projection comparison, with m=8, and the nature of NAND Flash probing operation makes it such that the random access cycle of a row is limited to time τ whether in conventional read mode or the interlocked probing mode (which we will later refer to as the query mode). Therefore, overall search time can be roughly m times as fast. If 32 cells are in a series, then peak power may be reduced to $1/2^{16} = 1/65536$. There is, however, a trade-off in recall and precision, because if BER is high, then having too many in-series pairs of interlocked fGTs will cause a lower match probability per series, and this may negatively affect precision (i.e., higher collision count) if we maintain the same number of projections and the same recall, because a smaller vote-count threshold factor T (and correspondingly T*L' instead of T*L) may be needed to maintain same recall, at the cost of higher collision count (hence lower precision). It may be possible to maintain the same collision count and same recall by increasing the number of projections, albeit at the cost of using more fGTs. Further below, a design according to various embodiments will be described that can improve per-series match probability.

[00146] The above interlocked design uses 2 fGTs per represented bit (as opposed to 1 fGT in design of basic vote-count), but it allows multiple projection comparison (as described above) within time τ, thus reduces search time, and it also allows significant power savings. It does not require any modification in the physical design of fGTs, nor require any modification in read/write circuit of NAND Flash. Conventional writing procedures may be used as-is, as long as the data to-be-written conform to the interlocked

design representation, and this conformance can be implemented at the software level (which is easier to implement) and need not be at the NAND Flash circuit level. The interlocked design only requires some modification to the probing circuits on each wordline, in addition to the per-column vote counter circuitry in basic vote-count. Such modification creates a new mode other than read and write: a query mode. In this query mode, the NAND Flash circuit does not necessarily know the data stored in the fGT cells, and it only knows whether the stored data match the query input pattern of probing voltages. If the pattern exactly matches, the circuit knows what is stored, but if no match, the circuit knows it didn't match and wouldn't necessarily know what exactly is stored.

[00147] CMOS may have some notion of interlocking in its design, but CMOS ensures that in any stable state the MOSFETs are interlocked in a way that they never conduct current. In contrast, the above interlocked design in this disclosure will conduct current when the query input pattern matches the stored data pattern, and is therefore a completely different design and has completely different functionality.

[00148] In the following, a wild-card/don't care/always-match capability according to various embodiments will be described.

[00149] The length of the series may be chosen to correspond exactly to the length of query input pattern. For example, the series may be 16 cells long when using 8-bit patterns (which would generate 16 probing voltages on 16 wordlines). However, such choice may be inflexible because the length of the series is generally determined at NAND Flash manufacturing time, while the length of query input pattern may vary at run-time. A more flexible way is to create a series whose length (fixed at manufacturing time) is long enough (e.g., to have enough reduction in search time and/or to provide the

desired power-saving factor, and use special query input patterns that ignore irrelevant rows in a series/string. For example, the series may be 48 cells long, but an 8-bit pattern (16 rows) may be used for query. The remaining 32 rows should be ignored, and to do that, the fGTs on these remaining rows must always conduct. Therefore, these remaining rows should have probing voltages of (hi,hi) at each pair's wordlines. Such (hi,hi) combination is not allowed in previous example of the interlocked design, but may be used to ignore certain rows of data. Since (hi,hi) always causes conduction channel in the respective cell pair, it can be thought of as a "wild-card", "always-match", or "don't care" input (the 3 terms are used interchangeably), with the corresponding abstractions defined as described above.

[00150] FIG. 23 shows an illustration 2300 of such an example, where the series is 6 cells long, and query pattern is 10X, meaning the 3rd bit (that is represented in the two bottom fGT cells) should be ignored during probing. In this example the encoding convention is assumed to be the same as in FIG. 22B. In FIG. 23, an example of using wild-card/don't care capability is shown. The query pattern may be 10X where X may mean "don't care". FIG. 23 may correspond to the query side, wherein FIG. 24 (like will be described in more detail below) may correspond to the reference side.

[00151] Although the examples so far appear to hint that a pair of interlocked fGTs must be connected together back-to-back in-series, they don't have to. They do need to be connected in-series (thus within the same series/string), but need not be back-to-back, and the upper fGT in the pair need not even be in the upper position physically. The two fGTs in a pair may be far apart in physical layout in the series/string, but as long as the query circuit knows on which two rows they are located and the layout is identical across

columns in the same Flash array, it will be fine. For example, if fGT f0, f1, f2, f3, f4, f5 are 3 pairs, and actually series lay out could be f0, f2, f4, f1, f3, f5. This layout decision may be performed at manufacturing time, but may also be, and more likely, at run-time when storing the data.

[00152] Similarly, for example with 8-bit match pattern, the 8 pairs of fGTs need not be consecutive in the physical layout: these 8 pairs can be allocated on any 16 rows within a series as long as the query circuit knows on which 16 rows and the layout is identical across columns in the same Flash array, and the series may be longer than 16 cells as noted before.

[00153] In the following, use of wild-card for weak-bit will be described.

[00154] Testing for a multi-bit pattern with an interlocked series design may reduce search time and may save power, but may reduce match probability if the pattern is long and/or BER is high. To mitigate that problem, weak bits may be used and the "don't care"/"always-match"/"wild-card" pattern may be used as an efficient alternative to weak-bit enumeration. It is expected that, certain projections of a vector are close to quantization boundaries and therefore vulnerable to bit errors, because even a small distortion may cause the quantization to differ between query and original reference. Weak-bit enumeration has been proposed in both audio/video fingerprinting and LSH to reduce the impact of high BER due to quantization error in such cases. First, the search algorithm identifies which projections and bit positions (conventionally only at the query side for efficiency reasons) are weak and vulnerable, possibly using some distance threshold to quantization boundaries, and then enumerate some or all variations at these bits, with each variation generating a particular query probe. If there are for example 12

weak bits in the query codeword, then 2^{12} variations may be probed against the hash indexing table. If the number of weak bits is high, it may become very slow to enumerate all these variations.

[00155] The interlocked design enables an efficient alternative to enumerating weak bits, because it can have a "don't care"/"always-match"/"wild-card" input pattern. Since enumerating a weak bit implies that the said weak bit becomes a don't care bit, consequently, for all weak bits in the query codeword, we can map it to a don't care pattern, such as a (hi,hi) probe pattern just like in FIG. 23, and this may be highly attractive when the match pattern is very long and the number of weak bits fairly high. For example, if the query codeword is 64-bit long and on average 30 weak bits are expected, then a 128-cell series string may be created on NAND Flash that will support 64-bit look-up with arbitrary number of weak bits in time τ (which is one NAND Flash row random access cycle, typically 5-50μs). In comparison, if 30 weak bits are enumerated, even if each enumeration takes only 1ns, it will take a total of $2^{30}\times 1\text{ns} \approx 1\text{sec}$ to complete all enumeration. With the approach of FIG. 23, the number of weak bits does not affect the vote-count portion of search time, because there is no enumeration involved. It may however, generate more unrelated candidate matches if too many weak bits are detected and used: for example if a 64-bit codeword has 54 weak bits, then it will have only 64-54=10 effective differentiating bits, and if the database has 1 billion vectors, this would generate on average $10^9/2^{10} \approx 10^6$ candidate matches. Since the candidate matches should be checked for full (e.g., Euclidean) distance, it may significantly slow down the checking time. Of course, several shorter series may be used instead of performing a weak-bit based look-up on the entire codeword over a long series.

[00156] It is expected that weak-bit enumeration will also improve remaining BER, because the weak bits are expected to have high BER, and by ignoring those bits, the remaining (non-weak) bits are expected to have somewhat lower BER, since the average BER is based on averaging BER from both weak bits and non weak bits. If initially BER is 0.2 and the query pattern needs to be 8 bits or shorter to retain performance (i.e., low enough collision count and/or small enough number of projections, at a given recall p_{final}), then using interlocked design with weak-bit, a lower remaining BER (of say 0.1) may allow a longer query pattern (say 24 bits with 10 weak bits, resulting in 24-10=14 effective differentiating bits) at the same performance, and this would further reduce search time because m=24 instead of m=8, and would allow additional power savings (e.g., reduced to approximately $1/2^{14}$ vs. $1/2^{8}$ for 1-bit quantization and $p_2 = 0.5$).

[00157] In the following, reference-side wild-card according to various embodiments will be described.

[00158] Commonly weak-bit enumeration or its variant may be performed at the query side, i.e., the algorithm determines which bits are weak in the query codeword and enumerate those. It is usually impractical to enumerate weak bits on the reference codewords, because the weak bit positions may differ for each reference codeword, and there is no known efficient data structure that can index all these codewords in a weak-bit-invariant manner, and there is no simple way to enumerate weak bits for all the reference codewords in an effective manner. If there are 1 billion vectors in a database, and 10 weak bits per vector, then $2^{10} \times 10^{9} \approx 10^{12}$ reference-side enumerations are needed, much more than the $2^{10} \approx 10^{3}$ query-side enumerations. Alternatively, bitwise XOR operation may be used to verify whether the reference-side pattern matches the query

pattern, instead of enumerations, so that 10^9 XOR instructions are needed, which is still a lot more than the $2^{10} \approx 10^3$ query-side enumerations.

[00159] FIG. 24 shows an illustration 2400 of an example of reference-side wild-card. A reference vector may generate a wild-card pattern 10X represented in fGT states, and a query vector may generate non-wild-card pattern 101. FIG. 23 (as described above) may correspond to the query side, wherein FIG. 24 may correspond to the reference side.

