A VLSI Implementation of a Multicode Convolutional Encoder

L. J. Deutsch
Communications Systems Research Section

This article describes a VLSI architecture and layout for a convolutional encoder. This architecture allows a single chip implementation of an encoder that is capable of handling many different convolutional codes including all the convolutional codes that are presently used by NASA for deep space missions.

I. Introduction

Most present and all planned deep space missions make use of convolutional coding. One reason for this is that the encoder for these codes is very simple and can be made relatively small and light even with off-the-shelf components. A block diagram of a convolutional encoder is shown in Fig. 1. The convolutional code implemented in the figure is the NASA standard (7, 1/2) code. The input information bits $i_1 i_2 i_3 \cdots$ are shifted into a register. This register is used to generate two parity checks each time a new information bit is entered. A commutator interleaves these parity checks to produce the coded output stream $a_1 b_1 a_2 b_2 a_3 b_3 \cdots$

In a general convolutional decoder, the shift register is K bits long. The number K is called the "constraint length" of the code. In the most general situations, there can be more than one K-bit shift register in the encoder. For the purposes of this article, however, it will be assumed that there is only a single register. If there are P parity checks that are generated by the encoder, then the resulting code rate will be $1/P$. The code would then be referred to as a $(K, 1/P)$ convolutional code. This is not enough to define the code uniquely. The particular code is determined by the parity checks.

Consider a single parity check that is generated in the encoder. Let the positions in the shift register be labeled $1, 2, 3, \cdots$ with the position 1 being the one that the data bits enter first. It follows that this parity check can be described by a K-bit binary number where a one in position i indicates the presence of a parity check tap after state i of the shift register. The parity checks themselves can be ordered so that "parity check #1" is the first to be scanned by the commutator and the "parity check #P" is the last. Using these conventions, any $(K, 1/P)$ convolutional code can be represented by an ordered set of P K-bit binary numbers. For the (7, 1/2) code described above, the numbers are:

1011011

1111001

If one reads the columns of this binary matrix as binary numbers, this can be reduced to simply 3233013. This sequence of integers, together with the more common (7, 1/2) notation, completely describes the convolutional code.

Sometimes one or more of the parity checks are inverted before the multiplexing operation of the commutator. This

is done so that additional channel symbol transitions are introduced in the case that the information bits consist of long strings of either all zeroes or all ones. These transitions are needed in order to synchronize the symbols in the receiver before they can be sent to a decoder. These inversions have no effect on the performance of the code itself but their presence in any usable encoder is essential.

The performance of convolutional codes has been studied in Refs. 1-3. The problem of node synchronization (determining what the order of the parity checks is in the decoder) was treated in Refs. 3-5.

II. A Functional Description of the Chip

The chip, as implemented, is capable of supporting three codes. These are the standard NASA (7, 1/2) code 3233013, the standard NASA (7, 1/3) code 7376147, and a longer constraint length (10, 1/3) code 7461776427. For each of these codes, the capability of inverting any or all of the parity checks has been included in the design.

There are 22 pins that will be connected to the chip. These are:

INPUTS

Vdd	The voltage connection to the chip – nominally -5 Vdc.
Gnd	The ground connection to the chip.
Sub	A connection to allow biassing of the chip substrate.
1/3	Selects the code rate. A "one" signifies that a rate 1/3 code has been selected, while a "zero" means the rate is 1/2.
SEVEN	Selects the code constraint length. A "one" indicates a constraint length seven code while a "zero" indicates a constraint length of ten.
NRM1	This signal determines whether or not the first parity check is inverted. A "one" indicates the noninverted state.
NRM2	The same as NRM1 for the second parity check.
NRM3	The same as NRM1 for the third parity check. This signal has no effect if a rate 1/2 code has been selected.
BITIN	This is the data input.
PHI1IN	The input for the first phase of a two-phase clock.
PHI2IN	The input for the second phase of a two-phase clock.
CLOCKIN	The input for a one-phase clock.
CLKSEL	Selects between a one- and two-phase clock. A "one" indicates that a one-phase external clock has been selected.

OUTPUTS

OUTPUT	The output for the encoded channel symbols.
PHI1OUT	Output of first phase of the internal two-phase clock. If a two-phase external clock has been selected (CLKSEL=0), then this is the same as PHI1IN.
PHI2OUT	Similar to PHI1OUT for the second clock phase.
S1	Indicates when the commutator is scanning the first parity check.
S2	Indicates when the commutator is scanning the second parity check.
S3	Indicates when the third parity check is being scanned. In the case that a rate 1/2 code is selected, S3 is always zero.
P1OUT	The first phase of the shift register advancing clock.
P2OUT	The second phase of the shift register advancing clock.
X#9	The output of the shift register. This is simply BITIN delayed by 10 bit times.

