Some VLSI decompositions of the de Bruijn graph*

Sam Dolinar, Tsz-Mei Ko and Robert McEliece

Jet Propulsion Laboratory and Department of Electrical Engineering, California Institute of Technology, Pasadena, CA 91125, USA

Received 12 March 1992

Abstract


We define a VLSI decomposition of a directed graph $G$ to be a collection of isomorphic vertex-disjoint subgraphs of $G$ which together contain all of $G$'s vertices. We call the isomorphic subgraphs comprising the decomposition building blocks for the graph $G$, and we refer to the edges contained in the collection of subgraphs as internal edges. The efficiency of a VLSI decomposition of $G$ is the fraction of the total number of edges in $G$ which are internal edges. In this paper we will present a general construction for efficient VLSI decompositions for the family of de Bruijn graphs. Using the methods to be explained in this paper, we have found a 64-chip VLSI decomposition of the de Bruijn graph $B_{13}$ with efficiency 0.754, which is being used by JPL design engineers to build a single-board Viterbi decoder for the $K = 15$, rate $1/4$ convolutional code which will be used on NASA's Galileo mission.

1. Introduction

Let $G$ be a directed graph. We think of $G$ as a representation of the wiring diagram for an electronic circuit, with the vertices of $G$ representing arithmetic processors and the edges of $G$ representing wires connecting the processors. In modern VLSI technology, if the circuit is too large to fit on a single chip, it may be possible to build it by wiring together two or more appropriately designed chips. Each processor must then be placed on one of the chips, but the wires of the circuit may be either internal to the chips (intrachip wires) or external (interchip wires). For reasons of economy and simplicity, it is desirable, though

*The research described in this paper was partially carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration. Tsz-Mei Ko's contribution was also supported by National Security Agency Grant No. MDA904-90-H-1007. Robert McEliece's contribution was also supported by AFOSR grant 91-0037 and a grant from Pacific Bell.
S. Dolinar et al.

not necessary, for the chips to be identical. Returning now to the graph $G$, we are thus motivated to define a **VLSI decomposition** of $G$ as a collection of isomorphic vertex-disjoint subgraphs of $G$ which together contain all of $G$'s vertices, and a subset of its edges. We call the isomorphic subgraphs comprising the decomposition **building blocks** for the graph $G$, and we refer to the edges contained in the collection of subgraphs as **internal** edges. Since internal edges are, as a rule, much more convenient for the circuit designer than external edges, we define the efficiency of a VLSI decomposition of $G$ as the fraction of the total number of edges in $G$ which are internal edges.

In this paper we will present a general construction for efficient VLSI decompositions for the family of de Bruijn graphs (a formal definition of the de Bruijn graph $B_n$ will be given in Section 3). We focus on this class of graphs because they represent the circuit diagrams for fully parallel Viterbi decoders. Indeed, a constraint length $K$, rate $1/n$ convolutional code has the de Bruijn graph $B_{K-2}$ as the circuit diagram for its Viterbi decoder, and NASA is using a $K = 15$, rate $1/4$ convolutional code on the **Galileo** mission. As we will see, the methods described in this paper can be used to design a 64-chip VLSI decomposition of $B_{13}$ with efficiency 0.754, and this chip is being used by JPL design engineers to build a single-board Viterbi decoder for the **Galileo** code. (Earlier, but less efficient, VLSI decompositions for de Bruijn graphs were presented in [1] and [2]. Those led to a 256-chip VLSI decomposition of $B_{13}$ of efficiency 0.563, which was used to design a multi-board decoder for the **Galileo** code.)

As a sample of our results, consider the de Bruijn graph $B_3$ in Fig. 1. It contains 8 vertices and 16 edges. In Fig. 2 we see a two-chip VLSI decomposition of $B_3$, in which the underlying building block contains four vertices and three

---

Fig. 1. The de Bruijn graph $B_3$.

Fig. 2. A two-chip VLSI decomposition of $B_3$, with efficiency 0.375.
Some VLSI decompositions of the de Bruijn graph

edges. This decomposition has efficiency 3/8. It turns out that the building block in Fig. 2 is a universal building block, in the sense that it can be used to build any de Bruijn graph $B_n$ with $n \geq 3$. Indeed this building block is just one of a large family of universal building blocks we will describe in Theorem 3.1.