[00160] The interlocked design allows wild-card at the reference side, an abstraction as described above. If a reference vector's codeword is found to have weak bits during database build time, a (erased,erased) combination may be stored in the fGT pair for each such reference-side weak bit, as illustrated in FIG. 24. In such case, the pair is technically no longer trivially "interlocked", instead it will match any input pattern. There may be advantages in reference-side wild-card, because it may be better to conceptually draw a radius R from the original audio clip's feature vector(s) and hoping the distorted version is within that radius and it should have low collision count, assuming different reference vectors are at least radius R apart from each other. In contrast, although drawing a radius R from a potentially distorted version may also find the original version, more unrelated reference vectors may fall within that radius.

[00161] Once the reference-side wild-cards are generated and written to NAND Flash, the query side may generate query input patterns that do not use wild-cards, and the system may still be expected to identify the right matches with reasonably high accuracy. Since reference-side wild-cards are generated at database build time, it may save computational time in determining query-side wild-cards at query run-time. Alternatively, the query side may also generate its own weak-bits in the query input pattern, to further

improve the match success probability (but at the cost of potentially more unrelated matches), and in such cases, the query side's number of weak bits and its criteria for choosing weak bits need not be the same as the reference side's. For example, reference side may generate approximately 10 weakest bits, and query side may generate approximately 6 weakest bits.

[00162] In the following, a generalization to m-bit fGT cells (for example multi-level cells; MLCs) according to various embodiments will be described.

[00163] In recent years, multi-level cells (MLCs) are becoming more popular in NAND Flash. In comparison to single-level cell (SLC) which stores 1-bit per cell, MLC can store 2 or more bits per cell. MLC works by differentiating between the amount of charges stored in the fGT. Instead of 1 programmed state, in case of 2 bits per cell, the fGT has 3 programmed states, each in progressively larger amount of electron charges. Together with the erased state, there are 4 states and it is capable of storing 2 bits per cell. The nominal amount of charges between successive states are chosen to have sufficient difference (in charges and in V_{th}) to tell the exact state. MLCs are cheaper per bit than SLC, but are less reliable and wear out more quickly due to a smaller number of allowable erase cycles. Instead of probing with different input voltages at the wordline of interest and detecting status of conduction to infer and readout the store data in a cell, which would take several row access cycles, in some design one input voltage is used at the wordline of interest for reading data, and the cell current is compared to currents of several pre-calibrated fGT cells under the same input voltage, and the comparison results are used to infer and readout the stored data, in one row access cycle.

[00164] For any m-bit MLC in general, the previously described interlocked design can be extended accordingly and still keeping a 100% overhead (2 bits to represent 1 bit). Let's denote an m-bit MLC's set of states as $\{0, 1, 2, 3, \ldots, 2^m-1\}$, where a number i in the state set denotes (in monotonic but not necessarily in linear scale) an input voltage required to cause a conduction channel for an MLC with state 0 to i. That is, a cell with a state of 3 require an input voltage of symbol "3" to conduct, and the actual voltage value of "3" should be higher than the actual voltage value of "2", which in turn should be higher than the actual voltage value of "1", and so on. A larger state number i also corresponds to a larger amount of charges stored in the MLC. Previously for SLC we used notation like (mid,hi) to denote query input voltages and (erased,programmed) etc. to denote cell states. From here on for MLC we use the same state number i to denote both the query input voltage and the corresponding cell state. Formally defined, a cell state number i means the cell's threshold voltage is $V_{th}(i)$, and a query state number i means it will be converted to a voltage $V_G = f(i)$ such that $V_G \geq V_{th}(i)$ and if $i < 2^m-1$ also satisfying $V_G < V_{th}(i+1)$. For n-channel fGTs, f(i) is a monotonic increasing function defined over valid values of i. If there is variability or noise present in the amount of charges stored on the floating gate for storing state i, $V_{th}(i)$ is defined as the minimum (for n-channel fGTs) threshold voltage that can guarantee with a specified probability the conductable (or conductible or conductive) state of the said cell under the considered variability and/or noise conditions. f(i) and/or $V_{th}(i)$ may be different depending on how many rows of cells the query wordline is away from Ground, and/or depending on the query pattern, in order to compensate for effects that alter V_{th} of a cell, such as the Body effect. In other words, general mapping functions f(i) and Vth(i) may be provided. This

may enable more robust pattern matching at the electrical level. Then a pair of such MLC fGTs may use the following encoding, with the notation (top, bottom) :

$$(0, 2^m\text{-}1)\ (1, 2^m\text{-}2)\ (2, 2^m\text{-}3)\ \ldots\ (i, 2^m\text{-}i\text{-}1)\ \ldots\ (2^m\text{-}2, 1)\ (2^m\text{-}1, 0).$$

[00165] The above encoding may encode one of 2^m values using two m-bit MLC fGTs, for example with the above i-th state to represent value i (or use any mapping that is desired), and to test for == i, the query input pattern need to be $(i, 2^m\text{-}i\text{-}1)$. If such input pattern is used on a different state j, if $j < i$, then the bottom voltage would not be enough to cause conduction in the bottom fGT, and if $j > i$, then the top voltage would not be enough to cause conduction in the top fGT. Therefore, the above encoding design is also interlocked, and is in fact, a generalization of the SLC case, with the exception that the definition of encoded/represented value (0 or 1) for m=1 is swapped compared to the SLC case (which means m=1) in FIG. 22B. If instead the i-th state is used to represent/encode value $2^m\text{-}i\text{-}1$, then the m-bit general case is the same as the SLC case in FIG. 22B for m=1. Alternatively, we can still let state i denote encoded/represented value i, but change the definition of state i to conform to the convention used in NAND Flash, where a state 0 means having the most charge instead of the least charge. Then, provided that $V_{th}(i)$ decreases as state number i increases, and query state number i is converted to a voltage $V_G = f(i)$ such that $V_G \geq V_{th}(i)$ and if $i > 0$ also satisfying $V_G < V_{th}(i\text{-}1)$, i.e., $V_{th}(i)$ and f(i) are monotonically decreasing (as opposed to increasing when using the previous convention) functions over state number i, the above encoding will still achieve interlocking, and the m-bit general case is the same as the SLC case in FIG. 22B for m=1.

[00166] Although input voltage (required to cause a conduction channel) is used to describe the MLC state, other metrics may be used, such as the amount of charges stored

in the MLC, or the current flowing through the MLC at a fixed given input voltage. Of those, the amount of charges is not directly measurable. Note the exact input voltage for a given MLC state may take on different values across use cases, as long as these voltages for successive states follow a monotonic trend and are separated with sufficient voltage difference to tolerate noise in MLCs and/or query input voltage generation circuitry.

[00167] fGTs have mainly two types, the n-channel type (described in FIG. 20), and the p-channel type. The latter has a p-n-p junction instead of the n-p-n junction, and Flash memories built from p-channel fGTs have certain advantages in low power consumption while programming the cells (writing data). In contrast to n-channel, the control-gate voltage (V_{CG} or V_G in short) of a p-channel fGT usually has to be negative (when measured with respect to the Bulk in p-channel, which is usually V_{dd}) to cause a conduction channel (hence its threshold voltage is usually negative), and a sufficiently negative V_G will always cause a conduction channel. In essence, the n-channel fGT implements a >= test logic, and the p-channel fGT implements a <= test logic. m-bit (m >= 1) p-channel fGTs can also be used for interlocked representation, for example, by denoting its state 0 to be the state requiring the least negative V_G (i.e., has the most amount of negative charges on its floating gate) to cause a conduction channel, and state 2^m-1 requiring the most negative V_G (i.e., has the least amount of negative charges on its floating gate) to cause a conduction channel, and state i's input voltage should be more negative than state i-1's, etc. $V_{th}(i)$ is defined as the maximum (for p-channel fGTs) threshold voltage that can guarantee with a specified probability the conductable (or conductible or conductive) state of the said cell under the considered variability and/or noise conditions. For $i < 2^m-1$, $f(i) \leq V_{th}(i)$ and $f(i) > V_{th}(i+1)$. Then, the same interlocked

notation may be used for both n-channel and p-channel fGTs as-is and it will have the semantic meaning (including Table 2 as will be described in more detail below) for both types of fGTs. Alternatively, state 0 can be defined as the state requiring the most negative V_G to cause a conduction channel, and state 2^m-1 requiring the least negative V_G to cause a conduction channel, and state i's input voltage should be less negative than state i-1's, etc., which is the same as the NAND Flash state naming convention for p-channel fGTs, but the same interlocked notation will have an opposite meaning, where >= test becomes a <= test, etc., so the interlocked notation will need to be adjusted accordingly to achieve the desired test semantics. For example, an interlocked pair notation (i, 2^m-i-1) should be changed to (2^m-i-1, i) for both reference and query side and then the same semantics can be achieved.

[00168] Of course, if the fGT can support m-bit per cell, the user may still choose to use fewer than m bits per cell for implementation of vote-count ANN search, albeit at lower efficiency. For example, 2 bits may be used in a 3-bit capable fGT MLC and a pair of such MLCs may be used to represent 2 bits of data (either using it as a 2-bit MLC, or as a 3-bit MLC but ignores its least significant bit) in an interlocked format.