As implemented, there are two ways of getting clocking information into the chip. A two-phase clock can be wired to the device by using PHI1IN and PHI2IN and grounding the input CLKSEL. Also, a one-phase clock can be used on the CLOCKIN input. In the latter case, the input pin CLKSEL must be wired to voltage. This two-choice system is possible because a two-phase clock generator has been included in the chip itself.

A timing diagram that shows the relationship between the time-varying signals is shown in Fig. 2. It is assumed, for the moment, that the selected code rate is 1/3. Notice that the clock frequency must be P times the input data rate (recall that P is the inverse of the code rate). The data that is entered on pin BITIN is clocked through the shift register by the signals P1 and P2. P1 and P2 are simply the logical "and" of S1 and the signals PHI1 and PHI2 respectively. The data in the register will remain stable during the three clock times that it takes to scan the three parity checks. The particular parity checks that are scanned are also functions of the input signal SEVEN. The signals S1, S2, and S3 represent the position of the commutator. It is important that the input bits must be synchronized with the commutator so that a new bit is entered whenever P1 is high. The output symbols on the pin OUTPUT are synchronized with PHI2. Since all the important timing signals are available as pins, it would be an easy task to interface this chip with data sources.

In the case that the selected code rate is 1/2, then the clock frequency is twice that of the input data rate and the signal S3 is always low. This case is indicated in the timing diagram of Fig. 3.

Although the actual clock speeds of the chip cannot be determined until the fabricated device is tested, it is likely that a maximum clock rate of at least 1-2 MHz will be achievable. Such clock rates are nominal for the NMOS technology that is being used for this project.

III. Chip Architecture

The VLSI multicode convolutional encoder logic is divided into two major sections. The first is the timing signal generator and the second is the convolutional encoder array.

The timing signals that were described in Section II are generated in the timing signal generator subblock. A programmable logic array (PLA) (Ref. 6) is used to implement a counter for generating the commutator signals S1, S2, and S3. If the input "1/3" is high, then the counter is set to count modulo three. If "1/3" is slow, then it counts modulo two. P1 and P2 are generated by ANDing S1 with PHI1 and PHI2 respectively. This subblock also contains an output latch that synchronizes the output channel symbols with the signal PHI2 before they are sent to the output pad OUTPUT.

The actual encoding takes place in the convolutional encoder array subblock. It is shown conceptually in Fig. 4. This figure also indicates the geometric layout of this section of the chip. The input bits from BITIN are shifted through the ten-bit parallel output register shown along the bottom of the figure. The output of each register cell is available at the top as well as the right end. Above each cell are six blocks, each of which is either a PRTY (or "parity") cell or a NPRTY (or "no parity") cell. Each row of parity cells corresponds to a particular parity check on the ten outputs of the register. The presence of a PRTY cell in row i and column j means that the i^{th} parity check has a tap at position j in the register. The top three parity checks are for the (10, 1/3) code. The lower three are for the (7, 1/3) code. The (7, 1/2) code uses the first two parity checks from the (7, 1/3) code.

The parity checks are not implemented as exclusive OR operations as is done in a conventional encoder. Instead, there are two signal paths through each row of the parity check matrix. At the extreme left of the array, a "one" is connected to one of these and a "zero" to the other. Each time this pair of signals passes through a PRTY cell, they exchange places if the output of the corresponding register cell is a "one." A NPRTY cell has no effect on these signals (circuit diagrams for these two cells appear in Fig. 5). In this way, the pair of signals has gone through a number of path exchanges equal to the number of ones in the shift register at the taps that correspond to that particular parity check. If the parity is even, then the signals are in the same place as when they started. If the parity is odd, they have come out reversed. Also, since a complementary pair of signals is used, the complement of the parity check, as well as the check itself, is available at this point.

The signals NRM1, NRM2, and NRM3 are used to select either the parity information or its complement at the end of each row of the parity array. The signal SEVEN is then used to select either the top three or bottom three parity checks. The commutator is implemented as a multiplexer in which the signals S1, S2, and S3 are used to select the appropriate parity check.

