Wiring Viterbi Decoders (Splitting deBruijn Graphs)

O. Collins, F. Pollara, S. Dolinar, and J. Statman
Communications Systems Research Section

A new Viterbi decoder, capable of decoding convolutional codes with constraint lengths up to 15, is under development for the DSN. A key feature of this decoder is a two-level partitioning of the Viterbi state diagram into identical subgraphs. The larger subgraphs correspond to circuit boards, while the smaller subgraphs correspond to VLSI chips. The full decoder is built from identical boards, which in turn are built from identical chips. The resulting system is modular and hierarchical. The decoder is easy to implement, test, and repair because it uses a single VLSI chip design and a single board design. The partitioning is completely general in the sense that an appropriate number of boards or chips may be wired together to implement a Viterbi decoder of any size greater than or equal to the size of the module.

I. Introduction

A new Viterbi decoder [1], capable of decoding convolutional codes with constraint lengths up to 15, is under development for the DSN. This article describes a novel partitioning of the decoder's state transition diagram that forms the basis for the new decoder's architecture.

The Viterbi algorithm is naturally fully parallel [2]. However, a fully parallel implementation of a large constraint length Viterbi decoder requires an impractical amount of hardware. The first question to be faced when building such a decoder is what part of this parallelism to throw away. We decided to retain a fully distributed architecture for computing and exchanging accumulated metrics, but to perform the arithmetic computations bit-serially. The arithmetic computations are 16-bits long, so the decoding speed will be greater than 1 Mbit/sec with a 20-MHz system clock.

In a fully distributed architecture, there are 2^{K-1} basic computational elements called *add-compare-select* circuits [1] for a constraint length K decoder. When K is large, it is desirable to take a modular, hierarchical approach to organizing the huge number of required elements. Many add-compare-select circuits can be implemented on a single VLSI chip, and many chips can be mounted on a single printed circuit board. The full decoder is implemented by wiring together the required number of chips and boards.

The main problem is wiring. How can the 2^{K-1} basic elements, each with two inputs and an output (going to two different elements' inputs), be partitioned into chips and boards without using too many pins per chip or too large a board edge connector? This article shows first how pairs of add-compare-select circuits group to form elements called butterflies. The connection diagram of these 2^{K-2} butterflies is a deBruijn

graph [3]; the butterflies are nodes in the graph and the edges of the graph represent wires between butterflies. The rest of the article shows how the set of butterflies can be split into modules called boards and the boards split into modules called chips, in such a way that a large proportion of the required connections between butterflies are implemented internally within the modules. The chips are all identical and the boards are all identical. Furthermore, their internal structure does not depend on the size of the decoder, and an appropriate number of board modules and chip modules can be wired together to make a decoder of any size equal to or greater than that of the smallest module.

The constraint length 15 Viterbi decoder under development for the DSN is currently being designed with 16 boards and 512 chips. Each chip in this design contains 16 butterflies, and each board has 32 chips. However, the theory developed in this article is completely general and produces a modular, hierarchical partitioning of any size deBruijn graph into any number of first-level and second-level subgraphs (boards and chips). The exposition of the theory and the examples in this article are selected without reference to a specific configuration of the DSN's new decoder.

II. Butterflies and deBruijn Graphs

All 2^{K-1} states in a constraint length K Viterbi decoder are labeled with $(K-1)$-bit binary strings. An add-compare-select circuit takes as inputs the accumulated metrics of two states whose labels differ only in the rightmost bit. Each of these accumulated metrics has a different branch metric added to it and the smaller of the two sums is selected.

Two add-compare-select circuits take inputs from the same pair of states. The output of one of these goes to a state obtained by discarding the rightmost bit of the input states and prefixing a *0* on the left. The output of the other add-compare-select circuit goes to the state defined similarly but with a prefixed *1* instead of *0*. These two add-compare-select circuits group to form a *butterfly*, depicted in Fig. 1. The butterfly has two input wires and two output wires for transmission of accumulated metrics. The butterfly needs only four wires, because its two add-compare-select circuits get their inputs from the same pair of states. Also, it can be shown [4] that a butterfly's two add-compare-select circuits can share the same hardware for computing branch metrics. These facts make butterflies natural elements to work with.

A butterfly is labeled by dropping the rightmost bit of the label of either of its input states. The butterfly connection diagram is a deBruijn graph with 2^{K-2} nodes. Each node in this graph is labeled by a $(K-2)$-bit binary string and each edge is labeled by a $(K-1)$-bit binary string.[1] Each node is connected to four other nodes via four directed edges. A node receives its inputs via the pair of edges obtained by appending a *0* or *1* to the right of the node's label, and it sends its outputs via the pair of edges obtained by prefixing a *0* or *1* to the left of the node's label. A diagram of the connections for an arbitrary butterfly is given in Fig. 2.

III. Wiring Approaches

The full deBruijn graph of 2^{K-2} butterflies requires exactly 2^{K-1} wires for the exchange of accumulated metrics. This total number of connections cannot be increased or reduced by any wiring scheme. However, it is advantageous to capture as many of these required connections as possible within identical, small, modular units (chips and boards). Wires internal to modules can be implemented by duplicating the small module's simple wiring diagram, while external wires between modules must be implemented wire-by-wire.