[00169] Instead of the above interlocked representation (which uses 2 bits of storage to represent 1 bit of data), other interlocked representation (albeit potentially less efficient) may be used. For example, to describe k states, k SLCs may be used in a series/string with a unary representation, where SLC i (i = 0, 1, 2, ..., k-1) is in programmed state if and only if the represented state is i (note here a represented state is different from a cell state). So for a represented state i, all other SLCs are in erased state, and only SLC i is in programmed state. To test whether the represented state is i, the query input voltage for

all other SLCs should be mid, and only SLC i's input voltage should be hi. If the represented state is not i, the input voltage pattern will not be able to cause conduction in the series. Such unary representation is however, less efficient, since it uses k bits of storage to represent ceil($\log_2 k$) bits of data. If p-channel SLCs are used, the same query pattern may be used, provided mid < hi in terms of voltage including the +/- sign, and that mid and hi voltages should be chosen such that it should cause the conductible state in an erased and in a programmed cell, respectively. Such definitions of mid and hi is the same as n-channel in a verbatim manner, but with the exception that in n-channel, a voltage of hi or above will guarantee the conductible state in the probed cell, but in p-channel, a voltage of mid or below will guarantee the conductible state in the probed cell.

[00170] Alternatively, for a unary representation, SLC i is in erased state if and only if the represented state is i. So for a represented state i, all other SLCs are in programmed state, and only SLC i is in erased state. To test whether the represented state is i, the query input voltage for all other SLCs should be hi, and only SLC i's input voltage should be mid. If the represented state is not i, the input voltage pattern will not be able to cause conduction in the series. The minor difference between the first and the alternative unary representation is that, if the data stored on the SLCs do not always conform to the supposed representation, then when there is conduction for the first unary representation, the stored data on that particular conducting column must be conforming to supposed representation, because there is only one stored pattern that could cause conduction. Whereas with the alternative unary representation, a conducting column may in fact have one of several patterns, since there are multiple patterns that could cause conduction assuming the stored data do not always conform to the supposed representation. If p-

channel SLCs are used, the same query pattern from this paragraph may be used, provided we follow the same mid and hi definitions as mentioned in the previous paragraph. Note that when data stored on the SLCs do not always conform to the supposed representation, the first unary representation for p-channel behaves like the second unary representation for n-channel in the sense that a conducting column may have one of several patterns and thus be ambiguous. Conversely, the second unary representation for p-channel behaves like the first unary representation for n-channel in the sense that a conducting column must be conforming to supposed representation.

[00171] Furthermore, n-state MLCs may be used (albeit even more inefficiently than SLCs) for the first and alternative unary representations. If we denote cell state 0 being the erased state (i.e., having less negative charges stored on the floating gate compared to all other cell states) and state w+1 storing more negative charges than state w, and $V_{th}(w)$ being the threshold voltage for a cell in state w, then when $V_G*sgn \geq V_{th}(w)*sgn$, where sgn = +1 for n-channel and -1 for p-channel, the cell in state i will become conductible when probed with V_G. To adapt n-state MLCs to the unary representation, to represent a state i out of k possible states, MLC i should be in a cell state x, and all other k-1 MLCs should be in some other cell state y (but the k-1 MLCs need not have the same y) and satisfying $V_{G_sel}*sgn \geq V_{th}(x)*sgn$ and $V_{G_other}*sgn \geq V_{th}(y)*sgn$, where V_{G_sel} is the V_G applied to MLC i and V_{G_other} is the V_G applied to remaining k-1 MLCs when the query probing state is i, plus the following conditions. For adapting to the first unary representation, the following must hold true: $V_{th}(x) > V_{th}(y)$ and $V_{G_other} < V_{th}(x)$. Since the k-1 MLCs need not have the same y, V_{G_other} need not be the same for these k-1 MLCs either. If the y value for each of k-1 MLCs is predictable (i.e., deterministic), then

V_{G_other} can also be chosen deterministically and applied to each of k-1 MLCs. If the y values are not predictable, a large enough (but still not necessarily constant) V_{G_other} can be chosen. Also, V_{G_sel} need not remain constant across different probing operations, whether on the same or different series. To adapt n-state MLCs to the alternative unary representation, the following must hold true: $V_{th}(x) < V_{th}(y)$ and $V_{G_sel} < V_{th}(y)$. Similarly, the k-1 MLCs need not have the same y, and V_{G_other} need not be the same for these k-1 MLCs either. Also similarly, V_{G_sel} need not remain constant across different probing operations, whether on the same or different series. These first and alternative unary representations for n-state MLCs share the same "ambiguity" (or lack thereof) property as the SLC case, when data stored do not always conform to the definition. Also, the various "need not" statements in this paragraph can also apply to SLC case.

[00172] Also, an m-bit MLC may be used to represent k states where $k < 2^m$. For example, a 2-bit MLC may have only 3 states, by either using it as a 2-bit MLC but ignoring and never using state 3, or adjusting the input voltages such that the voltage for state 2 becomes equal to the voltage for what was previously state 3 as a 2-bit MLC. Such representation can still be used for interlocked design as-is, for example by replacing 2^m-1 with k-1 in the state pair definition, resulting in (i, k-i-1) instead of (i, 2^m-i-1). This may be useful, for example when the quantizer has only $k < 2^m$ quantization bins.

[00173] As mentioned before, an MLC's state may be described in terms of the input voltage required to cause a conduction channel. Usually both such state and input voltage are discrete, with 2^m states and voltages for an m-bit MLC. If instead, such state as well as the input voltage is allowed to be continuous, then it becomes an analog MLC, where the MLC can store a state as a real number. This analog behavior may be approximated

by using a very large m for the MLC. The amount of noise and/or variability in the amount of charges will affect the precision of the input voltage and determine how precise such an analog MLC is. If such a state is represented by a real number $x \in$ real number range [0,1), with 0 being the cell state requiring the smallest input voltage to conduct, then the interlocked representation for value x is preferably defined as the pair (x, 1-x), which is the limiting case of the m-bit MLC interlocked notation $(i, 2^m\text{-}i\text{-}1)$ after dividing it by 2^m, as $m \rightarrow \infty$. Of course, a range other than [0,1) may also be used.

[00174] In the following, a range query, for example a generalization of wild-cards for MLC, according to various embodiments will be described.

[00175] If the query input or cell state pair is not $(i, 2^m\text{-}i\text{-}1)$, then it may be a form of wild-card, or its generalization. First, at the reference side, it can represent a range interval instead of an always-match wild-card. To represent a range of [a,b], where a <= b and $a, b \in \{0, 1, 2, 3, \ldots, 2^m\text{-}1\}$, it may be stored on a pair of MLC fGTs as: $(a, 2^m\text{-}b\text{-}1)$, and at query time, for any input value $x \in \{0, 1, 2, 3, \ldots, 2^m\text{-}1\}$, a query probe pattern will be $(x, 2^m\text{-}x\text{-}1)$, and it will match (i.e., the fGT pair conducts) if and only if $x \in$ range [a,b]. This is because to make the top fGT conduct, x must be >= a, and to make the bottom fGT conduct, x must be <= b. It should be evident that such range match is a generalization of wild-card match. In this disclosure, querying over such a range is also referred to as a range query, which is a generalization of wild-card with its abstraction as described above.

[00176] A range query can be used to efficiently replace the more general small-range enumeration of weak projections, and works well when number of bits in the MLC (m) and in the quantizer (c) are the same (m == c). If m > c, then m-c bit(s) at fixed bit

positions (preferably the Least Significant Bits/LSBs) may be ignored or always set to 0, or such an m-bit cell can be used as a c-bit cell directly, since most MLCs are essentially the same as SLCs with the main exception in the use of many threshold voltages for both reading and writing operations. If $m < c$, a c-bit quantized value may need to be stored in 2 or more pairs of MLCs, which may affect the effectiveness of range query, because a single MLC range query description may not be able to cover a desired quantized value range exactly. Either multiple MLC range query descriptions may be needed, or a single MLC range query description may need to cover a larger than desired range. For example, if $m = 2$, $c = 4$, and desired quantized value range is [3,4] ([0011,0100] in binary), then an MLC range pair (Most Significant Bits/MSBs MLC listed first) of [0,1]:[0,3] is needed to cover [3,4] with a single MLC range query description, covering 8 of 16 possible quantized values. Alternatively, 2 MLC range pairs, [0,0]:[3,3] and [1,1]:[0,0], are needed to cover [3,4] exactly. Although multiple MLC range descriptions can cover the desired range more tightly, they have to be enumerated and the enumeration would increase search time. If $m = c = 4$, then a single MLC range query description, [3,4], same as the quantized range query, would suffice in this example.

[00177] If only SLCs are available, but a projection uses c-bit quantization and $c > 1$, the quantized value (c bits) may be stored in c interlocked pairs of SLCs within a column, preferably within the same series/string. In such cases, exact pattern matches works well, but non-trivial range query, where the range is not of the type [x,x], becomes difficult. This is because when the quantized value x increases or decreases by a small amount, each bit in x may change, depending on the value of x and the amount of increase/decrease. Therefore, it is hard to confine the number of weak bits to a small

number. Alternatively, Gray code may be used to encode a quantized value. Because Gray code is designed to have only 1 changed bit for any x to x+1, a short range like [x,x+1] will modify only 1 bit at a known bit position provided x is known, and this bit becomes the weak bit. For a range [x-1,x+1], 2 bits will be modified with known bit positions provided x is known, and these 2 bits can become the weak bits. With [x-1,x+1], 3 choices, x-1, x, x+1 are tested, but 2 weak bits generate 4 variations, so 1 out of 4 variations does not correspond to the range [x-1,x+1], which could cause more unrelated candidate matches. As the range becomes wider, even with Gray code, it becomes increasingly difficult and less effective because the number of modified (thus also weak) bits increases quickly. Both query and reference side small range queries can be supported using Gray code and weak bits on SLCs.