This architecture is a good example of how an algorithm that works well in a SSI system is not nearly optimal in VLSI. In the conventional encoder, it would be impractical to provide all the interconnections that are necessary for implementing the above algorithm. This VLSI architecture is compact, fast, and easily extendable to longer constraint lengths and more parity checks. In fact, it would be a relatively simple matter to write a silicon compiler (Ref. 7) that would automatically lay out masks for similar encoders for arbitrary sets of convolutional codes.

It should be mentioned that the overall size of this chip was determined by the number of pads rather than the size of the logic. This means that there is some extra space available. Some of this space is filled with a long array of "butting contacts." A butting contact is a structure for connecting the polysilicon and diffusion layers in NMOS. There is some doubt as to whether this structure will work reliably in the 2.0-

micron NMOS technology. By including this test array it will be possible to gather some additional data on this subject. The two ends of the array of contacts are connected to the pins BUTT-TEST1 and BUTT-TEST2. It should be noted that the butting contact was not used in the remainder of the chip.

IV. The Design Process

The multicode convolutional encoder chip was designed according to the NMOS design rules that appear in Ref. 6. The design philosophy was influenced by the JPL VLSI design course as taught by Dr. G. Lewicki and Dr. R. Nixon.

The actual chip layout was performed using the interactive graphics editor "CAESAR" that was developed at the University of California at Berkeley by Dr. J. Ousterhout (Ref. 8). This software design system allows the user to develop a chip in a hierarchical fashion with full editing capabilities. There are also a number of programs including a PLA generator and optimizer that can be used in conjunction with CAESAR. A picture of the finished layout as produced with CAESAR appears in Fig. 6. The UC Berkeley software also includes a circuit extractor called MEXTRA that works in conjunction with CAESAR to create a file that can be used with a number of simulation packages. One very nice feature of the extractor is that it keeps track of labels that the user creates with CAESAR so they can be referred to during simulation. A complete description of the Berkeley software can be found in Ref. 9. The chip was completely simulated on a logical level using the program ESIM that was developed at the Massachusetts Institute of Technology. Although it was not used for this project, the circuit level simulator SPICE (Ref. 10) is also integrated with CAESAR and MEXTRA.

V. Conclusions

It is evident, even in a simple project such as this one, that VLSI technologies allow the implementation of much more efficient algorithms than conventional circuit design. This is because all intermediate signals that are generated in a circuit can be made available to other portions of the circuit without a proliferation of pins and drivers that can greatly reduce the computational speed of the system.

The chip described here was sent out over the ARPANET for fabrication on December 2, 1982. When the completed chips return they will be evaluated and tested.

These chips will probably be used as test function generators in research on convolutional decoding. They will not be radiation hardened and so they cannot be flight qualified. However, the architecture described in this article could be easily used to produce flight qualified chips if the need ever arises. Also, the experience gained on this project will be invaluable in the design of the complex decoding systems that are planned for future projects.

References

1. Layland, J. W., "Information Systems: Performance of Short Constraint Length Convolutional Codes and a Heuristic Code-Construction Algorithm," *Space Program Summary 37-64, Vol. II*, Jet Propulsion Laboratory, Pasadena, Calif., Aug. 1970.

2. Miller, R. L., Deutsch, L. J., and Butman, S. A., *On the Error Statistics of Viterbi Decoding and the Performance of Concatenated Codes*, Publication 81-9, Jet Propulsion Laboratory, Pasadena, Calif., Sept. 1, 1981.

3. Liu, K. Y., and Lee, J. J., "An Experimental Study of the Concatenated Reed-Solomon/Viterbi Channel Coding System and Its Impact on Space Communications," Publication 81-58, Jet Propulsion Laboratory, Pasadena, Calif., Aug. 15, 1981.

4. Deutsch, L. J., and Miller, R. L., "The Effects of Viterbi Decoder Synchronization Losses on the Telemetry Receiving System," *TDA Progress Report 42-68*, Jet Propulsion Laboratory, Pasadena, Calif., Apr. 15, 1982.

5. Deutsch, L. J., and Miller, R. L., "Viterbi Decoder Node Synchronization Losses in the Reed-Solomon/Viterbi Concatenated Channel," *TDA Progress Report 42-71*, Jet Propulsion Laboratory, Pasadena, Calif., Oct. 15, 1982.