Here is a summary of the rest of the paper. In Section 2 we will introduce some algebraic notation and prove two simple lemmas that will be needed in the proof of our main theorem. In Section 3 we will give a formal definition of the de Bruijn graph $B_n$ and show that subgraphs of $B_n$ defined by certain rank functions are universal de Bruijn building blocks. In Section 4 we will use the results of Section 3 to exhibit a 4-chip VLSI decomposition of $B_5$ of efficiency 0.50. Finally, in Section 5 we will list the most efficient universal building blocks we know of (including, in Fig. 8, a description of the building block which is being used on the one-board Viterbi decoder for the Galileo code), and state some asymptotic results.

2. Algebraic preliminaries

Let $V_n$ be the set of all $n$-dimensional binary vectors. We begin by defining three linear mappings $L$, $R$, and $C$ ('left', 'right', and 'center') from $V_n$ to $V_{n-1}$. (Technically, the mappings $L$, $R$, and $C$ are each families of mappings, one for each $n \geq 2$.) If $x = [x_1, \ldots, x_n]$ is a binary vector of length $n$, then

$Lx = (x_1, \ldots, x_{n-1}),$

$Rx = (x_2, \ldots, x_n),$

$Cx = (x_1 + x_2, \ldots, x_{n-1} + x_n).$

For example, if $x = [10110]$, then $Lx = [1011]$, $Rx = [0110]$, and $Cx = [1101].$

**Lemma 2.1.** The mappings $L$ and $R$ commute with $C$, i.e., $CLx = LCx$ and $CRx = RCx$ for any binary vector $x$ of length $\geq 3$.

**Proof.** By direct computation we find that if $x = [x_1, \ldots, x_n]$, then

$CLx = LCx = [x_1 + x_2, \ldots, x_{n-2} + x_{n-1}],$

$CRx = RCx = [x_2 + x_3, \ldots, x_{n-1} + x_{n}].$  \[\square\]

Now we define the burst agreement $B(x, y)$ between two binary $n$-vectors $x$ and $y$ as the length of the largest block of consecutive components on which $x$ and $y$ agree. For example if $x = [11010010]$ and $y = [01110001]$, then $B(x, y) = 3$ because $x$ and $y$ agree in positions 4, 5, and 6, but in no set of four or more consecutive positions.

**Lemma 2.2.** If $x$ and $y$ are two $n$-vectors with $C'x = C'y$, and $B(x, y) \geq r$, then $x = y$. 
Proof. We use induction on \( r \). For \( r = 1 \), the assertion is that if \( Cx = Cy \), and if \( x \) and \( y \) agree in at least one coordinate, then \( x \) and \( y \) are identical. To see that this is so, note that \( C \) is a linear mapping from \( V_n \) to \( V_{n-1} \). Its nullspace, i.e., the set of \( x \)'s such that \( Cx = 0 \), is the set of vectors \([x_1, \ldots, x_n]\) such that \( x_1 + x_2 = x_2 + x_3 = \cdots = x_{n-1} + x_n = 0 \). This set contains only the two vectors \([00 \cdots 0]\) and \([11 \cdots 1]\). Thus if \( Cx = Cy \), then either \( x = y \) or \( x = y + [11 \cdots 1] \), i.e., \( x \) and \( y \) differ in all \( n \) positions. It follows that if \( Cx = Cy \) and if \( x \) and \( y \) agree in at least one place, then \( x = y \). This completes the proof for \( r = 1 \).

We now assume \( r \geq 2 \), and that the lemma has been proved for all \( r' < r \). If \( B(x, y) \geq r \), i.e., if \( x \) and \( y \) agree on \( r \) consecutive positions, then clearly \( Cx \) and \( Cy \) agree on at least \( r - 1 \) consecutive positions. Thus if we let \( x' = Cx \) and \( y' = Cy \), then \( B(x', y') \geq r - 1 \). Also, the hypothesis \( C'x = C'y \) is equivalent to \( C^{-1}x' = C^{-1}y' \). Thus by the induction hypothesis, \( x' = y' \), i.e., \( Cx = Cy \). But also \( B(x, y) \geq r \geq 1 \), so that by the \( r = 1 \) case of the lemma, which has already been proved, \( x = y \). \( \square \)

3. A construction for universal de Bruijn building blocks

The de Bruijn graph \( B_n \) can be defined as the directed graph with \( V_n \) as vertex set, and with a directed edge from \( x \) to \( y \) if and only if \( Lx = Ry \). (See Golomb [3, Sections 2.2 and 6.2] for more about the de Bruijn graph.)

