3/5-32
272 331
118

N90-21895

February 15, 1990

A VLSI Decomposition of the deBruijn Graph

O. Collins
Johns Hopkins University, Baltimore, Maryland

S. Dolinar, R. McEliece, and F. Pollara
Communications Systems Research Section

A new Viterbi decoder for convolutional codes with constraint lengths up to 15, called the Big Viterbi Decoder, is under development for the Deep Space Network. It will be demonstrated by decoding data from the Galileo spacecraft, which has a rate 1/4, constraint-length 15 convolutional encoder on board. In this article, the mathematical theory underlying the design of the very-large-scale-integrated (VLSI) chips that are being used to build this decoder is explained. The deBruijn graph B_n describes the topology of a fully parallel, rate $1/\nu$, constraint length $n+2$ Viterbi decoder, and it is shown that B_n can be built by appropriately "wiring together" (i.e., connecting together with extra edges) many isomorphic copies of a fixed graph called a "B_n building block." The efficiency of such a building block is defined as the fraction of the edges in B_n that are present in the copies of the building block. It is shown, among other things, that for any $\alpha < 1$, there exists a graph G which is a B_n building block of efficiency $> \alpha$ for all sufficiently large n. These results are illustrated by describing a special hierarchical family of deBruijn building blocks, which has led to the design of the gate-array chips being used in the Big Viterbi Decoder.

I. Introduction and Summary

The nth order *deBruijn graph* B_n is the state diagram for an n-stage binary shift register. It is a directed graph with 2^n vertices, each labeled with an n-bit binary string, and 2^{n+1} edges, each labeled with an $(n+1)$-bit binary string. The vertex labels represent the contents of the shift register at a given point of time. The label on an edge connecting one vertex to another represents the contents of the shift register preceded by the bit that is being input to the shift register as it changes from one state to the next. Figure 1 is a representation of B_3.

The deBruijn graph B_n gives the topology for a fully parallel Viterbi decoder for any rate $1/\nu$ convolutional code with a constraint length of $n+2$ ([3], Chapter 7). In such a decoder, a "butterfly" (a pair of add-compare-select units) must be located at each vertex of the graph and all communication between butterflies takes place along the edges of the graph. In fact, the Deep Space Network (DSN) is currently developing such a decoder, called the Big Viterbi Decoder (BVD), for constraint-length 15 convolutional codes, to be used for the Galileo mission using a rate 1/4 code. The BVD has 2^{13} butterflies connected according to the topology of B_{13}. It is constructed from 256 identical gate-array chips, each containing 32 butterflies. These chips are arranged on 16 identical printed-circuit boards, each containing 16 chips. Of the 2^{14} "wires" (butterfly interconnections) in the decoder, 56 percent are internal to the chips, another 17 percent are internal to the boards, and 27 percent are inter-board, or "backplane" connections. Furthermore, these chips and boards are universal, in the sense that any deBruijn graph B_n with $n \geq 5$ can be built from copies of these same chips, and any B_n with $n \geq 9$ can be built from copies of these same boards. In this article, the theoretical background that led to the design of these chips and boards is given. See [1] and [5] for further details. (The BVD will be discussed at the end of the article—see Example K.)

II. Preliminaries About Strings

In this section, some notation is introduced and a few elementary facts about binary strings, which will be needed throughout the rest of the article, are established.

Definitions. A binary string is a sequence of bits, i.e., 0s and 1s. The *length* of a binary string x, denoted by $|x|$, is the number of bits in x. The *empty string* ϵ is the string with no bits. Thus, $|\epsilon| = 0$. The set of all strings of length n is denoted by $\{0,1\}^n$. If x and y are two strings, the *concatenation* of x and y, denoted by xy or $x * y$, is the string obtained by following the bits of x by the bits of y. If $x = a * b * c$, then a is called a *prefix*, b is a *substring*, and c is a *suffix* of x. If $b * c$ isn't empty, then a is called a *proper prefix* of x; if either a or c is nonempty, b is called a *proper substring* of x; and, if $a * b$ isn't empty, c is called a *proper suffix* of x. If x is a nonempty binary string, then the symbol x^L (the *left part* of x) denotes the string obtained by removing the rightmost bit of x; similarly, x^R (the *right part* of x) denotes the string obtained by removing the leftmost bit of x.

If S and T are sets of binary strings, then S *covers* T if every string in T has a substring in S. Similarly, S *prefixes* T if every string in T has a prefix in S. If T consists of a single string t, S covers or prefixes t. Also, S is *irreducible* if no string in S is a substring of any other. Finally, the *cost* of a set of strings S is $\text{cost}(S) = \sum_{s \in S} 2^{-|s|}$, where $|s|$ denotes the length of the string s.

Example (A). If $x = 1011$, then $|x| = 4$, $x^L = 101$, and $x^R = 011$. The set $S = \{10, 111\}$ covers $\{010, 100, 101, 110, 111\}$, and $\{1, 0000\}$ covers $\{0,1\}^n$ for all $n \geq 4$. Similarly, $\{1, 000\}$ prefixes $\{1, 00000\}$, and $\{0, 10, 110, 111\}$ prefixes $\{0,1\}^n$ for all $n \geq 3$. Also, $\{1, 000\}$ is irreducible, but $\{1, 001\}$ is not. The cost of the set $\{10, 111\}$ is 3/8, $\text{cost}(\{1, 000\}) = 5/8$, and $\text{cost}(\{0,1\}^n) = 1$, for all $n \geq 1$.