6. Mead, C., and Conway, L., *Introduction to VLSI Systems*, Addison-Wesley Publishing Company, Reading, Mass., 1980.

7. McNair, R., and Miller, M., "Bristle Blocks – Scrutinized and Analyzed," Computer Science Department Display File, California Institute of Technology, 1982.

8. Ousterhout, J. K., "Caesar: An Interactive Editor for VLSI Layouts," *VLSI Design*, Volume II, No. 4, Fourth Quarter 1981.

9. Fitzpatrick, D. T., et al., "A RISCy Approach to VLSI," *VLSI Design*, Volume II, No. 4, Fourth Quarter 1981.

10. Negal, L. W., and Pederson, D. O., "SPICE – Simulation Program with Integrated Circuit Emphasis," Memorandum No. ERL-M382, Electronics Research Laboratory, University of California, Berkeley, April 12, 1973.

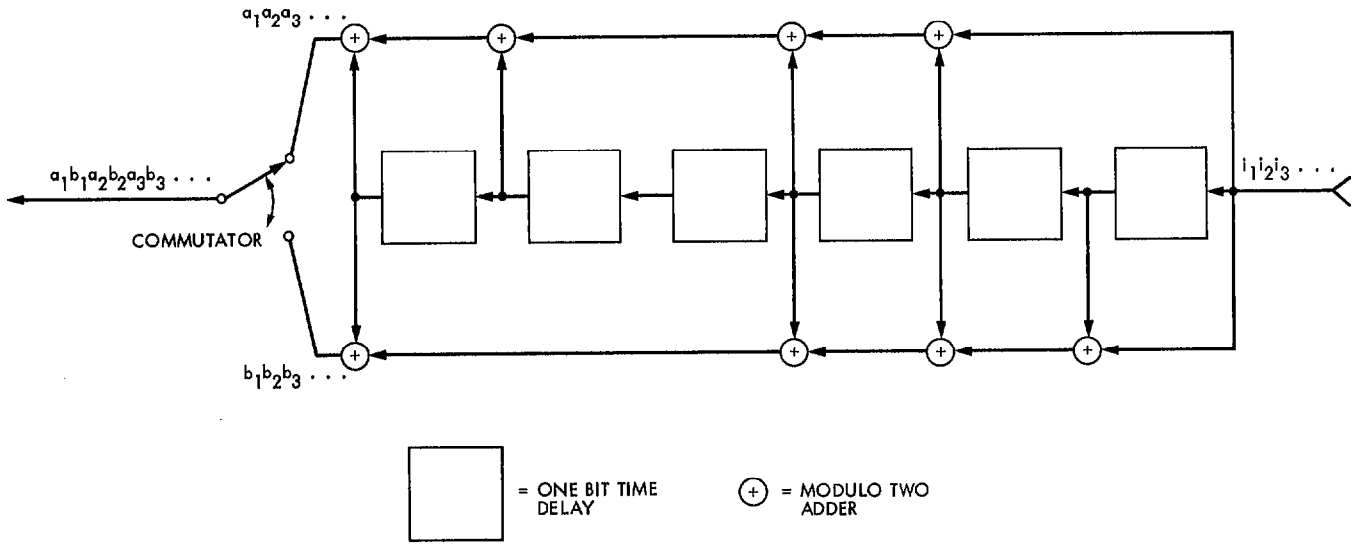

Fig. 1. Conceptual diagram of a (7, 1/2) convolutional encoder