One mathematically appealing way of creating identical modular units that incorporate a reasonable proportion of internal wires is to exploit one of the Hamiltonian paths [3] of the deBruijn graph. One of the two outputs of each butterfly is connected to one of the two inputs of another butterfly in a big ring (Fig. 3a). This ring contains all of the butterflies and half of their connections. The remaining half of the connections form an irregular pattern across the interior of the ring, as illustrated in Fig. 3(a). Identical modules can be constructed by slicing the Hamiltonian ring into equal-size *linear* segments (Fig. 3b). Almost half of the wires required for accumulated metric exchange can be implemented internally within the modules.

A second wiring approach is based on FFT-type connection patterns. Modules (chips and boards) are constructed from disjoint subsets of butterflies called *roots*. Each module contains its root butterflies, first-generation descendants of these roots, descendants of these descendants, and so forth. The descendants of a butterfly are the two butterflies to which it sends its outputs. The module contains all descendants at each generation except those that are roots of another module.

If a set of 2^b root butterflies is consecutive in the last b bits (i.e., the last b bits take on all possible values and all other bits are the same), then their descendants through b generations are a block of butterflies obtained by cyclic shifting the roots by b bits or less. A module containing the roots and all of

[1] In the remainder of this article, the terms *butterfly*, *node*, *butterfly label*, and *node label* will be used interchangeably, as will the terms *state*, *wire*, *edge*, *state label*, and *edge label*.

these descendants would have $(b + 1)2^b$ butterflies and the same connection pattern as an ordinary signal processing FFT of $(b + 1)$ stages, as shown in Fig. 4. This cyclic shifting may, however, generate one of the input strings. Unfortunately, it is impossible to completely partition any deBruijn graph into non-overlapping full-FFT modules. A module's connection diagram must be punctured at those nodes corresponding to root nodes of another module. The result is a *crenellated-FFT* connection pattern, a subgraph of a full FFT.

If the root butterflies are selected wisely, most of the full decoder's 2^{K-2} butterflies are found in some module's crenellated-FFT diagram. However, some butterflies do not belong to any crenellated FFT. These butterflies are *free* in the sense that their wiring is not specified by the crennellated-FFT construction. The free butterflies must physically reside within modules, but their connections to other butterflies must be implemented by external wiring (outside the modules), or else the modules' internal wiring would not be identical.

A module based on the crenellated-FFT construction thus contains two types of butterflies. The majority of butterflies belong to a crenellated-FFT pattern, and some or all of their required connections are implemented by internal wiring (within the module) which is identical from module to module. The remaining free butterflies typically have no internal connections, but instead communicate via four external pins (two for input and two for output). The pin reductions which free-butterfly interconnections make possible are trivial.

For the DSN's new Viterbi decoder, the set of root butterflies is taken to be the set of all 2^{K-4} butterflies having the common prefix *10*. This selection of root butterflies works well (i.e., captures a large fraction of wires within modules) for module sizes from 2^4 to about 2^9 butterflies.[2] The full block of root butterflies is subdivided into consecutive blocks of roots for board modules, which are further subdivided into consecutive blocks of roots for the chip modules on each board. The crenellated FFTs generated from these root butterflies are hierarchical in the sense that the crenellated FFT for the board is constructed without breaking any of the connections in the crenellated FFTs for the chips on the board.

A single shift of a string having *10* as a prefix cannot produce another string having *10* as a prefix. Hence, for modules constructed from $B_0 = 2^b$ consecutive root butterflies with the prefix *10*, the number B_1 of first-generation descendants in the crenellated FFT equals the number of roots B_0. The number of butterflies B_g in each succeeding generation, g, of the crenellated FFT is given by the linear recurrence

$$B_g = B_{g-1} - \frac{B_{g-2}}{4}$$

for $2 \leqslant g \leqslant b = \log_2 B_0$. The module only contains descendants through the bth generation; $(b + 1)$th-generation descendants cannot be included because their parent nodes belong to two different modules. It can be shown by evaluating the recursion formula that the number of free butterflies is $b + 3$ and the total number of butterflies in the module (free butterflies plus butterflies in the crenellated FFT) is 2^{b+2} or four times the number of roots. The number of external wires[3] leading off the module is $2^{b+2} + 4(b + 3)$, an average of $1 + (b + 3)2^{-b}$ external wires per butterfly on the module.

Figure 5 shows the connection diagram for a 32-butterfly chip module based on roots with the prefix *10*. The crenellated FFT is on the left and the six free butterflies on the right have all their wires leading off chip. The crenellated FFT for the chip starts with eight root butterflies and continues for three generations of descendants from these roots. The crenellated FFT resembles an ordinary 8 × 4-stage FFT, except for punctures eliminating six of the nodes. The number of external wires per 32-butterfly chip is 56.