Theorem (1). If S is an irreducible set of strings, then every string x covered by S can be factored uniquely in the form $x = \lambda s \rho$, where $s \in S$, λ and ρ are (possibly empty) strings, and $(\lambda s)^L$ has no substring from S. This factorization will be called the *S-factorization* of x.

Proof: Since x is covered by S, x will have one or more substrings from S. Among these S-substrings, there will be a unique *leftmost* one, since no string in S covers any other. Call this unique leftmost S-substring s. Then plainly, $x = \lambda s \rho$ is the desired unique factorization.

Example (B). As noted above, $S = \{1, 0000\}$ is irreducible and covers all strings of length 4. The S-factorization of 1010 is $\epsilon * 1 * 010$, the S-factorization of 0101 is $0 * 1 * 01$, and the S-factorization of 0000 is $\epsilon * 0000 * \epsilon$.

Lemma (1). If S is irreducible and covers $\{0,1\}^n$, then every string x of length n or greater will have a unique S-factorization, and if $x = \lambda s \rho$ is this factorization, then $|\lambda s| \leq n$.

Proof: Every string of length n or greater will have a substring of length n. This substring will be covered by S and, hence, so will x. Now let x be a string of length $\geq n$, and let $x = \lambda s \rho$ be its S-factorization, as described in Theorem (1). By definition of the S-factorization, $(\lambda s)^L$ is not covered by S. However, if $|\lambda s| > n$, then $|(\lambda s)^L| \geq n$, which would imply that $(\lambda s)^L$ is covered by S, a contradiction.

III. deBruijn Graphs and Subgraphs

The deBruijn graph B_n, which is the state diagram for an n-stage shift register, can be described as follows. There are 2^n vertices, each labeled with an n-bit binary string x. There is a directed edge from the vertex with label x to exactly two other vertices, viz., those with labels $0x^L$ and $1x^L$. The edge from x to $0x^L$ is labeled $0x$ and the edge from x to $1x^L$ is labeled $1x$. Similarly, there are exactly two edges directed into x, from $x^R 0$ and $x^R 1$, which are labelled $x0$ and $x1$. This definition is summarized in Fig. 2(a). For example, in Fig. 1, from the vertex 101 there are edges leading to $0(101)^L = 010$ and to $1(101)^L = 110$. Equivalently, one can define B_n by saying that it has 2^{n+1} edges, each labeled with an $(n+1)$-bit binary string, and that the edge labeled X connects the vertices with n-bit labels X^R and X^L. This alternative definition is summarized in Fig. 2(b). For example, in Fig. 1, the edge labeled 1011 is a directed edge from $(1011)^R = 011$ to $(1011)^L = 101$.

Next, the notion of a *subgraph* of a deBruijn graph needs to be defined. In this definition, and later, the symbols $E(G)$ and $V(G)$ stand for the number of edges and vertices in the graph G. Thus, for example, $E(B_n) = 2^{n+1}$ and $V(B_n) = 2^n$.

Definition. If H and G are graphs, H is called a *G-subgraph*, written $H \subseteq G$, if H has the same vertex set as G and an edge set that is a subset of the edge set of G. The *density* of a G-subgraph H, denoted by $\text{den}(H : G)$, is defined as $\text{den}(H : G) = E(H)/E(G)$.

Example (C). A B_n-subgraph of density 0 consists of 2^n isolated vertices, and a B_n-subgraph of density 1 is B_n itself. Figure 3 shows a B_3-subgraph of density 6/16, consisting of the eight vertices of B_3 and the six edges labeled $\{0010, 0011, 0100, 0101, 0101, 0111\}$.

Our goal is to build a large deBruijn graph B_N by connecting together multiple copies of a smaller graph, called a "building block." If one thinks of the building blocks as very-large-scale-integrated (VLSI) chips, it is natural to want to minimize the number of edges needed to connect the building blocks together. This goal leads to the following definition.

Definition. A graph H is a *building block* for a graph G if there exists G-subgraph $\bar{H}$, which is the disjoint union of several copies of H. The *efficiency* of H as a building block for G, denoted by $\text{eff}(H : G)$, is defined to be $\text{den}(\bar{H} : G)$. In other words, $\text{eff}(H : G)$ represents the fraction of the edges of G that are accounted for by the edges in the building blocks.

Theorem (2). If H is a building block for G, then

$$\text{eff}(H : G) = \frac{V(G)E(H)}{V(H)E(G)}$$

Proof: The G-subgraph $\bar{H}$ has the same number of vertices as G and is the disjoint union of several disjoint copies of H. Since G has $V(G)$ vertices, and H has $V(H)$ vertices, this means that $\bar{H}$ is the union of exactly $V(G)/V(H)$ copies of H. Since each of these copies of H has $E(H)$ edges,

$$E(\bar{H}) = E(H)(V(G)/V(H))$$

Thus,

$$\begin{aligned}\text{eff}(H : G) &= \text{den}(\bar{H} : G)\\ &= E(\bar{H})/E(G)\\ &= (E(H)(V(G)/V(H))/E(G)\\ &= (V(G)E(H)/V(H)E(G))\end{aligned}$$

Example (D). Any B_n-subgraph H is a building block of efficiency $\text{den}(H : B_n)$ for B_n. A B_n-subgraph of density 0 is a building block of efficiency 0 for any B_N with $N \geq n$. A B_n-subgraph of density 1 (i.e., B_n itself) cannot be a building block for any larger deBruijn graph, since the disjoint union of 2^{N-n} copies of B_n is a disconnected graph with the same number of edges as B_N, which is connected. In Fig. 4 are two copies of the graph H (from Fig. 3), relabeled and wired together with 20 new edges to form a graph isomorphic to B_4. Since B_4 has 32 edges, and the two copies of H together have 12 edges, it follows that H is a building block for B_4 of efficiency $12/32 = 37.5$ percent. In fact, the graph in Fig. 3 is a building block of efficiency 3/8 for *any* B_N with $N \geq 3$, as will be seen in Example (F), below. The building block of Fig. 3 is an example of what is called a *universal deBruijn building block.*

Definition. A *universal deBruijn building block of order n* is a B_n-subgraph that is a building block for any deBruijn graph B_N with $N \geq n$.