Now let \( \rho \) be a mapping from \( V_n \) to the set \( \{0, 1, \ldots, n\} \). For \( x \in V_n \), we call \( \rho(x) \) the rank of \( x \). We define the \( \rho \)-subgraph of \( B_n \), denoted by \( B_n(\rho) \), as follows. The vertex set of \( B_n(\rho) \) is \( V_n \), and \( x \) and \( y \) are connected by a directed edge in \( B_n(\rho) \) if and only if \( (a) \) \( x \) and \( y \) are connected by a directed edge in \( B_n \), and \((b) \) \( \rho(y) = \rho(x) + 1 \). For example, consider the rank mapping \( \rho : V_3 \rightarrow \{0, 1, 2, 3\} \) described in Table 1. The corresponding graph \( B_3(\rho) \) is shown in Fig. 3.

It turns out, rather surprisingly, that any graph of the form \( B_n(\rho) \) can be used as a building block for any de Bruijn graph \( B_N \) with \( N \geq n \). For this reason we call

<table>
<thead>
<tr>
<th>Table 1</th>
<th>A rank function on the vertices of ( B_n ). The corresponding graph ( B_n(\rho) ) is shown in Fig. 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x )</td>
<td>( \rho(x) )</td>
</tr>
<tr>
<td>000</td>
<td>3</td>
</tr>
<tr>
<td>001</td>
<td>2</td>
</tr>
<tr>
<td>010</td>
<td>1</td>
</tr>
<tr>
<td>011</td>
<td>2</td>
</tr>
</tbody>
</table>
the graph $B_n(\rho)$ a universal de Bruijn building block (UBB). Theorem 3.1, which follows, spells this out.

**Theorem 3.1.** The graph $B_n(\rho)$ is a UBB, i.e., it can be used to build any de Bruijn graph $B_N$ with $N \geq n$.

**Proof.** We will show that $B_n(\rho)$ builds $B_{n+r}$ for all $r \geq 1$. For any $X = [X_1, X_2, \ldots, X_{n+r}] \in V_{n+r}$, suppose that $C'X = x \in V_n$, and $\rho(x) = i$. We define the r-bit chip number of $X$, denoted by $\text{num}(X)$, as

$$\text{num}(X) = [X_{i+1}, \ldots, X_{i+r}].$$

(3.1)

Note that since $0 \leq i \leq n$, then $1 \leq i + 1 \leq i + r \leq n + r$, so that the chip number as defined in (3.1) 'fits' within the field of $X$. In building $B_{n+r}$, with $2^r$ copies of $B_n(\rho)$ ('chips'), numbered $00 \cdots 0$ to $11 \cdots 1$, we place vertex $X$ on the chip numbered $\text{num}(X)$, at the location corresponding to $x = C'X$. Lemma 2.2 shows that no two vertices of $B_{n+r}$ can be assigned the same location on the same chip, so that each of the $2^{n+r}$ vertices in $B_{n+r}$ is assigned a unique 'home' on one of the $2^r$ chips. What remains to show is that the connections within the chips correspond to connections in the big graph $B_{n+r}$, i.e., that if $\text{num}(X) = \text{num}(Y)$ and if $C'X$ and $C'Y$ are connected on $B_n(\rho)$, then $X$ and $Y$ are connected in $B_{n+r}$, i.e., $LX = RY$.

To see this, we reason as follows. Since $C'X$ and $C'Y$ are connected on $B_n(\rho)$, then $\rho(C'Y) = \rho(C'X) + 1$. Thus if $\rho(C'X) = i$, then $\rho(C'Y) = i + 1$, and so, since $\text{num}(X) = \text{num}(Y)$, we have

$$[X_{i+1}, \ldots, X_{i+r}] = [Y_{i+2}, \ldots, Y_{i+r+1}].$$

Thus $LX$ and $RY$ agree on $r$ consecutive positions, i.e.,

$$B(LX, RY) \geq r.$$  

(3.2)
But also, since $C'X$ and $C'Y$ are connected on $B_n(\rho)$, we have $LC'X = RC'Y$, which, by Lemma 2.1, implies

$$C'LX = C'RY, \quad (3.3)$$

Combining (3.2) and (3.3), using Lemma 2.2, we find that $LX = RY$, which is what we set out to prove. \(\square\)

The following theorem, whose proof may be found in [5], gives a partial converse to Theorem 3.1. To state it, we introduce the notion of a graded digraph. A digraph $G$ with vertex set $V$ is graded of rank $m$ if there is a rank function $\rho : V \rightarrow \{0, 1, \ldots, m\}$, such that $\rho(y) = \rho(x) + 1$ if there is a directed edge from $x$ to $y$. Thus $B_n(\rho)$ is a graded subgraph of $B_n$ of rank $n$.