Figure 6 shows the connection pattern for a 512-butterfly board module based on roots with the prefix *10*. The crenellated FFT contains 128 roots and 7 generations of descendants. The 128 × 8-stage ordinary FFT template is obvious, even though over half the nodes from this template are missing in the crenellated version. The crenellated-FFT structure includes 502 of the board module's 512 butterflies, leaving just 10 free butterflies per board. The number of external wires per 512-butterfly board is 552, just over 1 wire per butterfly (about half as many external wires as for a same-size module based on the Hamiltonian path construction).

Figures 5 and 6 illustrate how the definition of the first-level subgraph (a board) is completely consistent with the definition of the second-level subgraph (a chip). The 512-butterfly board in Fig. 6 is built from sixteen of the 32-butterfly chips in Fig. 5. In Fig. 6 arrows correspond to chip pins, and unconnected arrows represent board pins (which must be connected to pins on other boards via the backplane). Heavy lines represent wires on the board between chip pins,

[2] *01* would have been an equally good choice, but not *00* or *11*. Other prefixes (such as *100*) or combinations of prefixes (such as *100, 1101*) work better for larger modules. A full discussion of the efficiencies of various root selections is beyond the scope of this article.

[3] External wire and pin counts quoted in this article refer only to the wires required for exchange of accumulated metrics and do not include additional wires and pins needed for power and so forth.

and thin lines represent internal connections within the chip. Pictorially, the crenellated-FFT portions of eight of the sixteen chips in Fig. 6 are identical copies of the crenellated-FFT portion of the chip in Fig. 5, and the crenellated-FFT portions of the other eight chips are depicted by their mirror images (for convenience of display). Similarly, the depictions of the six free butterflies in each chip are displaced horizontally by varying amounts to emphasize the crenellated-FFT structure of the board.

The hierarchical nature of the crenellated-FFT construction holds not just for 32-butterfly chips and 512-butterfly boards but also for all other module sizes 2^{b+2}. Each module constructed from consecutive roots with the prefix *10* can be built from two modules half its size constructed from the same type of roots.

IV. Butterfly Addressing

Each butterfly, described by a $(K-2)$-bit binary string, must be assigned a $(K-2)$-bit address or location. The full address specifies the butterfly's exact position in the modular hierarchy. The most significant bits of the address correspond to the butterfly's board and chip location. For example, in a 2^4-board/2^8-chip configuration for a constraint length 15 decoder (2^{13} total butterflies), the four most significant bits of the address specify the board, and the next four bits specify the chip within a board. The five least significant bits of the address specify the position of the butterfly within a chip.

The addressing formula is somewhat arbitrary, but it must satisfy two basic conditions: (1) it must be a one-to-one mapping from $(K-2)$-bit butterflies to $(K-2)$-bit addresses and (2) it must be consistent with the partition of the deBruijn graph into crenellated FFTs, i.e., all butterflies assigned to certain chip and board locations by the crenellated-FFT construction should be mapped to those same locations by the addressing formula. Free butterflies may be mapped to any convenient free address.

The specification of a butterfly's $(K-2)$-bit address proceeds as follows. First, compute the butterfly's *partial address* by dropping from its $(K-2)$-bit label all of the most significant bits through and including the first occurrence of the string *10*. The partial address consists of all the bits to the right of the first *10*, and it is empty if there is no occurrence of *10* in the butterfly's $(K-2)$-bit label or if *10* first occurs in the two least significant bits. The partial address is the only part of the full address that is specified by the crenellated-FFT partition. For example, in a 2^8-chip decoder, a partial address of 8 bits will determine exactly which chip a given butterfly belongs to, but a butterfly with a partial address of 7 bits or less is one of the free butterflies that is not assigned to any chip's crenellated FFT.

The partial address sets the most significant bits of a butterfly's full address. The remaining part of the address, called *arbitrary bits*, is completely arbitrary in the sense that any choice will be consistent with the crenellated-FFT construction. However, the arbitrary bits for all butterflies must be chosen in a way that assigns each $(K-2)$-bit butterfly to a unique $(K-2)$-bit address. One simple rule for guaranteeing a one-to-one mapping is to choose the arbitrary bits as the reversal of the most significant bits (through and including the first occurrence of *10*) that were dropped to extract the partial address. Then,

$$\begin{aligned} butterfly &= (prefix, partial\ address) \\ &= (\rho(suffix), partial\ address) \end{aligned}$$

$$\begin{aligned} address &= (partial\ address, suffix) \\ &= (partial\ address, \rho(prefix)) \end{aligned}$$

where *suffix* are the arbitrary bits and *prefix* are the most significant bits of *butterfly* up to and including the first occurrence of *10*. The notations $\rho(prefix)$ and $\rho(suffix)$ denote the reversals of the indicated bit strings. For example, $butterfly = (abcde10, fghijk)$ gives $address = (fghijk, 01edcba)$, assuming that *abcde* does not contain the string *10*.

This rule produces a one-to-one mapping because it is obviously invertible. Given any $(K-2)$-bit address, first determine the partial address by dropping all of the least significant bits through and including the last occurrence of *01*. The dropped bits are the arbitrary bits. Now compute the unique butterfly label corresponding to that address by concatenating the reverse of the arbitary bits with the partial address.