The following theorem shows that it is easy to compute the efficiency of any universal deBruijn building block.

Theorem (3). Let H be a universal deBruijn building block of order n. Then, for all $N \geq n$, $\text{eff}(H : B_N) = \text{den}(H : B_n)$. This common value will be called the *efficiency* of H as a deBruijn building block.

Proof: By Theorem (3), the efficiency of H as a building block for B_N is

$$\begin{aligned}\text{eff}(H : B_N) &= \frac{V(B_N)}{E(B_N)} \cdot \frac{E(H)}{V(H)}\\ &= \frac{2^N}{2^{N+1}} \cdot \frac{2^{n+1}\text{den}(H : B_n)}{2^n}\\ &= \text{den}(H : B_n)\end{aligned}$$

In the next section, a general construction for universal deBruijn subgraphs is described (Theorem 4), and in

Section V (Theorem 9) it will be seen that there exist universal deBruijn subgraphs whose efficiency approaches 1 as n approaches infinity.

IV. A General Construction for Universal deBruijn Subgraphs

In this section, the main theorem (Theorem 4) is presented, which gives a general construction for universal deBruijn building blocks. The key to this construction is a way of constructing a B_n-subgraph from a set of strings of length $\leq n$.

Definition. If S is a set of strings, $B_n(S)$ is defined to be the B_n-subgraph obtained by deleting from B_n all edges whose labels have a prefix in S.

Lemma (2). If S is irreducible, $E(B_n(S)) = 2^{n+1} \times (1 - \text{cost}(S))$; equivalently, $\text{den}(B_n(S) : B_n) = 1 - \text{cost}(S)$.

Proof: The 2^{n+1} edges in B_n are labeled with the $(n+1)$-bit strings, and there are 2^{n+1} of them. For each $s \in S$, there are exactly $2^{n+1-|s|}$ $(n+1)$-bit strings with s as a prefix. Since no $(n+1)$-bit string can have two prefixes in S (no string in S is a prefix of any other, since S is irreducible), the subgraph $B_n(S)$ will have exactly $2^{n+1} - \sum_{s\in S} 2^{n+1-|s|} = 2^{n+1}(1 - \text{cost}(S))$ edges.

The main theorem of this article is the following.

Theorem (4). If S is irreducible and covers $\{0,1\}^n$, then $B_n(S)$ is a universal deBruijn building block of order n with efficiency $1 - \text{cost}(S)$.

The proof of Theorem (4) is postponed until several examples have been given and a stronger but more technical result has been stated and proved (Theorem 5).

Example (E). The set $S = \{0,1\}$ is irreducible, has cost 1, and covers $\{0,1\}^n$ for any $n \geq 1$. The corresponding subgraph $B_n(S)$ is a B_n-subgraph of density zero, and is plainly a building block of efficiency zero for any deBruijn graph with $N \geq n$.

Example (F). The set $S = \{1, 000\}$ is irreducible, has cost 5/8, and covers $\{0,1\}^3$. In this case, $B_3(S)$ is identical to the B_3-subgraph in Fig. 3. Thus, Theorem (4) implies that the graph $B_3(\{1,000\})$ is a universal deBruijn building block with efficiency 3/8, as asserted in Example (D).

The next theorem concerns a family of relabeled copies of the graph $B_n(S)$. If A is any binary string, the graph $B_n(S, A)$ is constructed from $B_n(S)$ by inserting the string A into each vertex or edge label just after the first (leftmost) occurrence of a substring from S. In Fig. 5, this construction is illustrated for the graph $B_3(\{1,000\})$.

Theorem (5). For all $N \geq n$,

$$\bigcup_{|A|=N-n} B_n(S, A) = B_N(S)$$

Example (G). In Fig. 6, Theorem (5) is illustrated by showing the two graphs $B_3(S, 0)$ and $B_3(S, 1)$ for $S = \{1, 000\}$. When taken together, these two graphs form the graph $B_4(S)$, which is a subgraph of B_4. Thus, by adding the 20 edges whose labels have a prefix from S (16 with prefix 1 and 4 with prefix 000), B_4 is obtained, and indeed, this is how one arrives at Fig. 4.

Proof of Theorem (5): Call the graph

$$\bigcup_{|A|=N-n} B_n(S, A)$$

appearing in the statement of the theorem the ***union graph.*** To prove Theorem (5), one needs to show that the union graph is a B_N-subgraph, and that its edges are exactly those whose labels have no prefix from S. To do this, one must prove the following four assertions (A always denotes a string of length $N - n$):

(1) Every N-bit string occurs as a vertex label in some $B_n(S, A)$.

(2) Every edge in $B_n(S, A)$ is an edge of the deBruijn graph B_N.

(3) No edge label in $B_n(S, A)$ has an S-prefix.

(4) Every $(N+1)$-bit string without an S-prefix appears as an edge label on some $B_n(S, A)$.

Taken together, (1) and (2) show that the union graph is a B_N-subgraph, and (3) and (4) show that the edge labels in the union graph are the $(N+1)$-bit strings without an S-prefix.