[00178] If the MLC is used in analog form as described above, then to represent a real number range [a,b], where a, b are real numbers with [0,1) and a <= b, the preferred interlocked notation is (a, 1-b), which is the limiting case of the m-bit MLC interlocked notation (a, 2^m-b-1) after dividing it by 2^m, as m $\rightarrow \infty$. If used as a reference-side range, [a,b] should be wide enough to cover the possible values of query value x with notation (x,1-x) if the corresponding reference vector is supposed to be the true NN of query vector. [a,b] should also be wide enough to cover the noise and/or variability both in the state of the analog MLC (reference side), and in the query side input voltage.

[00179] The range query can also be implemented at the query side. If the query input voltage pattern is (y, 2^m-x-1) (provided x <= y) and reference side pattern is (a, 2^m-b-1) (provided a <= b), then it tests whether query range [x,y] has any non-empty intersection (i.e., overlap) with reference range [a,b], i.e., it tests the condition ([x,y] $\cap$ [a,b]) $\neq \varnothing$, as

described above. This is because for two ranges to have an overlap, the condition y >= a and x <= b must hold, since b is represented as 2^m-b-1, x <= b becomes 2^m-x-1 >= 2^m-b-1, thus x needs to be represented as 2^m-x-1.

[00180] If the query input voltage pattern is instead (x, 2^m-y-1) and reference side pattern is still (a, 2^m-b-1) provided still x <=y and a <= b), then it tests whether query range [x,y] is a subset of reference range [a,b], i.e., it tests the condition ([x,y] $\subseteq$ [a,b]). This is because x tests whether x >= a, and y tests whether 2^m-y-1 >= 2^m-b-1, i.e., whether y <= b. As before, reference-side and query-side range query may be combined or used alone, and need not use the same criteria for choosing the range query.

[00181] If the reference side pattern is (a, 2^m-b-1) but a >= b, and query input voltage pattern is (x, 2^m-y-1), and if x > y, then it tests whether x >= a and y <= b, i.e., it tests whether reference-side range [b,a] $\subseteq$ query-side range [y,x] (note the ranges' boundaries are flipped compared to previous paragraphs). If x <= y, the test will always be false (i.e., the fGT pair will never conduct) since it would have to satisfy x >= a and y <= b, but since y >= x >= a > b, we have y > b and thus y <= b cannot be satisfied. The scenario where the test will always be false is called "anti-match" (in contrast to the "always-match" pattern). For SLC and 1-bit quantization, the "always-match" pattern at query-side is (1, 1) or in voltage (hi, hi), and at reference-side is (0, 0) or in charge state (erased, erased); whereas the "anti-match" pattern at query-side is (0, 0) or equivalently in voltage (mid, mid) provided reference does not store (0, 0) or equivalently in charge state (erased, erased), and at reference-side is (1, 1) or equivalently in charge state (programmed, programmed) provided query is not (1, 1) or in voltage (hi, hi). For the more general MLC and multi-bit quantization, whether a query or reference side pattern is "anti-match"

can be determined using the inequalities described earlier in this paragraph. "anti-match" may be used for example where a projection should be ignored for vote count update.

[00182] Table 2 below lists all the range query scenarios described above and what each scenario means. Note the 2nd scenario of x >= y as (x, 2^m-y-1) and a <= b is equivalent to previously described case of x <= y as (y, 2^m-x-1) and a <= b. Note that for analog MLC use case, the following table still applies provided the previously described discrete to analog range notation conversion is used.

Range query description in fGT state pair		Tests for	Meaning	Illustration
Query-side (x, 2^m-y-1)	Reference-side (a, 2^m-b-1)			
x <= y	a <= b	x >= a && 2^m-y-1 >= 2^m-b-1 $\Leftrightarrow$ x >= a && y <= b	[x,y] $\subseteq$ [a,b]	FIG. 25A
x >= y	a <= b		[y,x] $\cap$ [a,b] $\neq \varnothing$ (most often used)	FIG. 25B
x > y	a >= b		[b,a] $\subseteq$ [y,x]	FIG. 25C
x <= y	a > b		Anti-match (always false)	y >= x >= a > b conflicts with y <= b

Table 2. Types of range queries in an m-bit fGT MLC pair and their semantic meanings

[00183] FIG. 25A shows an illustration 2500, FIG. 25B shows an illustration 2502, and FIG. 25C shows an illustration 2504, like stated in Table 2.

[00184] Also it is to be noted that while the above description uses a pair of m-bit MLCs for interlocked design where the number of states in a cell is 2^m, it is also possible to use a different number of states, say integer k where $k > 1$. The interlocked design will still work with such k-state cell, if we change the interlocked notation from (a, 2^m-b-1) to (a, k-b-1) and from (x, 2^m-y-1) to (x, k-y-1) in Table A1. This may be useful if one wants a k-state (i.e., k bins or k intervals) quantization instead of a m-bit or c-bit (i.e., 2^m or 2^c state) quantization. For example, if the current NAND Flash technology can only support 3-bit MLC but 10-state quantization is desired, it may be feasible since 10 and 2^3 is not so different, but using it as a 4-bit MLC may result in much less reliable operation. Note that if $k = 2^m$, it becomes the same as the description we have before.

[00185] If p-channel fGTs are used, and the definition of states is the same as the default definition as described above, then the interlocked notation may be used as-is for Table 2 and the same semantics will be achieved. If the alternative definition of states as described above is used, where a larger state number i corresponds to more negative charges on the floating gate and thus a less negative threshold voltage, the query side notation (x, k-y-1) and reference side notation (a, k-b-1) may be changed to (k-x-1,y) and (k-a-1,b), respectively, and it will still achieve the same semantics as in Table A1 (except for the change of query and reference side notation).

[00186] Ternary Content Associative Memory (TCAM) has been proposed for very fast LSH-based ANN search. TCAM is hardware based and supports weak-bit without enumeration, similar in functionality to what has been described above, but both its design and implementation are very different from this invention, and it only supports reference-side weak bit and not query-side weak bit. Also, TCAM does not support non-trivial range query with multi-bit quantization where the number of bits $c > 1$. While very fast, TCAM is also very expensive and has high power consumption, making it unsuitable for large-scale databases. TCAM does not have the notion of vote count: a reference word is returned in search result if and only if the entire word (with weak bits if any) matches the query input word.

[00187] These range queries can help mitigate symbol error rates (SERs, which are generalization of BERs at projection level as opposed to bit level) that are caused by weak projections or bits, but in an even more effective manner than the wild-card. For example, instead of 1-bit quantization (with say 0 as quantization boundary), 2-bit quantization may be used (with say equal probability mass in each of 4 quantization bins), but coupled with range query, such that a reference side projection always covers 2 consecutive quantization bins instead of 1, and this results in effectively 1-bit differentiation. Here x-bit differentiation means that on a given projection, an unrelated vector would accidentally match the query vector (after applying query and reference side range query if any) with approximately $1/2^x$ probability (assuming each quantization bin has same probability mass). Compared to 1-bit quantization, such 2-bit (4 bins) quantization with 2-bin range query also achieves effectively 1-bit differentiation per projection, but reduces BER and SER, and therefore may help increase the upper limit of

number of pairs of fGTs probed in a series while maintain high enough recall and low enough collision count, and thus save even more power. Range query may also be applied directly to feature vectors.

[00188] FIG. 26 shows an illustration 2600 of an example of range query. A filled circle or rectangle indicates a non-trivial range query from query or reference side, respectively. 3-bit MLCs and 3-bit quantization assumed.

[00189] FIG. 26 shows an example of range query, by adapting from FIG. 11A. The quantized values are shown next to the circles denoting query input, and also shown next to the rectangles corresponding to reference frame 0 and 1. In this example, the query and reference side range query results in a match on reference frame 1. Note the m in FIG. 26 refers to the number of projections compared at a time, not to be confused with the number of bits in an fGT MLC.

[00190] The analog MLC paradigm allows certain interesting analyses on performance trade-offs among the width and location of the range in a range query, probability of successful matching for true NN on the said range, and the differentiation level (which affects collision count). As has been described above, a (reference-side) range quantizer may be provided that can achieve much lower collision rate at the same recall rate of a conventional 1-bit quantizer. The distribution of $\vec{a}.\vec{q}$, $\vec{a}.\vec{v}$ and $\vec{a}.\vec{v}_i$ may have a wide range, and it is to be noted as described above that since an analog MLC's state can describe a real number in [0,1), a monotonic, reversible mapping function may be used to map the allowed range of $\vec{a}.\vec{q}$, $\vec{a}.\vec{v}$ and $\vec{a}.\vec{v}_i$, such as $(-\infty,+\infty)$ to [0,1) using a monotonically increasing mapping function, and hence as has been described above, we can describe in terms of the range before mapping, for convenience.