V. Making Full Decoders from Chips and Boards

The board and chip modules defined by the crenellated-FFT construction have the property that full Viterbi decoders of all sizes at least equal to the size of the module can be constructed by appropriately connecting identical copies of the module, without revising the internal wiring within any module. Figure 7 shows a 32-butterfly chip wired as a constraint length 7 decoder, and Fig. 8 shows two 32-butterfly chips wired as a constraint length 8 decoder. Arrows correspond to chip pins and heavy lines represent external wires between chip pins. Thin lines represent internal connections within the chip. Note that many of the heavy lines in Fig. 8 connect butterflies within the same chip, as do all the heavy

lines in Fig. 7. However, these connections cannot be incorporated internally within the chip, because the chip would no longer be a universal module, i.e., some larger constraint length decoder could not be built from the more tightly wired chips.

VI. Bounds, Improvements, and Further Applications

There exist lower bounds [4] on the number of edges crossing cuts which divide the nodes of a deBruijn graph into sets of equal or almost equal cardinality. These follow from the very small number of short cycles in the graph and do not depend on the sets having identical internal connections. The present board design is less than a factor of two away from these bounds.

Chip and board modules may include some additional internal connections if they are destined only for a particular size of decoder (e.g., just the constraint length 15 decoder). Also, by restricting the decoder to constraint lengths 15 and larger and allowing one of the boards to be different from the others, the number of wires between boards can be reduced without changing the chips. These facts offer some flexibility if the backplane presents unexpected wiring problems.

There are additional applications for these results unrelated to building Viterbi decoders. For example, the modular decomposition of the deBruijn graph might be useful for building very big spectrum analyzers and multipliers based on the Schronager-Strassen algorithm [5].

VII. Summary

A novel partition of the deBruijn graph inspired by the problem of building a large constraint length Viterbi decoder has been introduced. The full decoder is built from identical subgraphs called boards, which in turn are built from identical subgraphs called chips. The system is modular and hierarchical, and it implements a large proportion of the required wiring internally within modules. This results in a simpler design, reduced cost, and improved testability and repairability. A constraint length 15 decoder that uses 512 identical VLSI chips and 16 identical printed circuit boards based on this partitioning is a feasible design for decoding at a speed of 1 Mbit/sec.

References

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

[2] R. J. McEliece, *The Theory of Information and Coding*, Massachusetts: Cambridge Press, 1977.

[3] S. W. Golomb, *Shift Register Sequences*, California: Aegean Park Press, 1982.

[4] O. Collins, *Coding Techniques for Low Signal-to-Noise Ratios*, Ph.D. Thesis, California Institute of Technology, in preparation.

[5] A. V. Aho, J. E. Hopcroft, and J. D. Ullman, *The Design and Analysis of Computer Algorithms*, Massachusetts: Addison Wesley, 1974.

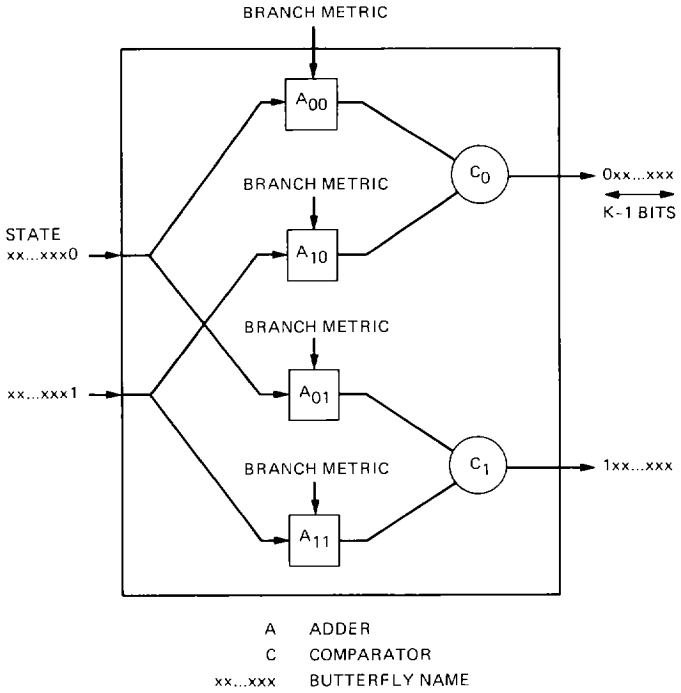


Fig. 1. The innards of a butterfly.

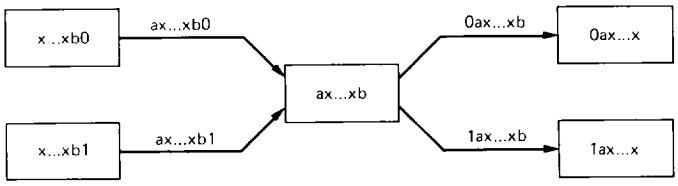


Fig. 2. Butterfly connections and labels.