Proof of (1): Let X be an arbitrary N-bit string, and let $X = \lambda s \rho$ be its S-factorization. By Lemma (1), $|\lambda s| \leq n$, so that $|\rho| \geq N - n$. If the leftmost $N - n$ bits of ρ are denoted by A, then $\rho = A\rho'$ and, hence, $X = \lambda s A \rho'$. It follows that X appears as the label of the vertex $\lambda s \rho'$ in $B_n(S, A)$. For example, if $n = 3$, $S = \{1, 000\}$, and $N = 8$, the 8-bit string $X = 01011110$ has S-factorization

$0 * 1 * 011110$. The first five bits of 011110 are 01111, and so 01011110 appears as the label on vertex 010 in $B_3(S, 01111)$.

Proof of (2): If an $(n+1)$-bit edge label x in $B_n(S)$ has S-factorization $x = \lambda s \rho$, then neither λ nor ρ is empty: λ isn't empty, because no edge label in $B_n(S)$ has an S-prefix, and ρ isn't empty, since $|x| = n+1$, and by Lemma (1), any S-factorization has $|\lambda s| \le n$. Thus, in $B_n(S)$, an edge with S-factorization $\lambda s \rho$ connects the vertices with labels $(\lambda s \rho)^R = \lambda^R s \rho$ and $(\lambda s \rho)^L = \lambda s \rho^L$. Furthermore, this representation of the vertex labels must in fact be the S-factorization of them, since an earlier occurrence of a substring from S in either $\lambda^R s \rho$ or $\lambda s \rho^L$ would imply an earlier occurrence of an S-string in $\lambda s \rho$. This means that in the graph $B_n(S, A)$, the edge $\lambda s A \rho$ connects the vertices $\lambda^R s A \rho = (\lambda s A \rho)^R$ and $\lambda s \rho^L = (\lambda s A \rho)^R$. In other words, an edge with label X in $B_n(S, A)$ connects X^R to X^L, and by the definition in Fig. 2(b), this is an edge in the deBruijn graph B_n. For example, in Fig. (5), the edge with label $001A0$ connects $001A0^R = 01A0$ to $001A0^L = 001A$, and this is an edge in the deBruijn graph $B_{|A|+3}$, for any string A.

Proof of (3): Let X be an edge label in the graph $B_n(S, A)$. Then by definition, X has the form $X = \lambda s A \rho$, where $\lambda s \rho$ is the S-factorization of an edge label in $B_n(S)$ with λ nonempty. If X had an S prefix, say s', then either s' would be a prefix of $\lambda s \rho$, or s would be a proper prefix of s'. But both of these alternatives are impossible: s' cannot be a prefix of $\lambda s \rho$, since $\lambda s \rho$, being an edge of $B_n(S)$, has no S-prefix; and s cannot be a proper substring of s', since no string in S is a proper substring of any other. Thus, every edge in $B_n(S, A)$ is an edge in $B_N(S)$, as asserted.

Proof of (4): To prove that every $(N+1)$-bit string with no S-prefix occurs as an edge label in $B_n(S, A)$ for some A, let X be such a string and let $X = \lambda s \rho$ be its S-factorization, in which necessarily λ is nonempty. If A denotes the leftmost $N-n$ bits of ρ, then as above, $X = \lambda s A \rho'$. The string $\lambda s \rho'$ cannot have a prefix in S, for if s' were such a prefix, then either s' would be a prefix of X, or else s would be a proper substring of s' (since λ is nonempty), and both of these alternatives are impossible. Thus, $\lambda s \rho'$ is the label on an edge of $B_n(S)$, and so $X = \lambda s A \rho'$ appears as the label corresponding to that edge in the graph $B_n(S, A)$. For example, let $S = \{1, 000\}$, $n = 3$, and $N = 8$, and consider the nine-bit string $X = 001011100$, which has no S-prefix. The S-factorization of X is $X = 00 * 1 * 011100$. The first five bits of 011100 are 01110, and so X appears as the label on the edge 001 in the graph $B_3(S, 01110)$.

This completes the proof of Theorem (5).

This section concludes with the proof of Theorem (4).

Proof of Theorem (4): Theorem (5) explicitly shows that the union of 2^{N-n} copies of $B_n(S)$ forms a subgraph (namely, $B_N(S)$) of the big deBruijn graph B_N, and so B_N can be constructed simply by adding the edges missing from B_N to this union. Thus, B_n is a universal deBruijn building block. According to Theorem (3), the efficiency of a universal deBruijn building block is the same as its density and, by Lemma (2), the density of $B_n(S)$ is $1 - \text{cost}(S)$.

V. Construction of Low-Cost Covers for $\{0,1\}^n$

In Theorem (4), it was shown how to construct universal deBruijn building blocks from covering sets for $\{0,1\}^n$ of small cost, but only a few examples were cited. In this section, several general constructions for low-cost covers for $\{0,1\}^n$ will be given, thereby automatically producing [via Theorem (4)] efficient universal deBruijn building blocks.

To produce a cover for $\{0,1\}^n$, begin with an arbitrary irreducible set S of strings of length $\le n$, which will be called a *precover* for $\{0,1\}^n$. In general, S will fail to cover a certain subset of $\{0,1\}^n$, which will be called $\text{Omit}_n(S)$. Denote the number of strings in $\text{Omit}_n(S)$ by $\text{omit}_n(S)$. Plainly, if $\text{Omit}_n(S)$ is adjoined to S, the resulting set, denoted by $C_n(S)$, will still be irreducible, will cover $\{0,1\}^n$, and its cost will be $\text{cost}(S) + 2^{-n}\text{omit}_n(S)$. This simple but useful construction is summarized in the following theorem.