[00191] As has been described above, the range used is a reference-side range, i.e., if using the interlocked design for n-channel fGTs, the cell pair for reference vector $\vec{v}$ may store two values (before mapping the allowed range of $\vec{a}.\vec{q}$, $\vec{a}.\vec{v}$ and $\vec{a}.\vec{v}_i$ such as (-∞,+∞) to [0,1)) $\vec{a}.\vec{v}-r$ and $-(\vec{a}.\vec{v}+r)$, respectively, and the query probing voltages (also before mapping) are preferably $\vec{a}.\vec{q}$ and $-\vec{a}.\vec{q}$. So this will test the conditions $\vec{a}.\vec{q} \geq \vec{a}.\vec{v}-r$ && $-\vec{a}.\vec{q} \geq -(\vec{a}.\vec{v}+r)$, the 2nd condition is equivalent to $\vec{a}.\vec{q} \leq \vec{a}.\vec{v}+r$, thus implement the test of whether $\vec{a}.\vec{q}$ falls within the range centered at $\vec{a}.\vec{v}$ and with radius r. Note if the projection distribution over the entire database or training dataset such as $pdf_{\vec{a}.\vec{v}_i}(x)$ is not centered at 0 but at some value c, then $pdf_{\vec{a}.\vec{v}_i}(x)-c$, which is centered at 0, can be used. Alternatively, the stored values may be $\vec{a}.\vec{v}-r-c$ and $-(\vec{a}.\vec{v}+r-c)$ instead of $\vec{a}.\vec{v}-r$ and $-(\vec{a}.\vec{v}+r)$. Alternatively, nothing needs to be done, but due to the off-0 center, the mapping function for mapping ranges to [0,1) may work less efficiently in the sense of being more susceptible to noise or imprecision such as when depositing charges to the fGT. Because the range is reference-side, the values are stored on the fGTs during database building time, and if radius r, which controls the robustness of the search, is to be changed on the fly, say at query time, it is hard to do because the entire database has to be rebuilt with the updated $\vec{a}.\vec{v}-r$ and $-(\vec{a}.\vec{v}+r)$ for each reference vector $\vec{v}$, to reflect the updated r.

[00192] According to various embodiments, a technique may be provided as will be described in the following, which may effectively adjust r on the fly at query time, by intelligently changing the query probing voltages. Instead of using $\vec{a}.\vec{q}$ and $-\vec{a}.\vec{q}$ as the query probing voltages, $\vec{a}.\vec{q}-\delta$ and $-\vec{a}.\vec{q}-\delta$ are used. This will test the conditions $\vec{a}.\vec{q}-\delta \geq \vec{a}.\vec{v}-r$ && $-\vec{a}.\vec{q}-\delta \geq -(\vec{a}.\vec{v}+r)$, which becomes $\vec{a}.\vec{q} \geq \vec{a}.\vec{v}-(r-\delta)$

and $\vec{a}.\vec{q} \leq \vec{a}.\vec{v} + (r - \delta)$, therefore, the effective radius has changed from r to $r - \delta$. If $\delta > 0$, this will reduce the effective radius, making the search less robust, and if $\delta < 0$, this will increase the effective radius, making the search more robust. In fact, it is even possible to initially choose $r = 0$ at database building time, but use δ to achieve the robustness desired. If $\delta = 0$, it becomes the default case described in the previous paragraph. Due to granularity and precision of depositing charges onto the fGT, as well as charge leakage over time, the effective radius (in this case, $r - \delta$) will usually have to be non-negligible to counteract those noises in actual charges vs. supposedly stored charges, so as to achieve a good balance of robustness, proper operation and low collision.

[00193] More formally, if we denote the monotonic, reversible mapping function from projected value x with a range such as (-∞,+∞) to a real number with a cell state range used by the range quantizer such as (0,1) (If the stored data i.e. cell state can support a range of [0,1), then an output range of (0,1) is preferred for $h_{map}(x)$ so that the interlocked pair do not exceed the [0,1) range when using the 1- operation.), as function $h_{map}(x)$, e.g. if $h_{map}(x)$ is monotonically increasing ($h_{map}(x)$ may also be monotonically decreasing, and the method described here can still work, by making $V_{th}(x)$*sgn to be a monotonic decreasing function instead. For brevity, such configuration is not described in further detail.), and we denote $V_{th}(x)$ as the threshold voltage when a cell stores a state x (where this x is in the range such as [0,1) or (0,1) and not (-∞,+∞)), and the state convention is chosen such that $V_{th}(x)$*sgn is a monotonic increasing function, where sgn = +1 for n-channel fGTs and -1 for p-channel fGTs. Then, $\vec{a}.\vec{v} - r$ and $\vec{a}.\vec{v} + r$ can be mapped to $h_{map}(\vec{a}.\vec{v} - r)$ and $1 - h_{map}(\vec{a}.\vec{v} + r)$ respectively (notice that we use 1- here instead of – operation in the previous paragraphs, because the stored data now has a range of [0,1)

instead of (-∞,+∞), and if the stored data range is [c,d) instead of [0,1), and $h_{map}(x)$ maps to (c,d), then the interlocked pair of $h_{map}(\vec{a}.\vec{v}-r)$ and $d-(h_{map}(\vec{a}.\vec{v}+r)-c)$ can be used instead.) and stored on the corresponding fGT pair as a reference-side range. At the query side, we denote function f(x) as a mapping function from a cell state x to a voltage V_G (i.e., applied to the Control Gate) that will cause the cell with a state x to be conductible, and f(x) may be defined as $V_{th}(x)$ (or $V_{th}(x)$ plus a small constant), and $\vec{a}.\vec{q}$ and the robustness adjustment δ create a pre-mapped range of $(\vec{a}.\vec{q}-\delta,\vec{a}.\vec{q}+\delta)$, which can be mapped to a query-side range of

$$\left(f\left(h_{map}(\vec{a}.\vec{q}-\delta)\right),f\left(1-h_{map}(\vec{a}.\vec{q}+\delta)\right)\right)$$

[00194] .Therefore, the query would test for

$$f\left(h_{map}(\vec{a}.\vec{q}-\delta)\right)\geq V_{th}(h_{map}(\vec{a}.\vec{v}-r)),\text{ i.e.,}$$

$$V_{th}\left(h_{map}(\vec{a}.\vec{q}-\delta)\right)\geq V_{th}(h_{map}(\vec{a}.\vec{v}-r)),$$

which for n-channel fGTs becomes $h_{map}(\vec{a}.\vec{q}-\delta)\geq h_{map}(\vec{a}.\vec{v}-r)$ and then becomes $\vec{a}.\vec{q}-\delta\geq\vec{a}.\vec{v}-r$ and thus $\vec{a}.\vec{q}\geq\vec{a}.\vec{v}-(r-\delta)$. Similarly, the query would also test for

$$f\left(1-h_{map}(\vec{a}.\vec{q}+\delta)\right)\geq V_{th}(1-h_{map}(\vec{a}.\vec{v}+r))$$

which for n-channel fGTs eventually becomes $\vec{a}.\vec{q}\leq\vec{a}.\vec{v}+(r-\delta)$, thus implementing the range query radius semantics plus robustness adjustment stated in previous paragraphs. For p-channel fGTs, which implements <= logic, i.e., $f\left(h_{map}(\vec{a}.\vec{q}-\delta)\right)\leq V_{th}(h_{map}(\vec{a}.\vec{v}-r))$ etc., because $V_{th}(x)$*sgn is a monotonic increasing function, which means for p-channel fGTs $V_{th}(x)$ is a monotonic decreasing function, it can be verified that it will also implement the range query radius semantics plus robustness adjustment.

[00195] If f(x) is defined as $V_{th}(x)$+c where c is a small non-zero constant, then it can be verified that its use shifts the center of the range, but in a non-trivial way and it also distorts the result so that it does not enforce the range quantizer radius and robustness adjustment semantics 100% of the time. Therefore, such f(x) is not the preferred choice compared to $V_{th}(x)$.

[00196] Furthermore, one can also choose a different radius and/or robustness adjustment for the left-side and right-side of a range, whether query-side or reference-side. E.g., , $\vec{a}.\vec{v} - r_1$ and $\vec{a}.\vec{v} + r_2$ may be used instead of $\vec{a}.\vec{v} - r$ and $\vec{a}.\vec{v} + r$, and/or $(\vec{a}.\vec{q} - \delta_1, \vec{a}.\vec{q} + \delta_2)$ may be used instead of $(\vec{a}.\vec{q} - \delta, \vec{a}.\vec{q} + \delta)$, and this would be a more generalized form of range quantizer based range query with robustness adjustment.

[00197] Note again that the top and bottom cells of a cell pair need not be adjacent to each other in the physical layout, as long as they are on the same series/string.

[00198] If the fGTs are p-channel, then it implements a <= logic instead of >= logic, but the above on-the-fly radius changing technique can still be used, by recognizing that analog MLC is a limiting case of m-bit MLC as $m \rightarrow +\infty$.

[00199] Although the hardware designs so far have only covered NAND Flash, any kind of storage cells that enter the conductable state if and only if a >= (like n-channel fGT) or <= (like p-channel fGT) test is satisfied between its probing voltage and the voltage that would have been required to cause the conductable state in the probed storage cell which is dependent on the data pattern stored in that storage cell, and that the conducting path of each probed storage cells against a query probing pattern are connected to each other (where each other means the conducting path) in series, can use the described interlocked design, weak-bit, range query, anti-match, and range quantizer.

[00200] A NAND Flash string may include or may consist of multiple Flash memory cells connected in series, whereby the D terminal of an fGT is connected to the S terminal of one of its adjacent fGTs (with potentially the exception at both ends of the string). Although the two serially connected D and S terminals of two such adjacent fGTs may be depicted as individual terminals in FIG. 21, these two terminals generally appear as one electrode in manufactured integrated circuits, as shown in FIG. 27.

[00201] FIG. 27 shows an illustration 2700 of an example of NAND Flash string in schematic (top) and in manufactured integrated circuit (bottom).