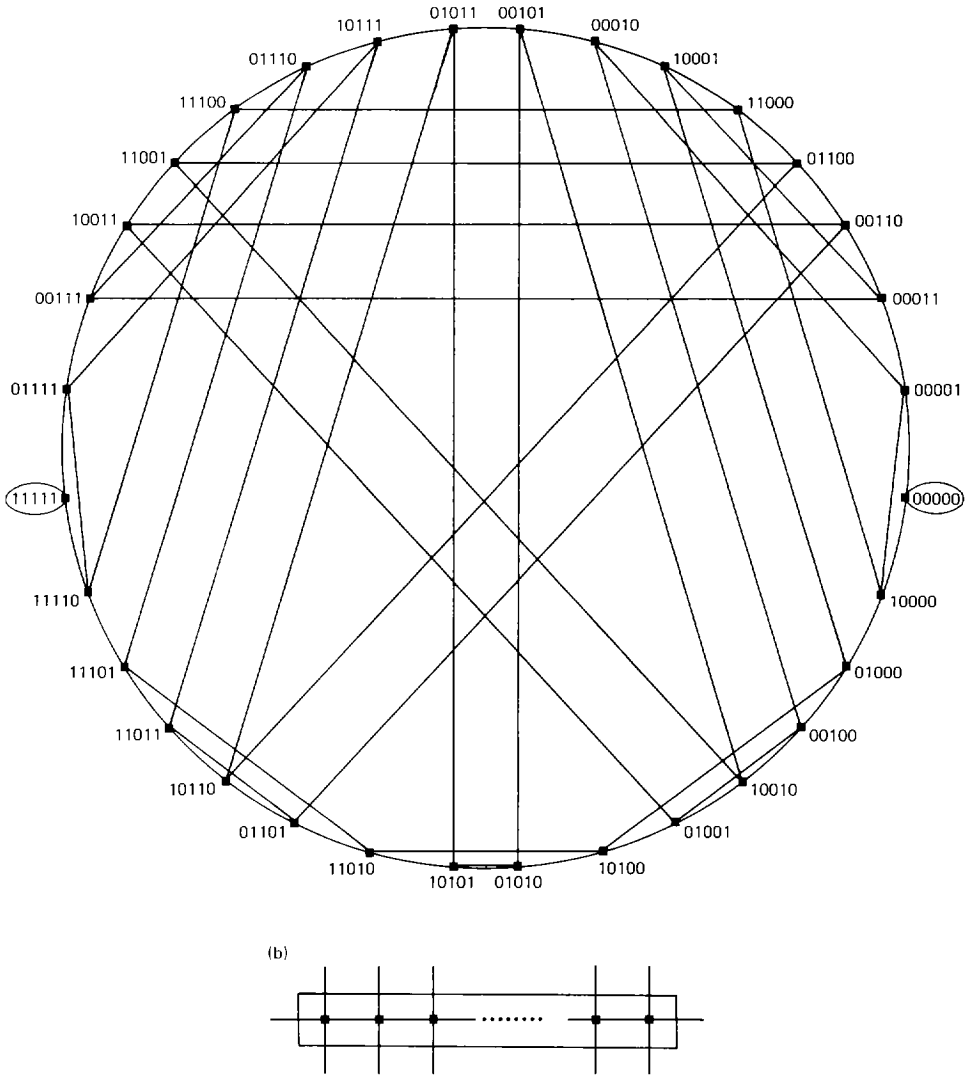


Fig. 3. (a) Butterfly connection topology for a 32-node deBruijn graph Hamiltonian path; (b) a linear module.

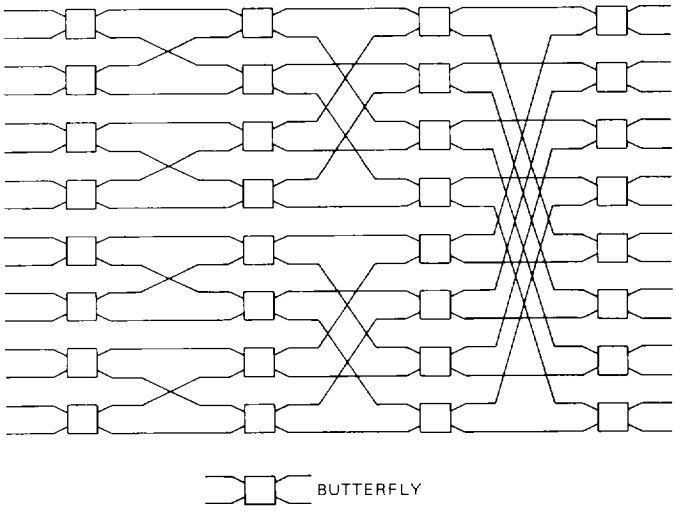


Fig. 4. Connection diagram for an 8 × 4-stage ordinary FFT.