Theorem (6). For any irreducible set S of strings of length $\le n$, the set $C_n(S)$ is an irreducible cover of $\{0,1\}^n$ of cost $(\text{cost}(S) + 2^{-n}\text{omit}_n(S))$.

Example (H). If $S = \{1\}$, then $\text{Omit}_n(S) = \{00\cdots0\}$, and $\text{omit}_n(S) = 1$ for all $n \ge 1$. Thus, $C_n(S) = \{1, 00\cdots0\}$ is a cover for $\{0,1\}^n$ of cost $1/2+2^{-n}$, for all $n \ge 1$.

Example (I). If $S = \{10\}$, then for $n \ge 2$, $\text{Omit}_n(S)$ consists of the $n+1$ strings of the form $0^k * 1^{n-k}$ for $0 \le k \le n$. Thus, $\text{omit}_n(S) = n+1$, and $C_n(S)$ is a cover for $\{0,1\}^n$ of cost $1/4 + (n+1)/2^n$, for all $n \ge 2$.

Example (J). If $S = \{100, 1101\}$, then for $n \ge 4$, it can be shown that $\text{omit}_n(S) = 1 + \binom{n}{1} + \binom{n}{2}$ and, thus, $C_n(S)$ is a cover for $\{0,1\}^n$ of cost $3/16 + (1 + \binom{n}{1} + \binom{n}{2})/2^n$ for all $n \ge 4$.

If Theorems (4) and (6) are combined, it is found that if S is an irreducible set of strings of length $\leq n$, then $B_n(C_n(S))$ is a universal deBruijn building block of order n and efficiency $1 - \text{cost}(S) - 2^{-n}\text{omit}_n(S)$. For simplicity, $B_n(C_n(S))$ is denoted by $\bar{B}_n(S)$, and this fact is given as a theorem.

Theorem (7). If S is an irreducible set of strings of length $\leq n$, then the graph $\bar{B}_n(S)$ is a universal deBruijn building block of order n and efficiency $1 - \text{cost}(S) - 2^{-n}\text{omit}_n(S)$.

The following theorem is a partial generalization of Examples (H) and (I).

Theorem (8). Fix $m \geq 1$. If $S_m = \{10^{m-1}\}$, then as $n \to \infty$, $\text{omit}_n(S_m) = O(\alpha_m^n)$, where α_m is the largest positive root of the equation $z^m - 2z^{m-1} + 1 = 0$, which is strictly less than 2. Thus, for all $n \geq m$, $C_n(S)$ is a cover for $\{0,1\}^n$ of cost $2^{-m} + O(\alpha_m/2)^n$, which approaches 2^{-m} as $n \to \infty$.

Proof: According to [2], if m is fixed, the generating function $f_m(z) = \sum_{n \geq 0} \text{omit}_n(S_m) z^n$ is given in closed form by $f_m(z) = 1/(1 - 2z + z^m)$. It follows then from the general theory of rational generating functions (see [4], Theorem 4.1.1), that $\text{omit}_n = O(\beta^n)$, where β is the reciprocal of the smallest positive root of the equation $1 - 2z + z^m = 0$, which is also the largest positive root of the "reciprocal" polynomial $P_m(z) = z^m - 2z^{m-1} + 1$. The largest root of $P_m(z)$ is strictly less than 2, since $P_m(1) = 0$, $P_m(2) = 1$, and $P_m'(z) > 0$ for $z > 2$.

Corollary. If c_n denotes the minimum cost for a cover for $\{0,1\}^n$, then $\lim_{n\to\infty} c_n = 0$.

Proof: Theorem (8) implies that for any $m \geq 1$, $\lim_{n\to\infty} c_n \leq 2^{-m}$.

Remarks. We conjecture, but cannot prove, that $c_n = \Theta(1/n)$. However, McEliece and Swanson, in a forthcoming paper, will show that $c_n = \Omega(1/n)$ and $c_n = O(\log n/n)$. [The latter result is based on a more careful analysis of the type given in Theorem (8).]

In view of the connection between covers for $\{0,1\}^n$ and universal deBruijn building blocks, the Corollary of Theorem (8) implies the following.

Theorem (9). There exist universal deBruijn building blocks whose efficiency is arbitrarily close to 1.

Although Theorem (8) gives an infinite family of reasonably cheap covers for $\{0,1\}^n$, it does not produce the cheapest possible covers for all values of n. Indeed, Table 1 gives the cheapest known covers of $\{0,1\}^n$, for $1 \leq n \leq 10$ and, therefore, also the most efficient universal deBruijn building blocks known, for orders up to 10. In every case, only the *precover* S is given, it being understood that the actual cover is the larger set $C_n(S)$. Notice that for $n \leq 7$, the "$10\cdots0$" construction of Theorem (8) gives the best cover known, while for $8 \leq n \leq 10$ (and presumably for all larger values of n, too) the best cover is considerably more complicated. For $1 \leq n \leq 5$, it is believed that the values in Table 1 are the best possible. For larger values of n, however, improvements may be possible. For $n \geq 8$, the covers described in the table are based on the general "$\{100, 1101\}$" cover described in Example (J). For example, for $n = 8$, $\text{omit}(\{100, 1101\}) = 37$, so that $C_8(\{100, 1101\})$ is a cover for $\{0,1\}^8$ of cost $1/8 + 1/16 + 37/256 = 85/256$. However, by trial and error, it is found that of the 37 strings in $\text{Omit}(\{100, 1101\})$, all but six are covered by $\{010101, 010111, 011111, 0000001, 0000101, 0000111\}$. Thus, if $\{100, 1101\} \cup \{010101, 010111, 011111, 0000001, 0000101, 0000111\}$ is used as a precover, Theorem (6) guarantees a cover of cost $1/8 + 1/16 + 1/64 + 1/64 + 1/64 + 1/128 + 1/128 + 1/128 + 6/256 = 72/256$, as shown in Table 1. (Using $\{10\}$ as a precover for precover for $n = 8$ results in a cover of cost 73/256.)