[00202] Furthermore, the D and S terminals may become invisible in some Flash memory designs, such as certain 3D NAND Flash designs like p-BiCS, TCAT, VSAT and VG. In some or all of those cases, the electrode that used to represent the two connected D and S terminals (n-type semiconductor on a p-type bulk in case of n-channel Flash) may disappear altogether, because the fringe electrical field emanating from each word line may be strong enough to form a conducting channel in the part of the bulk where the electrode used to be located. Such omission of electrodes is sometimes referred to as *junction-free* design, and makes manufacturing of 3D NAND Flash easier. For purpose of consistency in terminology, we still consider the concept of D and S terminals to be applicable in such designs that omit the electrodes, and this consideration of viewpoint is echoed in the literature too, as evidenced by the continued use of traditional schematic of NAND Flash. And because in 3D NAND Flash designs, the NAND string architecture is still used, the hardware design proposed in this disclosure is still applicable.

[00203] In some Flash memory designs, such as SONOS Flash in p-BiCS ,the memory cell is different from conventional fGTs because its charge storage site is not a

conventional floating gate (usually metal or semiconductor) but rather a charge-trapping insulator. However, they still implement a >= or <= semantics like conventional fGTs, and we still refer to them as fGTs in the general sense. Furthermore, other types of memory cells, such as based on Ferro-electricity, has been proposed to be used in a NAND string architecture, as a Fe-NAND Flash memory. Although the working mechanism of Fe-NAND Flash is storing the polarity of tiny ferroelectric crystals instead of trapping or releasing electrons, they still implement a >= or <= semantics like conventional fGTs, and is applicable to this invention. In fact, as previously mentioned, any memory cell technology that implements a >= (like conventional n-channel fGT) or <= (like conventional p-channel fGT) test semantics (over a specified operating range of word line voltage), is directly applicable to the proposed invention in this disclosure, and can use the described interlocked design, weak-bit, range query, anti-match, and range quantizer. Accordingly in this document, fGTs may be referred to in a general sense as any memory cell that implements a >= or <= test semantics.

[00204] With the interlocked design for fGT pairs, weak-bit support based on wild-cards, and its generalization to MLC representation and the more general range query support, such a modified NAND Flash based enhanced vote-count algorithm and system has the prospect of reducing ANN search time by several times, improving search success probability, reducing significantly the number of unrelated matches (i.e., low collision count) with the proposed range quantizer and for media fingerprinting in highly correlated data sets with an intelligent layout of data storage format that captures linear time trend information, and saving several orders of magnitude in power and energy, compared to the basic vote-count algorithm design and conventional solutions.

[00205] Enhanced Vote Count (EVC) according to various embodiments may provide a hardware-assisted fuzzy search algorithm, for example on high-dimensional data. Massive HW parallelism may make it fast. Every Database (DB) item forms its matching thread: $t_{search} = O(1)$. According to various embodiments, an interlocked design may be provided. Very low power may be consumed: Only a matching DB item will draw current; for example, an *m*-bit pattern may use $1/2^m$ power. According to various embodiments, a high search robustness may be ensured, which may allow a high degree of fuzziness in the search pattern. According to various embodiments, a vote-counting may both improve robustness and reduces false matching rate. Counting # (number) of matched patterns for each DB item may be provided. Only DB item(s) with counts larger or equal ($\geq$) to a pre-determined threshold may be reported. By modifying NAND Flash according to various embodiments, affordability and high density may be provided.

[00206] Applications of EVC according to various embodiments may include audio/video fingerprinting (for example content identification and/ or anti-piracy), similar image search, example-based image processing (for example super-resolution, de-noising, image compression, and so on), bioinformatics (for example DNA (deoxyribonucleic acid) pattern matching), and biometrics security (for example voiceprint, faceprint, and so on).

[00207] The query functionality of EVC may be provided as a network service, where the EVC hardware resides on a server machine, which may be located in a centralized infrastructure such as a Data Center, and a client machine may send a query request to the server via a transmission venue such as a data transport network like the IP (Internet Protocol) network, and the server performs the query according to the EVC query mechanism, and then returns the query result back to the client via the same or optionally

another transmission venue for use by the client. The query request may contain data that are useable in the EVC query mechanism, and for example may be a query feature vector, or one or more query patterns (where one pattern may correspond to simultaneously matching more than one projections at a time) containing quantized value(s), weak bit(s), and discrete and/or analog (e.g. floating-point) weak range(s), or other pattern elements conforming to the interlocked design.

[00208] According to various embodiments, a Flash cell may include or may be a floating gate transistor.

[00209] It may be considered what happens if all rows are probed with their expected voltages: For == 10111, it may be asked whether it may be probed with (mid hi mid mid mid), wherein mid = 1V, hi = 3.5V. It may seem ideal, but fGT may implement >= logic, so the series will conduct if stored bits are 11111 or 10111. According to various embodiments, == logic may be desired.

[00210] According to various embodiments, an interlocked design may be provided. A match pattern may draw power only if it matches. For example, for an 8-bit pattern, $1/2^8$ power vs. peak power may be provided. Various embodiments may work on NAND Flash. Various embodiments may allow weak-bit (don't care bit) based fuzzy search, which may reduce effective BER. According to various embodiments, even fuzzier weak-ranges may be provided, for example using NAND MLC, which may further reduces effective BER.

[00211] In an implementation aspects of EVC according to various embodiments, accessing circuits may include: an row address decoder; a word-Line (WL) decoder & driver. Commonly used may be 1 WL in a NAND string which has special (e.g., mid)

voltage. According to various embodiments, for EVC, each WL in a NAND string may have special voltage(s). According to various embodiments, furthermore the accessing circuits may include a column-specific vote-counting circuit; and a priority encoder for reporting matching column ID (identifier). According to various embodiments, Read/Write circuits may require no change and may use reference designs from NAND papers.

[00212] According to various embodiments, an apparatus may be provided for testing whether a query pattern matches the pattern stored on a string of data storage cells (for example also referred to as cells for brevity), where each cell consists of at least 3 terminals, denoted G (gate), D (drain), and S (source), with voltage of G controls the conductability state of the said cell between D and S, and the said query pattern controls the voltage of G at the said cells whose stored pattern is to be matched against, where a string is defined as a circuit in which two or more cells are connected in series with at least one of the said two or more cells' D connected to one of its adjacent cells' S, where the said query pattern is a sequence of query pattern elements, where the said sequence has at least two query pattern elements, and each query pattern element is converted to a set of voltages that are fed to the G terminals of a corresponding set of cells on the said string, where the cardinality of the said set of voltages is equal to the cardinality of the said corresponding set of cells and is at least two, where the said stored pattern on the said string is a sequence of stored pattern elements, where the said sequence has at least two stored pattern elements, and each stored pattern element is represented by a set of cells with the same cardinality as their corresponding said set of voltages in bullet item c, where the conductable state of a cell means the resistance of the said cell between its D

and S measured at a specified V_{DS} of the said cell is below a specified threshold, and if the cell is not in the conductable state, it is defined to be in the non-conductable state, where a cell is in the conductable state if the voltage of G on the said cell is within a range that is determined by the said cell's stored pattern, where at least two voltages out of all the said set(s) of voltages converted from the said query pattern are voltages that will cause the non-conductable state in any of the Corresponding Cells whose G terminals are fed by the said at least two voltages (referred to as CC) if a certain pattern is stored on CC, and where a match is deemed to occur if and only if all cells that are being matched against the said query pattern in the said string are in the conductable state during the matching operation.

[00213] According to various embodiments, the cardinality of the said set of voltages is two for all query pattern elements, that is, each said set of voltages are a pair of voltages and each said corresponding set of cells are a pair of cells, and the total number of cells whose stored pattern are to be matched against the said query pattern in the said string is an even number, the voltage of G is always measured with respect to Ground of the apparatus.

[00214] According to various embodiments, the said set of voltages are k voltages and the said corresponding set of cells are k cells, and k is a positive integer and $k > 1$, a cell C1 out of the said corresponding set of cells stores a value x and remaining k-1 cells store some other value y that is different from x, and the k-1 cells need not have the same y, the query pattern element, in general, is a value i that can take on one of k distinct values, where each of the k distinct values is mapped to a selected cell out of said k cells, where the said selected cell is distinct (i.e., unrepeated) for each of the k distinct values, where

the said selected cell is independent of C1, the stored pattern element of the said corresponding set of cells is a value j that can take on one of k distinct values, where each of the said k distinct values has a one-to-one correspondence with the k cells in the said corresponding set of cells, the voltage fed to the said selected cell, denoted V_{G_sel}, is chosen to be in a range dependent on value x that will cause the said selected cell to be in the conductable state if the said selected cell stores a value x, i.e., if the said selected cell is also C1, and the remaining k-1 voltages, each of which denoted V_{G_other}, are chosen to be in a range dependent on value y and x that will force any cell C2 of the corresponding k-1 cells into the conductable state if C2 stores value y, but that will force at least one of the said k-1 cells into the non-conductable state if C2 stores value x.

[00215] According to various embodiments, the said cells are floating gate transistors (fGTs) with G, D, S being the Control Gate, Drain, Source terminals of an fGT, respectively, and the said string is a NAND string as defined in NAND Flash terminology, the resistance threshold for determining a cell's conductability state is defined such that the said cell is in the conductable state if and only if its voltage at G (denoted V_G and measured with respect to the string's Bulk electrode and measured with respect to the string's Bulk electrode) satisfies $V_G*sgn \geq V_{th}*sgn$, where sgn is a variable denoting positive or negative signs and is defined as +1 for n-channel fGTs and -1 for p-channel fGTs, and where V_{th} is the threshold voltage of the said cell.