**Theorem 3.2.** If $G$ is a subgraph of $B_n$ with $2^n$ vertices, and which builds all de Bruijn graphs $B_N$ for $N \geq N_0$, then $G$ is a graded subgraph of $B_n$.

Combining Theorems 3.1 and 3.2, we see that a necessary condition for a subgraph of $B_n$ to be a UBB is that it be graded; and a sufficient condition is that it be graded of rank $\leq n$. Most graded subgraphs of $B_n$ with rank exceeding $n$, seem to be UBBs; however, there is a graded subgraph of $B_4$ of rank 5 which is not a UBB, so the whole story is apparently quite complicated.

4. Example

We illustrate the construction in Theorem 3.1, by building the graph $B_5$ with four copies of the graph $B_5(\rho)$ in Fig. 3. We begin with Table 2, which lists, for each of the 32 possible 5-bit vectors $X$, the 3-bit vector $x = C^5X$, and the corresponding rank $\rho(x)$.

We number the four copies of $B_5(\rho)$ 00, 01, 10 and 11. Table 2 can be used to find the chip number and the location within a chip of each 5-bit vector $X$, as follows. For a given $X$, the value $x = C^5X$ gives the location, and the two bits of $X$ in positions $\rho(x) + 1$ and $\rho(x) + 2$, which are underlined in the table, give the chip number. For example, consider $X = 11000$. According to the table, $x = 110$, $\rho(x) = 1$, and the underlined bits are 10. Thus $X$ must be placed in location 110 in chip number 10. The complete assignment of vertices of $B_5$ to the four chips is shown in Fig. 4.

5. The most efficient known UBBs

In Section 3 we saw that any subgraph of $B_n$ induced by a rank $n$ function is a UBB. The efficiency $e_n$ of a UBB is independent of the size of the de Bruijn
graph which it is used to build. Indeed, if there are \( E_n \) edges in \( B_n(\rho) \) and \( 2^r \) copies of \( B_n(\rho) \) are used to build \( B_{n+r} \), then there are a total of \( 2^rE_n \) internal edges out of \( 2^{n+r+1} \) total edges in \( B_{n+r} \). Thus the efficiency is
\[
e_n = \frac{2^rE_n}{2^{n+r+1}} = \frac{E_n}{2^{n+1}}, \tag{5.1}
\]

Table 2
A table for building \( B_r \) from 4 copies of the graph in Fig. 3

<table>
<thead>
<tr>
<th>( X )</th>
<th>( x = C^2X )</th>
<th>( \rho(x) )</th>
<th>( X )</th>
<th>( x = C^2X )</th>
<th>( \rho(x) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>00000</td>
<td>000</td>
<td>3</td>
<td>10000</td>
<td>100</td>
<td>0</td>
</tr>
<tr>
<td>00001</td>
<td>001</td>
<td>2</td>
<td>10001</td>
<td>101</td>
<td>0</td>
</tr>
<tr>
<td>00010</td>
<td>010</td>
<td>1</td>
<td>10010</td>
<td>110</td>
<td>1</td>
</tr>
<tr>
<td>00011</td>
<td>011</td>
<td>2</td>
<td>10011</td>
<td>111</td>
<td>1</td>
</tr>
<tr>
<td>00100</td>
<td>100</td>
<td>0</td>
<td>10100</td>
<td>001</td>
<td>2</td>
</tr>
<tr>
<td>00101</td>
<td>101</td>
<td>0</td>
<td>10101</td>
<td>000</td>
<td>3</td>
</tr>
<tr>
<td>00110</td>
<td>110</td>
<td>1</td>
<td>10110</td>
<td>011</td>
<td>2</td>
</tr>
<tr>
<td>00111</td>
<td>111</td>
<td>1</td>
<td>10111</td>
<td>010</td>
<td>1</td>
</tr>
<tr>
<td>01000</td>
<td>010</td>
<td>1</td>
<td>11000</td>
<td>110</td>
<td>1</td>
</tr>
<tr>
<td>01001</td>
<td>011</td>
<td>2</td>
<td>11001</td>
<td>111</td>
<td>1</td>
</tr>
<tr>
<td>01010</td>
<td>000</td>
<td>3</td>
<td>11010</td>
<td>100</td>
<td>0</td>
</tr>
<tr>
<td>01011</td>
<td>001</td>
<td>2</td>
<td>11011</td>
<td>101</td>
<td>0</td>
</tr>
<tr>
<td>01100</td>
<td>110</td>
<td>1</td>
<td>11100</td>
<td>011</td>
<td>2</td>
</tr>
<tr>
<td>01101</td>
<td>111</td>
<td>1</td>
<td>11101</td>
<td>010</td>
<td>1</td>
</tr>
<tr>
<td>01110</td>
<td>110</td>
<td>1</td>
<td>11110</td>
<td>001</td>
<td>2</td>
</tr>
<tr>
<td>01111</td>
<td>111</td>
<td>0</td>
<td>11111</td>
<td>000</td>
<td>3</td>
</tr>
</tbody>
</table>