VI. Hierarchical Building Blocks

It has been seen that the universal deBruijn building blocks described in Theorem (4) can be used to build deBruijn graphs of any size. Surprisingly, however, they can also be used as building blocks for larger universal deBruijn building blocks! This is useful in practice when many chips must be put on several boards, and the boards are then wired together to make the deBruijn graph. The main theorem here is the following.

Theorem (10). Suppose $k \leq n$. If S is irreducible and covers $\{0,1\}^k$, T is irreducible and covers $\{0,1\}^n$, and if S prefixes T, then $B_k(S)$ is a building block for $B_n(T)$. Furthermore,

$$\text{eff}(B_k(S) : B_n(T)) = \frac{1 - \text{cost}(S)}{1 - \text{cost}(T)}$$

Proof: Theorem (5) says that $B_n(S)$ is equal to

$$\bigcup_{A:|A|=n-k} B_k(S, A)$$

and so $B_k(S)$ is a building block for $B_n(S)$. But since every string in T has a prefix in S, then $B_n(S) \subseteq B_n(T)$, so that $B_k(S)$ is also a building block for $B_n(T)$. To calculate $\mathrm{eff}(B_k(S) : B_n(T))$, Theorems (2) and (4) are used:

$$\mathrm{eff}(B_k(S) : B_n(T)) = \frac{V(B_n(T))E(B_k(S))}{V(B_k(S))E(B_n(T))}$$

$$= \frac{2^n 2^{k+1}(1 - \mathrm{cost}(S))}{2^k 2^{n+1}(1 - \mathrm{cost}(T))}$$

$$= \frac{1 - \mathrm{cost}(S)}{1 - \mathrm{cost}(T)}$$

Lemma (3). If S is any set of strings, and if $k \leq n$, then $C_k(S)$ prefixes $C_n(S)$.

Proof: By definition, $C_k(S) = S \cup \mathrm{Omit}_k(S)$ and $C_n(S) = S \cup \mathrm{Omit}_n(S)$. A string $s \in \mathrm{Omit}_n(S)$ is a string of length n with no substring from S. The k-bit prefix of s is a string of length k, which also has no substring from S, and so this prefix is in $\mathrm{Omit}_n(S)$. Thus, every string in $C_n(S)$ is either in S or has a prefix in $\mathrm{Omit}_k(S)$.

Theorem (11). If $k \leq n$, then $\bar{B}_k(S)$ is a building block for $\bar{B}_n(S)$, and

$$\mathrm{eff}(\bar{B}_k(S) : \bar{B}_n(S)) = \frac{1 - \mathrm{cost}(S) - 2^{-k}\mathrm{omit}_k(S)}{1 - \mathrm{cost}(S) - 2^{-n}\mathrm{omit}_n(S)}$$

Proof: The proof follows from Theorem (10) and Lemma (3).

Example (K). Returning to the Big Viterbi Decoder mentioned briefly in Section I: the BVD requires the construction of the deBruijn graph B_{13} using 256 one-chip realizations of the graph $\bar{B}_5(\{10\})$, which, by Theorem (7) and Example (I), is a universal deBruijn building block of efficiency 18/32 (see Fig. 7), and so it contains exactly $64 \cdot \frac{18}{32} = 36$ edges.

According to Theorem (11), $\bar{B}_5(\{10\})$ is a building block for $\bar{B}_9(\{10\})$ and, in the BVD, 16 of the $\bar{B}_5(\{10\})$-chips are wired together on a printed-circuit board to make a $\bar{B}_9(\{10\})$ board. Now $\bar{B}_9(\{10\})$ is a universal deBruijn building block of efficiency 374/512, and so it contains exactly $1024 \cdot \frac{374}{512} = 748$ edges. However, $16 \cdot 36 = 576$ of these edges are internal to the component chips, so that each $\bar{B}_9(\{10\})$ board actually has only $748 - 576 = 172$ printed wires. Finally, since $\bar{B}_9(\{10\})$ has efficiency 374/512 as a deBruijn building block, in order to build B_{13}, there will be $2^{14} \cdot (1 - \frac{374}{512}) = 4416$ *backplane* wires, i.e., wires external to the board. In summary:

Unit type	Number of units	Wires/ unit	Total wires
chip	256	36	9216
board	16	172	2752
backplane	1	4416	4416
			16384

References

[1] O. Collins, F. Pollara, S. Dolinar, and J. Statman, "Wiring Viterbi Decoders (Splitting deBruijn Graphs)," *TDA Progress Report 42-96,* Jet Propulsion Laboratory, Pasadena, California, pp. 93–103, February 15, 1989.

[2] L. J. Guibas and A. M. Odlyzko, "String Overlaps, Pattern Matching, and Nontransitive Games," *J. Comb. Theory* A, vol. 30, pp. 183–208, 1981.

[3] R. J. McEliece, *The Theory of Information and Coding,* Reading, Massachussetts: Addison-Wesley, 1977.