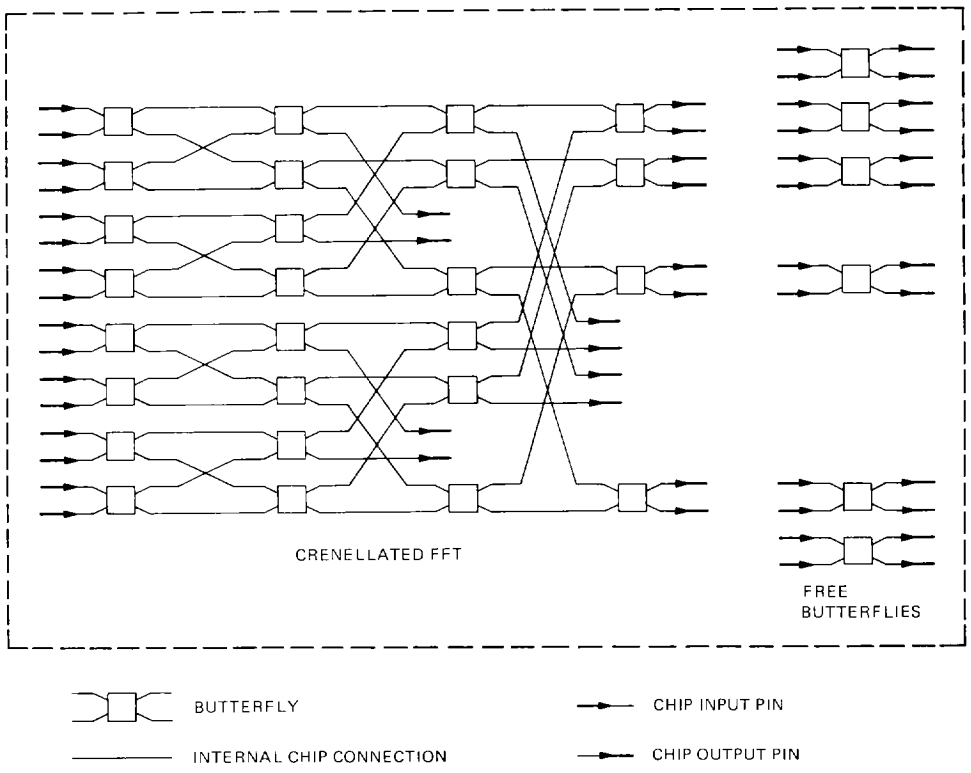


Fig. 5. Connection diagram for a 32-butterfly chip.