4. Four copies of the graph of Fig. 3 labelled to form a subgraph of \( B_r \). This is a four-chip VLSI decomposition of \( B_r \) of efficiency 0.50.
Table 3
The most efficient known universal de Bruijn building blocks, for $1 \leq n \leq 8$. The entries for $n = 4$ are known to be optimal. For larger values of $n$, improvements may be possible. (In the table $E_n$ denotes the number of edges, $e_n$ is the efficiency defined in (5.1), and $c_n$ is defined in (5.2.).)

<table>
<thead>
<tr>
<th>$n$</th>
<th>$E_n$</th>
<th>$e_n$</th>
<th>$c_n$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0.250</td>
<td>1.500</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>0.375</td>
<td>1.875</td>
</tr>
<tr>
<td>3</td>
<td>8</td>
<td>0.594</td>
<td>2.000</td>
</tr>
<tr>
<td>4</td>
<td>19</td>
<td>0.672</td>
<td>1.969</td>
</tr>
<tr>
<td>5</td>
<td>43</td>
<td>0.719</td>
<td>1.969</td>
</tr>
<tr>
<td>6</td>
<td>92</td>
<td>0.754</td>
<td>1.969</td>
</tr>
<tr>
<td>7</td>
<td>193</td>
<td>0.777</td>
<td>2.004</td>
</tr>
<tr>
<td>8</td>
<td>398</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 3 lists the number of edges in the most efficient UBBs we have been able to find, using ad hoc methods, for $1 \leq n \leq 8$. We have been able to show by exhaustive search that the entries for $1 \leq n \leq 4$ are optimal, but for larger values of $n$, improvements may be possible. For $n = 1$, an optimal building block consists of a single edge. The optimal universal building block for $n = 2$ is shown in Fig. 2 (to see that the chips in Fig. 2 do indeed correspond to subgraphs of $B_2$, just apply the operator $C$ to them). An optimal UBB for $n = 3$ has already been shown in Fig. 3. The best known UBBs for $n = 4$ through $n = 7$ are shown in Figs. 5 through 8. In these figures, the binary strings are represented by their decimal equivalents.

If $e_n^*$ denotes the maximum possible efficiency of a $B_n(p)$ graph, it is natural to wonder about the asymptotic behavior of $e_n^*$ as $n \to \infty$. Thanks to Eric Schwabe

![Fig. 5. The most efficient $B_4(p)$ building block.](image)

The edge count is 19.

![Fig. 6. The most efficient known $B_4(p)$ building block.](image)

The edge count is 43.

1 The building blocks for $n = 5$ and $n = 8$ were first discovered by a clever computer search algorithm developed by Gordon Oliver.
Some VLSI decompositions of the de Bruijn graph

Fig. 7. The most efficient known $B_{x}(p)$ building block. The edge count is 92.

Fig. 8. The most efficient known $B_{x}(p)$ building block. The edge count is 193.
[4], we know quite a lot about this. If we define the quantity $c_n^*$ by the formula

$$e_n^* = 1 - \frac{c_n^*}{n + 1},$$  \hspace{1cm} (5.2)$$

then Schwabe has proved that

$$A \leq \liminf_{n \to \infty} c_n^* \leq \limsup_{n \to \infty} c_n^* \leq B,$$  \hspace{1cm} (5.3)$$

where $A$ and $B$ are positive constants with $A \geq 1$ and $B \leq 8$. Thus the efficiency of the best UBB behaves like $1 - K/n$ as $n \to \infty$. However, if we look at Table 3, where $c_n$ for the best known chips is tabulated for $1 \leq n \leq 8$, something much stronger seems to be true, and we end our paper with a conjecture.

**Conjecture.** If $c_n^*$ is as defined above, then $\lim_{n \to \infty} c_n^* = 2$.

**References**