[4] R. Stanley, *Enumerative Combinatorics, Vol. 1,* Monterey, California: Wadsworth and Brooks-Cole, 1986.

[5] J. Statman, G. Zimmerman, F. Pollara, and O. Collins, "A Long Constraint VLSI Viterbi Decoder for the DSN," *TDA Progress Report 42-95,* Jet Propulsion Laboratory, Pasadena, California, pp. 134–142, November 15, 1988.

Table 1. Cheapest covers of $\{0,1\}^n$, for $1 \leq n \leq 10$

n	Cost$\cdot 2^n$	Efficiency	S (*precover*)
1	2	0.000	{1}
2	3	0.250	{1}
3	5	0.375	{1}
4	9	0.438	{1} or {10}
5	14	0.563	{10}
6	23	0.641	{10}
7	40	0.688	{10}
8	72	0.719	{100, 1101, 010101, 010111, 011111, 0000001, 0000101, 0000111}
9	127	0.752	{100, 1101, 0000001, 0101011, 0101111, 0111111, 00001011, 00001111, 01010101}
10	229	0.776	{100, 1101, 01010101, 01010111, 01111111, 000000001, 000000101, 000000111, 000010101, 000010111, 000011111}

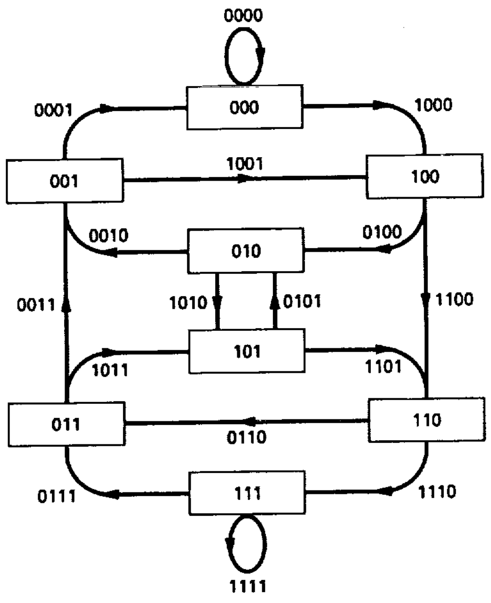

Fig. 1. The deBruijn graph B_3.

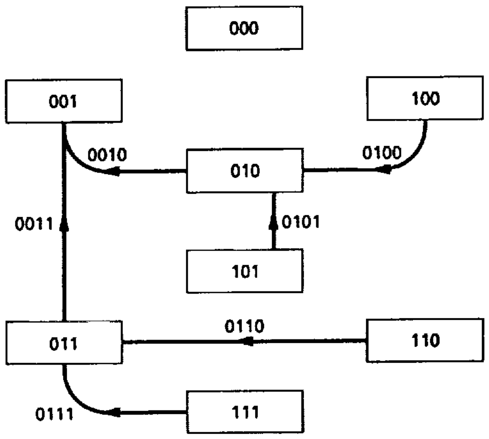

Fig. 3. A B_3 subgraph with density 6/16. In the notation of Section IV, this is the graph $B_3(\{1,000\})$. In the notation of Section V, it is the graph $\overline{B}_3(\{1\})$. It is a universal deBruijn subgraph of efficiency 3/8.

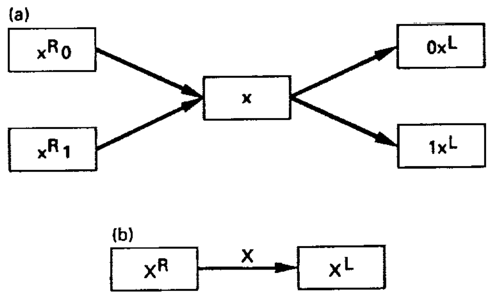

Fig. 2. Two equivalent definitions of the deBruijn graph B_n: (a) four vertices connected to the vertex labelled x (x is an n-bit string), and (b) vertices at the left and right ends of the edge labelled X (X is an $(n+1)$-bit string).