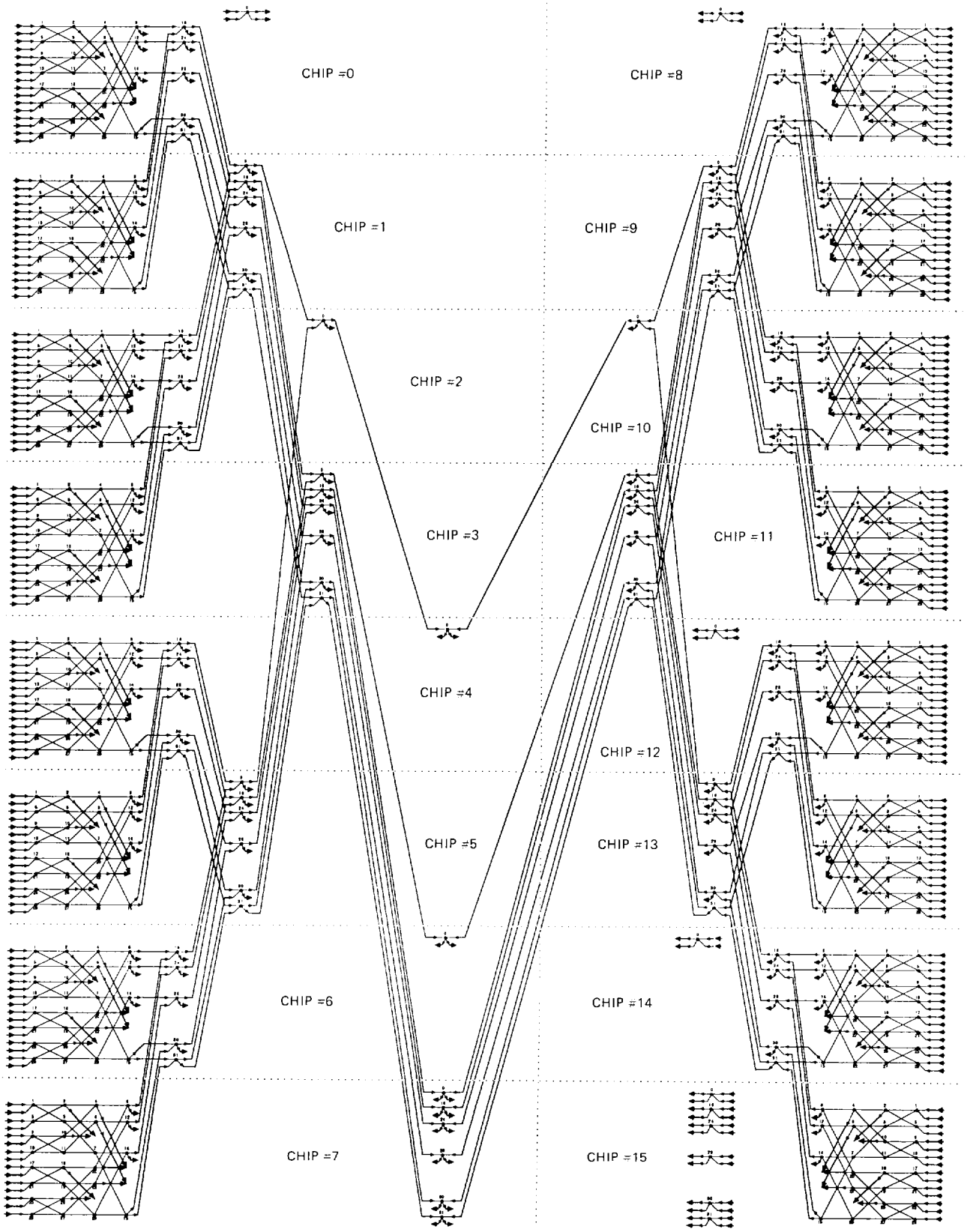
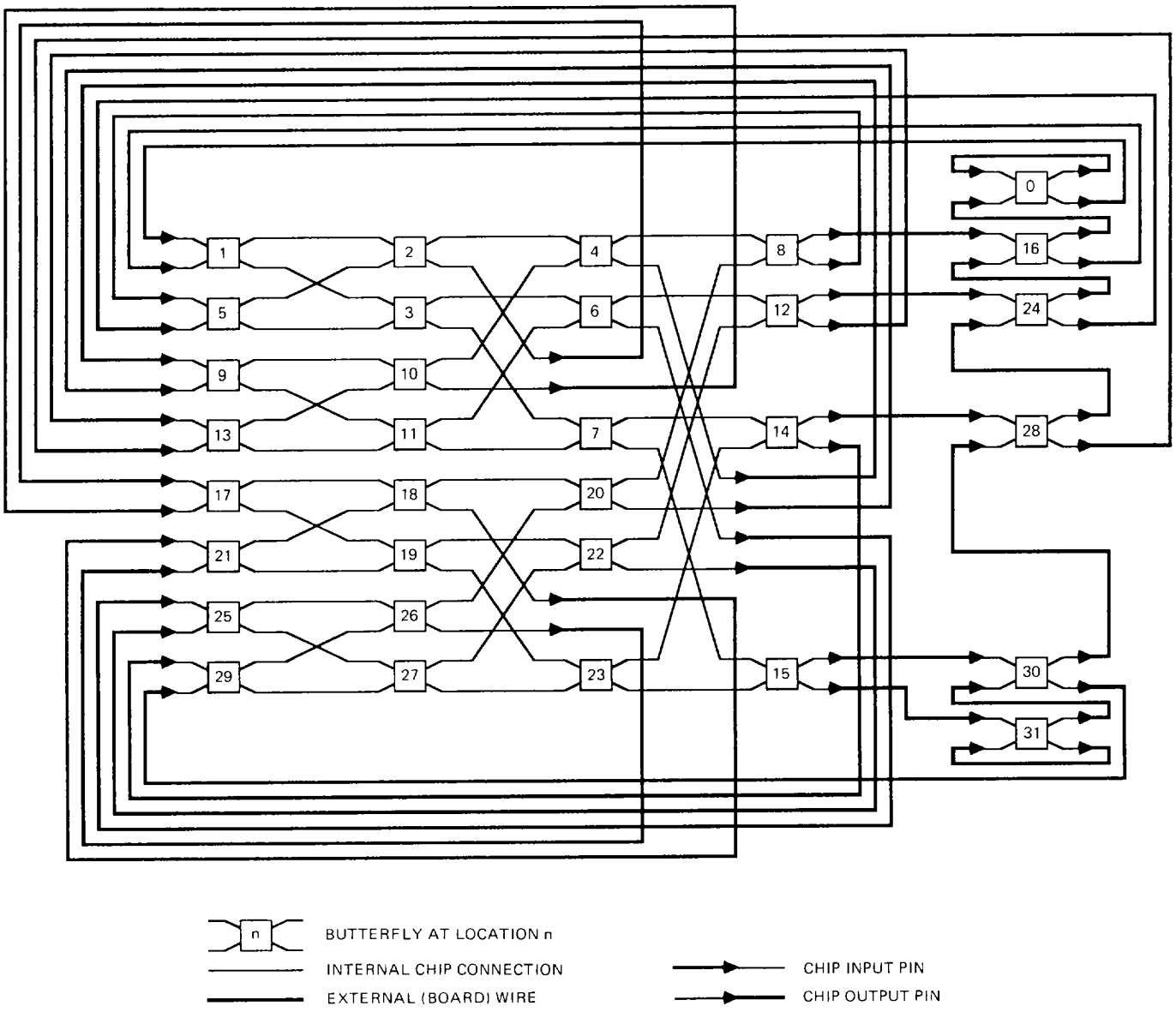
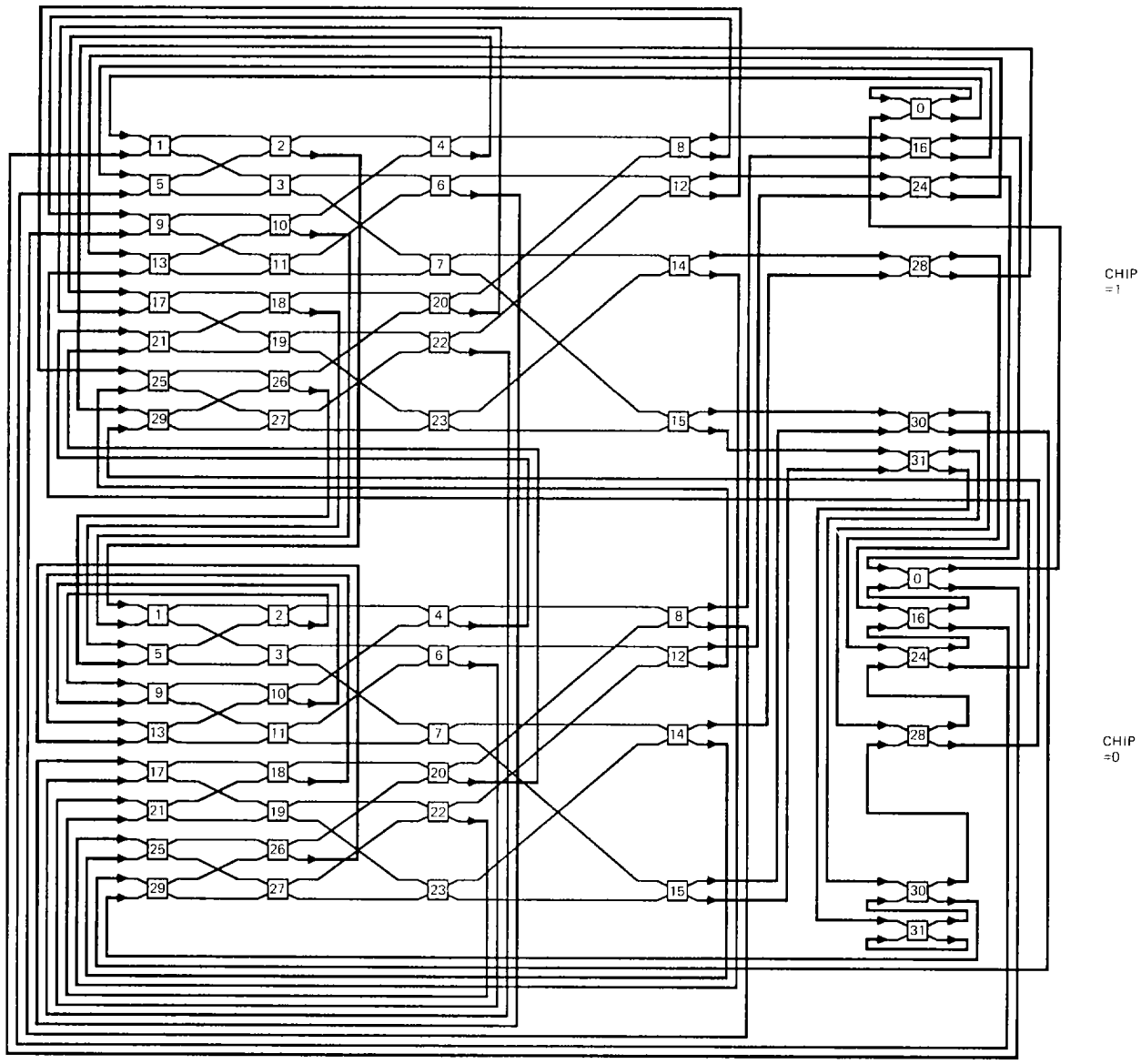