[00216] According to various embodiments, the fGTs support n cell states, i.e., can store an integer value between 0 and n-1, where $n \geq 2$, the value w and w+1 are defined as follows: for both n-channel and p-channel fGTs, value w+1 and w represents the more and the less amount of negative charges stored on the floating gate of an fGT,

respectively; $V_{th}(w)$ may be defined as the threshold voltage for a cell storing value w, and if there is variability or noise present in the amount of charges stored on the floating gate for storing value w, the said threshold voltage is defined as the minimum (for n-channel fGTs) or maximum (for p-channel fGTs) threshold voltage that can guarantee with a specified probability the conductable state of the said cell under the considered variability and/or noise conditions; satisfying $V_{G_sel}*sgn \geq V_{th}(x)*sgn$ and for any of the remaining k-1 voltages $V_{G_other}*sgn \geq V_{th}(y)*sgn$.

[00217] According to various embodiments, value x and y satisfy $V_{th}(x) > V_{th}(y)$, and for any of the remaining k-1 voltages $V_{G_other} < V_{th}(x)$. According to various embodiments, ,value x and y satisfy $V_{th}(x) < V_{th}(y)$, $V_{G_sel} < V_{th}(y)$.

[00218] According to various embodiments, the fGTs are used as k-state cells each capable of encoding k states, whose states are denoted as {0, 1, ...,k-1}, where a larger state number i for a cell always corresponds to more (for n-channel fGTs) or less (for p-channel fGTs) negative charges being stored on the said cell, and where $V_{th}(i)$ is the threshold voltage, and if there is variability or noise present in the amount of charges stored on the floating gate for storing state i, the said threshold voltage is defined as the minimum (for n-channel fGTs) or maximum (for p-channel fGTs) threshold voltage that can guarantee with a specified probability the conductable state of the said cell under the considered variability and/or noise conditions, a stored pattern element is a general range [a,b] and is represented by storing (a, k-b-1) on a pair of fGTs, where "general" means it is valid to have a > b for such a range, any state number i in the query pattern is converted to a voltage of f(i) that may be measured with respect to the NAND string's Bulk electrode and is fed to its corresponding cell, where f(i) is a monotonic function of i

over validly defined values of state number i, and f(i)*sgn is an increasing function, and f(i)*sgn $\geq V_{th}(i)$*sgn, and when i < k-1, also satisfying f(i)*sgn < $V_{th}(i+1)$*sgn, a pattern element in the said query pattern is a general range [x,y] and is represented by a pair of voltages f(x) and f(k-y-1), where "general" has the same meaning as in bullet item (b).

[00219] According to various embodiments, a larger state number i for a cell always corresponds to less (for n-channel fGTs) or more (for p-channel fGTs) negative charges being stored on the said cell, a stored pattern element is a general range [a,b] and is represented by storing (k-a-1,b) on a pair of fGTs, where "general" means it is valid to have a > b for such a range, f(i)*sgn is a decreasing function, a pattern element in the said query pattern is a general range [x,y] and is represented by a pair of voltages f(k-x-1) and f(y), where "general" has the same meaning as in bullet item (b).

[00220] According to various embodiments, the said query pattern element tests on the stored pattern element on the said pair of fGTs one of the following logical expressions, and each said logical expression is defined to be true if and only if the said pair of fGTs are in the conductable state under the said query pattern element:

a. $[x,y] \subseteq [a,b]$ by choosing x <= y and a <= b;

b. $[b,a] \subseteq [y,x]$ by choosing x > y and a >= b;

c. $[y,x] \cap [a,b] \neq \varnothing$ by choosing x >= y and a <= b;

d. $x \in [a,b]$ by choosing y =x and a <= b (special case of item a.);

e. $a \in [y,x]$ by choosing x > y and b = a (special case of item b.);

f. don't-care (i.e., a logical expression that always evaluates to true),

   i. by choosing a = 0 and b = k-1 for a reference-side don't care, and/or

   ii. by choosing x = k-1 and y = 0 for a query-side don't care,

g. anti-match (i.e., a logical expression that always evaluates to false) by choosing x <= y and a > b;

[00221] According to various embodiments, $k = 2^m$ and m is a positive integer and m > 1.

[00222] According to various embodiments, a, b, x, y are converted by a Gray code, and for each number in between [a,b] and [x,y] respectively, the Gray code is applied and the converted numbers from [a,b] and [x,y] are listed respectively, where the following is applied: For each converted number a1 from [a,b], all a1's are cross-compared and a range [a',b'] (where a' <= b' but a',b' is not necessarily the Gray code of a and b) that can cover all a1's, preferably a small range, are chosen and stored as reference-side range; AND; for each converted number x1 from [x,y], EITHER the $x1 \in [a',b']$ test, i.e., setting x = y, is performed, OR all x1's are cross-compared and a range [y',x'] (where x' >= y', but x',y' is not necessarily the Gray code of x and y) that can cover all x1's, preferably a small range, are chosen and used as query-side range to test for $[x',y'] \cap [a',b'] \neq \varnothing$.

[00223] According to various embodiments, quantization may require c bits and fGTs are m-bit MLCs and c > m, the c-bit quantized value is divided into multiple parts, where each part has <= m bits and is stored on one fGT, where the following is applied: Only exact pattern match is supported, by storing only exact bits and allowing only == tests; OR numbers in [a,b] and [x,y] are enumerated to obtain a set of ranges, one for each fGT holding part of the [a,b] or [x,y], so that collectively these ranges will cover all numbers in [a,b] and [x,y] respectively, and these ranges (reference-side for [a,b] and query-side for [x,y]) are used to perform range query test; Gray code may also be applied while obtaining these ranges, and these ranges are performed on Gray code converted numbers.

[00224] According to various embodiments, one or more strings are associated with a column, where each column is associated with a counter that is initially set to 0 at the beginning of a search operation, where L' rounds of pattern matches are tested and each round of pattern match is a test between a query pattern and the stored pattern on a corresponding string over one or more columns, where the counter of each said column is incremented by 1 on round i if and only if the query pattern at round i matches the stored pattern on the corresponding string of the said column, where after L' rounds are completed, the column(s) whose counter(s) is(are) greater than or equal to T*L' is(are) reported as candidates of the said search operation, where T is a threshold factor between 0 and 1.

[00225] According to various embodiments, the fGTs are used as analog cells each capable of storing an analog state up to the precision allowed by device characteristics, where such a state is denoted as a real number in the range of [c,d) where c < d, and a monotonically increasing function $h_{map}(x)$ is used to map a real number in the possible range of a projected value of any reference vector $\vec{v}$ on a projection vector $\vec{a}$, to (c,d), where $V_{th}(x)$ is the threshold voltage when a cell stores a state x, and the state convention is chosen such that $V_{th}(x)$*sgn is a monotonic increasing function, and $f(x) = V_{th}(x)$, where a stored pattern element for a reference vector $\vec{v}$ on a projection vector $\vec{a}$ given a specified reference-side robustness radius r corresponds to a range of $[\vec{a}.\vec{v} - r_1, \vec{a}.\vec{v} + r_2]$, and is stored as $h_{map}(\vec{a}.\vec{v} - r_1)$ and $f(d-(h_{map}(\vec{a}.\vec{q} + \delta_2)-c))$, on the correspond pair of cells, respectively, where a query pattern element for a query vector $\vec{q}$ on a projection vector $\vec{a}$ given a specified query-side robustness radius adjustment δ_1 and δ_2 corresponds to a tuple of values $(\vec{a}.\vec{q}, \delta_1, \delta_2)$, and is converted to a pair of voltages $f(h_{map}(\vec{a}.\vec{q} - \delta_1))$

and d-(f($h_{map}(\vec{a}.\vec{q}+\delta_2)$)-c), respectively, where $\vec{q}$'s projection on $\vec{a}$ is deemed to fall into $[\vec{a}.\vec{v}-(r_1-\delta_1), \vec{a}.\vec{v}+(r_2-\delta_2)]$, if the corresponding pair of cells are in the conductable state.

[00226] According to various embodiments, a monotonically increasing function $h_{map}(x)$ is used to map a real number in the possible range of a projected value of any reference vector $\vec{v}$ on a projection vector $\vec{a}$, to cell state [0,k-1], which is a discrete case that approximates analog MLC, and if k is very large, then the approximation will be very good, a stored pattern element for a reference vector $\vec{v}$ on a projection vector $\vec{a}$ given a specified reference-side robustness radius r corresponds to a range of $[\vec{a}.\vec{v}-r_1, \vec{a}.\vec{v}+r_2]$, and is stored as $h_{map}(\vec{a}.\vec{v}-r_1)$ and k-1-$h_{map}(\vec{a}.\vec{v}+r_2)$, on the correspond pair of cells, respectively, a query pattern element for a query vector $\vec{q}$ on a projection vector $\vec{a}$ given a specified query-side robustness radius adjustment δ_1 and δ_2 corresponds to a tuple of values $(\vec{a}.\vec{q}, \delta_1, \delta_2)$, and is converted to a pair of voltages f($h_{map}(\vec{a}.\vec{q}-\delta_1)$) and f(k-1-$h_{map}(\vec{a}.\vec{q}+\delta_2)$), respectively, $\vec{q}$'s projection on $\vec{a}$ is deemed to fall into $[\vec{a}.\vec{v}-(r_1-\delta_1), \vec{a}.\vec{v}+(r_2-\delta_2)]$, if the corresponding pair of cells are in the conductible state.