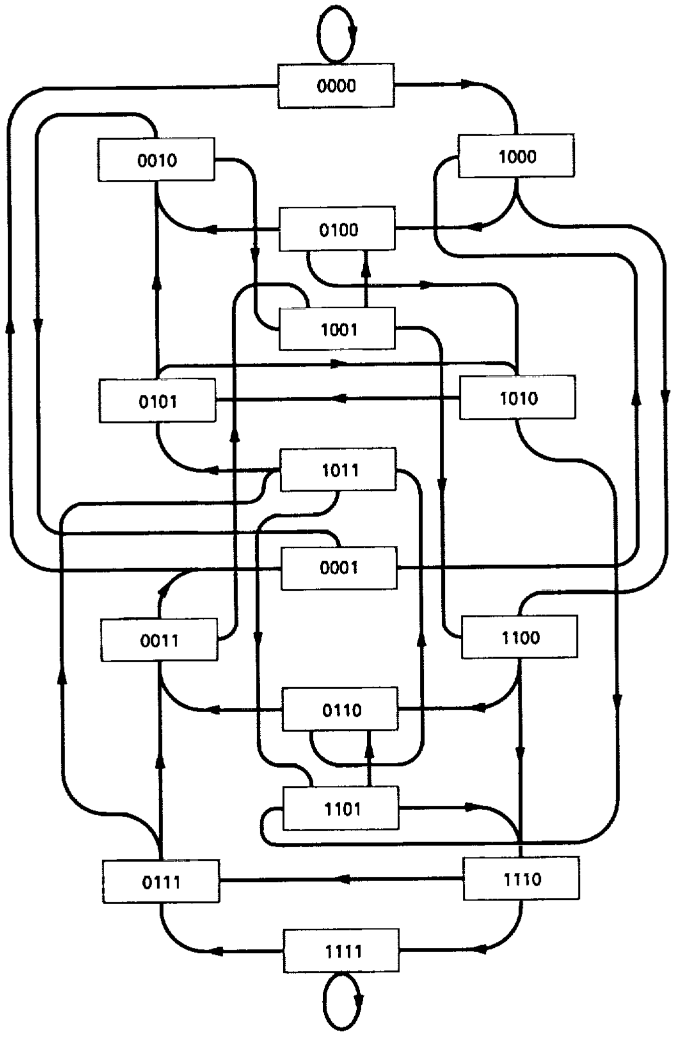

Fig. 4. Two copies of the B_3 subgraph H from Fig. 3, relabelled and wired together to make B_4 (edge labels omitted).

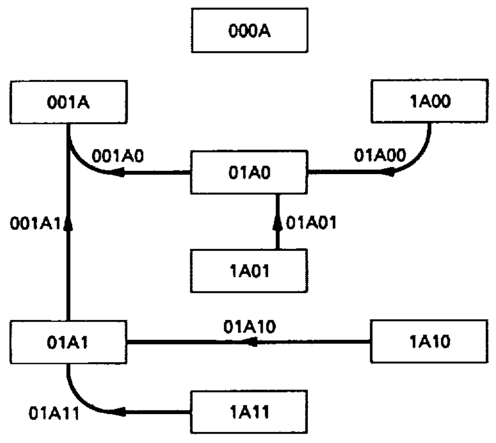

Fig. 5. The graph $B_3(S,A)$, obtained by inserting the symbol A immediately after the first occurrence of a substring from $S = \{1,000\}$ in the corresponding label in $B_3(S)$ (Fig. 3).

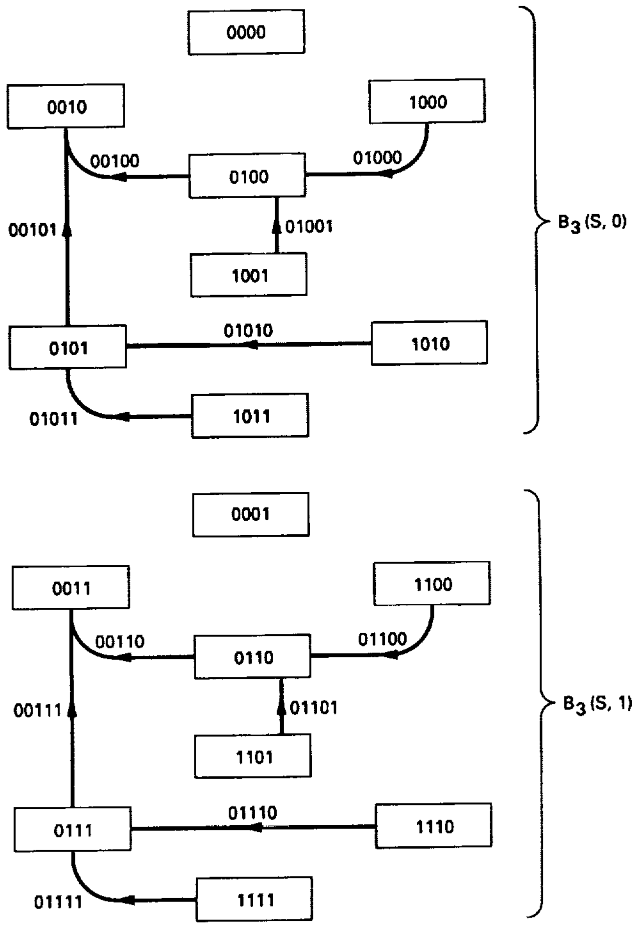

Fig. 6. Illustrating Theorem 4.6: $B_3(S,0) \cup B_3(S,1) = B_4(S)$ (compare with Fig. 4).

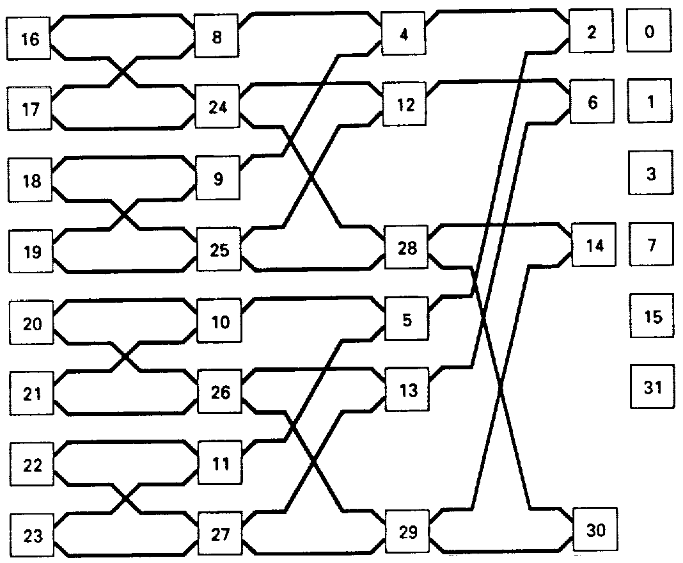

Fig. 7. One possible layout of the $\overline{B}_5(\{10\})$ chip used to build the BVD. All edges are directed from left to right. The vertex labels shown are the decimal equivalents of the actual five-bit binary labels, and the edge labels have been omitted.