Fig. 6. Connection diagram for a 512-butterfly board.



LOCATION	BUTTERFLY	LOCATION	BUTTERFLY
0	00000	16	00001
1	10000	17	10100
2	01000	18	01010
3	11000	19	11010
4	00100	20	00101
5	10001	21	10101
6	01100	22	01101
7	11100	23	11101
8	00010	24	00011
9	10010	25	10110
10	01001	26	01011
11	11001	27	11011
12	00110	28	00111
13	10011	29	10111
14	01110	30	01111
15	11110	31	11111

Fig. 7. A 32-butterfly chip, wired as a $K = 7$ decoder.



CHIP =0

RELATIVE LOCATION	BUTTERFLY	RELATIVE LOCATION	BUTTERFLY
0	000000	16	000010
1	100000	17	100100
2	010000	18	010010
3	110000	19	110010
4	001000	20	001001
5	100001	21	100101
6	011000	22	011001
7	111000	23	111001
8	000100	24	000110
9	100010	25	100110
10	010001	26	010011
11	110001	27	110011
12	001100	28	001110
13	100011	29	100111
14	011100	30	011110
15	111100	31	111110

CHIP =1

RELATIVE LOCATION	BUTTERLY	RELATIVE LOCATION	BUTTERFLY
0	000001	16	000011
1	101000	17	101100
2	010100	18	010110
3	110100	19	110110
4	001010	20	001011
5	101001	21	101101
6	011010	22	011011
7	111010	23	111011
8	000101	24	000111
9	101010	25	101110
10	010101	26	010111
11	110101	27	110111
12	001101	28	001111
13	101011	29	101111
14	011101	30	011111
15	111101	31	111111

Fig. 8. Two 32-butterfly chips, wired as a $K = 8$ decoder.