[00227] According to various embodiments, a monotonically decreasing function $h_{map}(x)$ is used, and $V_{th}(x)$*sgn is a monotonic decreasing function.

[00228] According to various embodiments, c = 0 and d = 1.

[00229] According to various embodiments, at least one of the following is true:

$r_1 = r_2 = r$, and/ or $\delta_1 = \delta_2 = \delta$.

[00230] According to various embodiments, the content from whom pattern elements are derived from has an intrinsic time relationship between successive snapshots of the

same piece of content, one or more pattern elements derived from each of two or more snapshots of a piece of reference content are stored in the same string of storage cells in the said apparatus, where the snapshots are separated by one or more predetermined time intervals, the said query pattern contains one or more pattern elements derived from each of two or more snapshots of a piece of query content, where the snapshots are separated by one or more predetermined time intervals; wherein for each snapshot, it may contribute one or more (stored) pattern elements, and they are placed in the same string.

[00231] According to various embodiments, any two snapshots adjacent to each other in time (whether the two belong to the said piece of reference content or belong to the said piece of query content) have the same predetermined time interval T, each snapshot contributes the same number of pattern elements in the same string, for both the said piece of query content and the said piece of reference content, the number of unique relative snapshot offsets stored on the said string is U, if pattern elements are derived at F frames per second, and if column 0 corresponds to time 0 and frame 0 of the said piece of reference content, then at column i, where n = int(i/F), i.e., n is the integer quotient of i/F, and j = i%F and % is the integer modulo operator, pattern elements derived from frame n*U*F+j, n*U*F+j+F,..., n*U*F+j+(U-1)*F are stored, the said pattern elements derived from the said snapshots in the said piece of query content are shifted in an integer multiple of T, where the minimum set of pattern elements that can be shifted out is referred to as a slot.

[00232] According to various embodiments, the said integer multiplier to T is enumerated from 0 to U-1, any vacant slot in the said query pattern due to shift is replaced by a query-side don't care pattern element, if the remaining non-vacant slot(s)

after shifting is(are) fewer than the number of slots shifted out, the query pattern elements from the said shifted out slots are used instead as the query pattern, given each said enumeration, pattern matches are performed for L' rounds as in claim 11, and the candidates from each said enumeration are combined to form a combined set of candidates.

[00233] According to various embodiments, an apparatus may be provided for testing whether a query pattern matches the pattern stored on a string of data storage cells (later referred to as cells for brevity), where each cell consists of at least 2+n terminals, denoted G_1, ..., G_n, D, and S, n >= 1 and n may vary from cell to cell, with voltages of G_1, ..., G_n control the conductability state of the said cell between D and S, and the said query pattern controls the voltages of at least a non-empty subset of G_1, ..., G_n at the said cells whose stored pattern is to be matched against, where a string is defined as a circuit in which two or more cells are connected in series with at least one of the said two or more cells' D connected to one of its adjacent cells' S, where the said query pattern is a sequence of query pattern elements, where the said sequence has at least two query pattern elements, and each query pattern element is converted to a set of voltages that are fed to at least a non-empty subset of the G_1, ..., G_n terminals of a corresponding set of cells on the said string, where the cardinality of the said set of voltages is at least the cardinality of the said corresponding set of cells and is at least two, where the said stored pattern on the said string is a sequence of stored pattern elements, where the said sequence has at least two stored pattern elements, and each stored pattern element is represented by a set of cells with the cardinality of at least two, where the conductable state of a cell means the resistance of the said cell between its D and S measured at a

specified V_{DS} of the said cell is below a specified threshold, and if the cell is not in the conductable state, it is defined to be in the non-conductable state, where a cell is in the conductable state if the voltages of at least a non-empty subset of G_1, ..., G_n on the said cell is within a range that is determined by the said cell's stored pattern, where at least two voltages out of all the said set(s) of voltages converted from the said query pattern are voltages that will cause the non-conductable state in any of the Corresponding Cells whose at least a non-empty subset of G_1, ..., G_n terminals are fed by the said at least two voltages (referred to as CC) if a certain pattern is stored on CC, where a match is deemed to occur if and only if all cells that are being matched against the said query pattern in the said string are in the conductable state during the matching operation.

[00234] According to various embodiments, n may not vary across cells.

[00235] According to various embodiments, the cell may have 3-input control gates, but only two may be used, and the remaining one may always have high voltage (i.e., does not add to non-conductability).

[00236] According to various embodiments, 2 cells may store 1 value, but if each cell is n-input, cardinality of input voltages is 2*n, not 2.

[00237] According to various embodiments, a single input control gate may control the On/Off state of a Flash cell. However, in various embodiments, a Flash cell may have multiple input control gates, as shown in FIG. 28A and FIG. 28B.

[00238] FIG. 28A shows a schematic symbol 2800 of the Flash cell. FIG. 29B shows an equivalent circuit 2802.

[00239] So in one embodiment, if it is a specific Vg range that will form a conducting channel in the Flash cell between Drain and Source, in another embodiment it may be any

voltage combination (Vg1, Vg2, ..., Vgn) that will form a conducting channel in the cell between Drain and Source; This may apply to both n-channel and p-channel multi-input Flash cells, although for p-channel, an overall more negative Vg1, ... Vgn is more likely to form a conducting channel.

[00240] According to various embodiments, the cells may still store data in interlocked form as usual, but each value in the query pattern is converted to up to 2*n voltages (as opposed to 2 voltages).

[00241] According to various embodiments, to determine the exact data to be stored, it may be assumed that n-1 inputs to a cell are fixed at a certain combination of voltages, while remaining c inputs ($1 \leq c \leq n$) may be shorted to the same input voltage i.e., degenerating an n-input cell to a 1-input cell when storing data. However, when querying, all n-inputs may be used.

[00242] While the invention has been particularly shown and described with reference to specific embodiments, it should be understood by those skilled in the art that various changes in form and detail may be made therein without departing from the spirit and scope of the invention as defined by the appended claims. The scope of the invention is thus indicated by the appended claims and all changes which come within the meaning and range of equivalency of the claims are therefore intended to be embraced.

Claims

What is claimed is:

1. A testing apparatus comprising:

a plurality of cells, each cell of the plurality of cells comprising a plurality of terminals, the plurality of cells configured to define a string, wherein each cell of the plurality of cells comprises a first terminal;

a controller configured to control voltages to the respective first terminals of the plurality of terminals of each cell of the plurality of cells based on a query pattern, wherein a state of each cell of the plurality of cells is defined by the voltage supplied to the first terminal of the cell; and

a determination circuit configured to determine whether the string corresponds to the query pattern based on the states of the plurality of cells.

2. The testing apparatus of claim 1,

wherein the plurality of cells are connected in series to define the string.

3. The testing apparatus of claim 1,

wherein states of each cell of the plurality of cells comprises a conductable state and a non-conductable state.

4. The testing apparatus of claim 3,

wherein each cell of the plurality of cells is configured to be in the conductable state if the voltage supplied to the first terminal of the cell is in a pre-determined range.

5. The testing apparatus of claim 3,

wherein the determination circuit is configured to determine that the string corresponds to the query pattern if the states of all cells of the plurality of cells is the conductable state.

6. The testing apparatus of claim 1,

wherein each cell of the plurality of cells comprises a transistor.

7. The testing apparatus of claim 6,

wherein the first terminal comprises a gate terminal.

8. The testing apparatus of claim 1,

wherein two cells of the plurality of cells define one bit of the string.

9. The testing apparatus of claim 8,

wherein a bit of the string may have a value selected from a list of values consisting of: low; high; and don't care.

10. The testing apparatus of claim 1,

configured to provide a test for at least one of: audio fingerprint; video fingerprinting; content identification; anti-piracy; similar image search; example-based image processing; super-resolution; de-noising; image compression; bioinformatics; DNA pattern matching; biometrics security; voiceprint; or faceprint.

11. A server comprising:

a receiver configured to receive a query pattern from a client;

the testing apparatus according to claim 1; and

a transmitter configured to transmit a result determined by the determination circuit of the testing apparatus to the client.

12. A testing method comprising:

controlling voltages to a respective first terminal of a plurality of terminals of each cell of a plurality of cells based on a query pattern, wherein the plurality of cells are configured to define a string, wherein a state of each cell of the plurality of cells is defined by the voltage supplied to the first terminal of the cell; and

determining whether the string corresponds to the query pattern based on the states of the plurality of cells.

13. The testing method of claim 12,

wherein the plurality of cells are connected in series to define the string.

14. The testing method of claim 12,

wherein states of each cell of the plurality of cells comprises a conductable state and a non-conductable state.

15. The testing method of claim 14,

wherein each cell of the plurality of cells is in the conductable state if the voltage supplied to the first terminal of the cell is in a pre-determined range.

16. The testing method of claim 14, further comprising:

determining that the string corresponds to the query pattern if the states of all cells of the plurality of cells is the conductable state.

17. The testing method of claim 12,

wherein each cell of the plurality of cells comprises a transistor.

18. The testing method of claim 17,

wherein the first terminal comprises a gate terminal.

19. The testing method of claim 12,

wherein two cells of the plurality of cells define one bit of the string.

20. The testing method of claim 19,

wherein a bit of the string may have a value selected from a list of values consisting of: low; high; and don't care.

21. The testing method of claim 12, further comprising:

providing a test for at least one of: audio fingerprint; video fingerprinting; content identification; anti-piracy; similar image search; example-based image processing; super-resolution; de-noising; image compression; bioinformatics; DNA pattern matching; biometrics security; voiceprint; or faceprint.