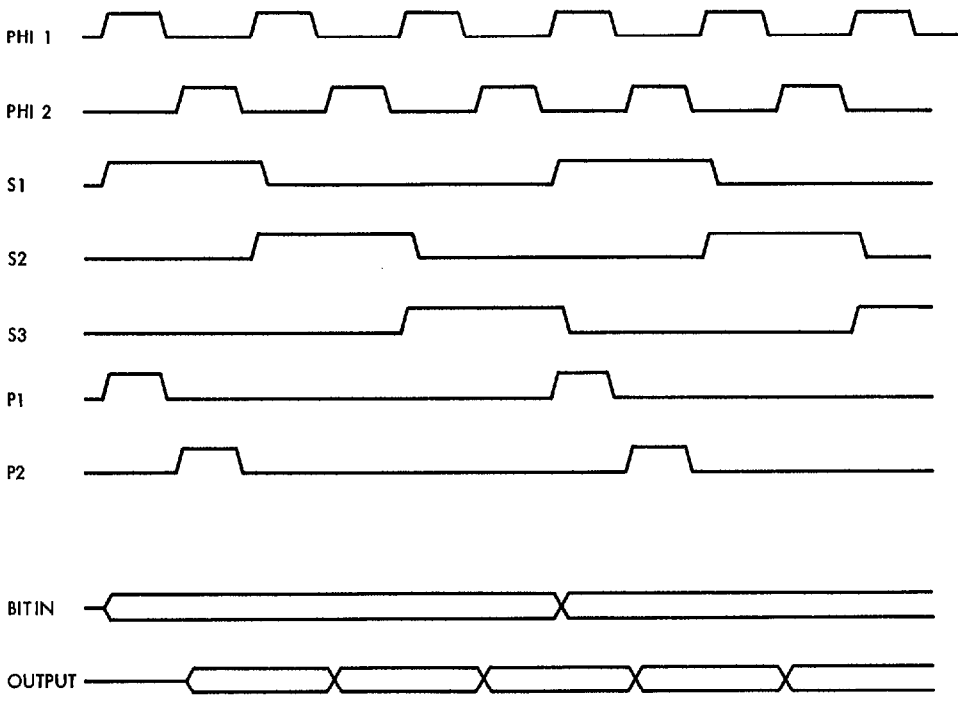

Fig. 2. Timing signals for the convolutional encoder, rate 1/3 codes

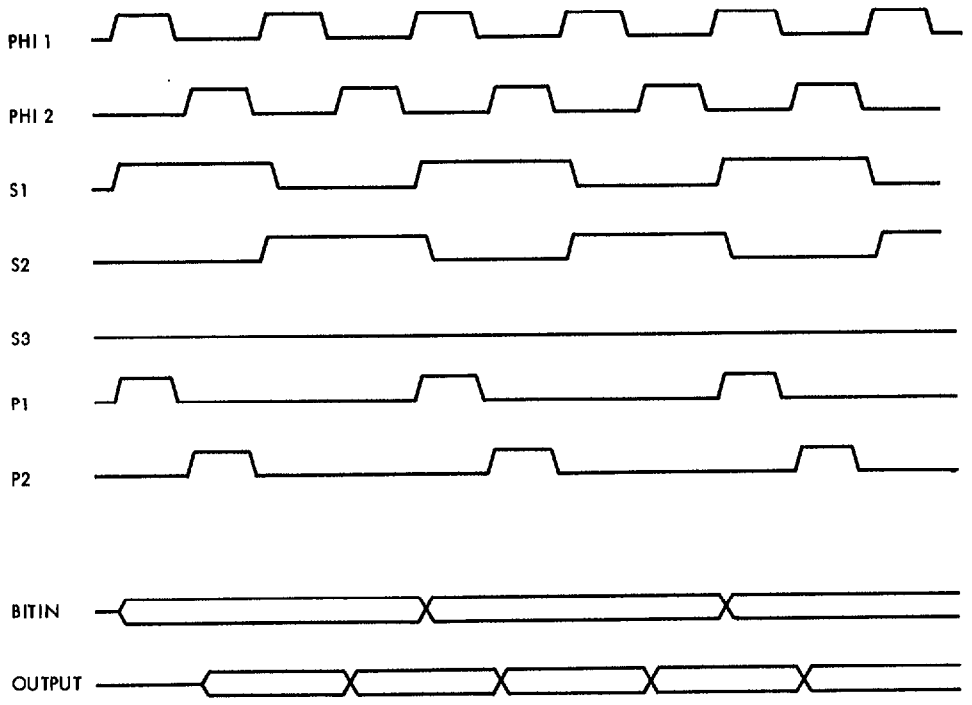

Fig. 3. Timing signals for the convolutional encoder, rate 1/2 codes

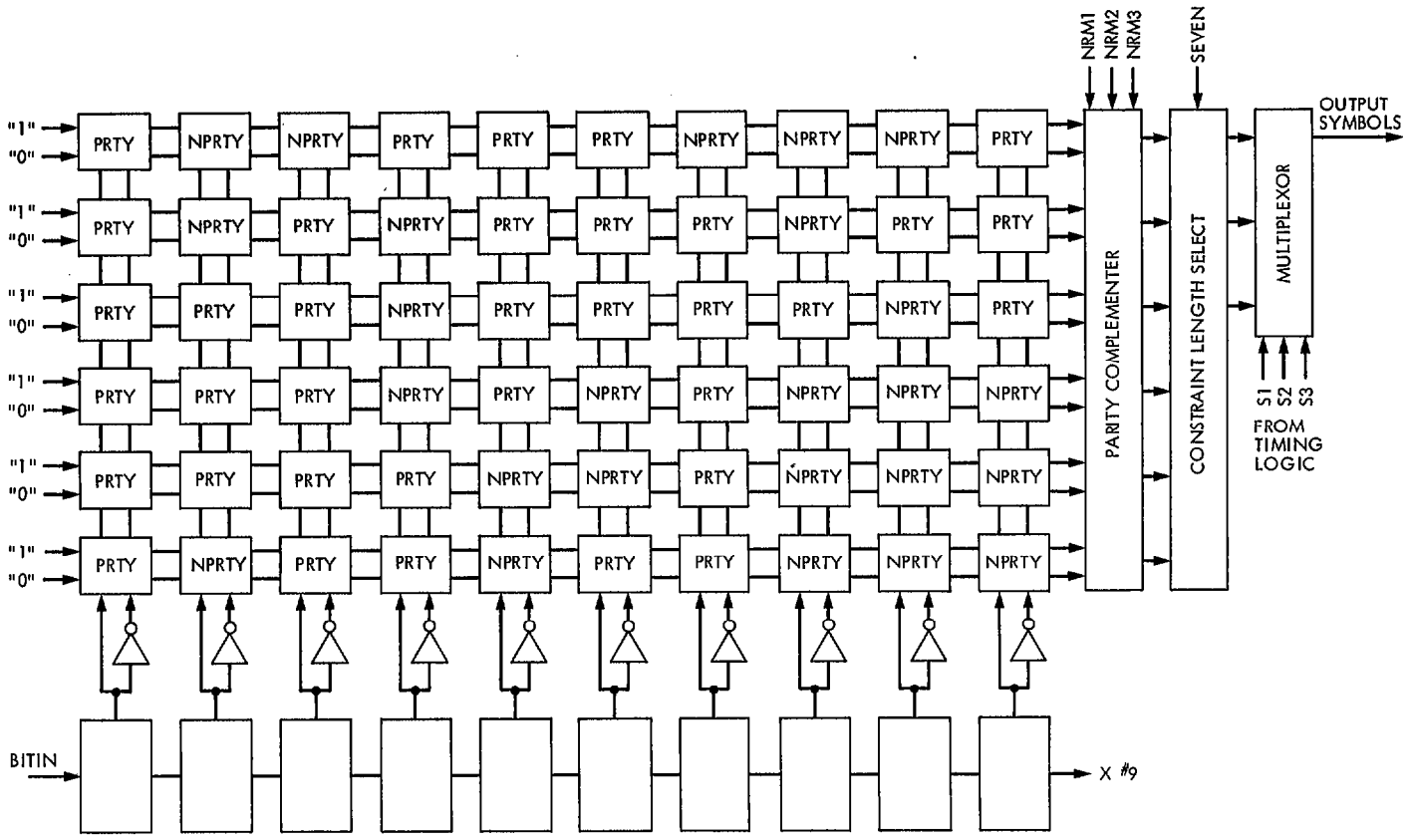

Fig. 4. Conceptual diagram of the encoder logic array

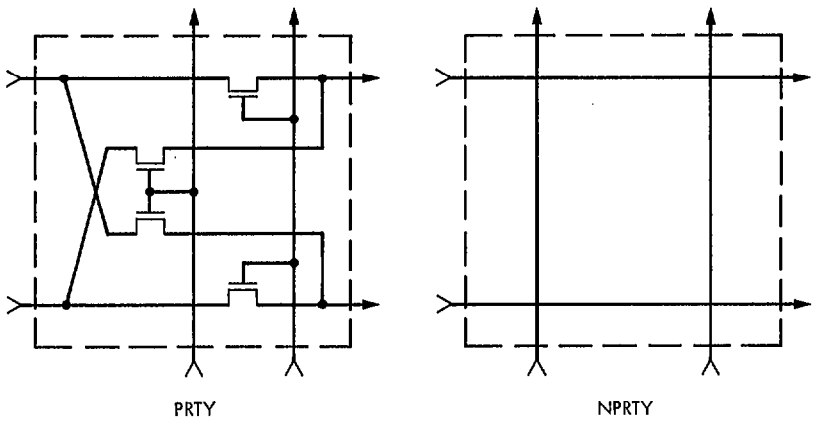

Fig. 5. The PRTY and NPRTY cells

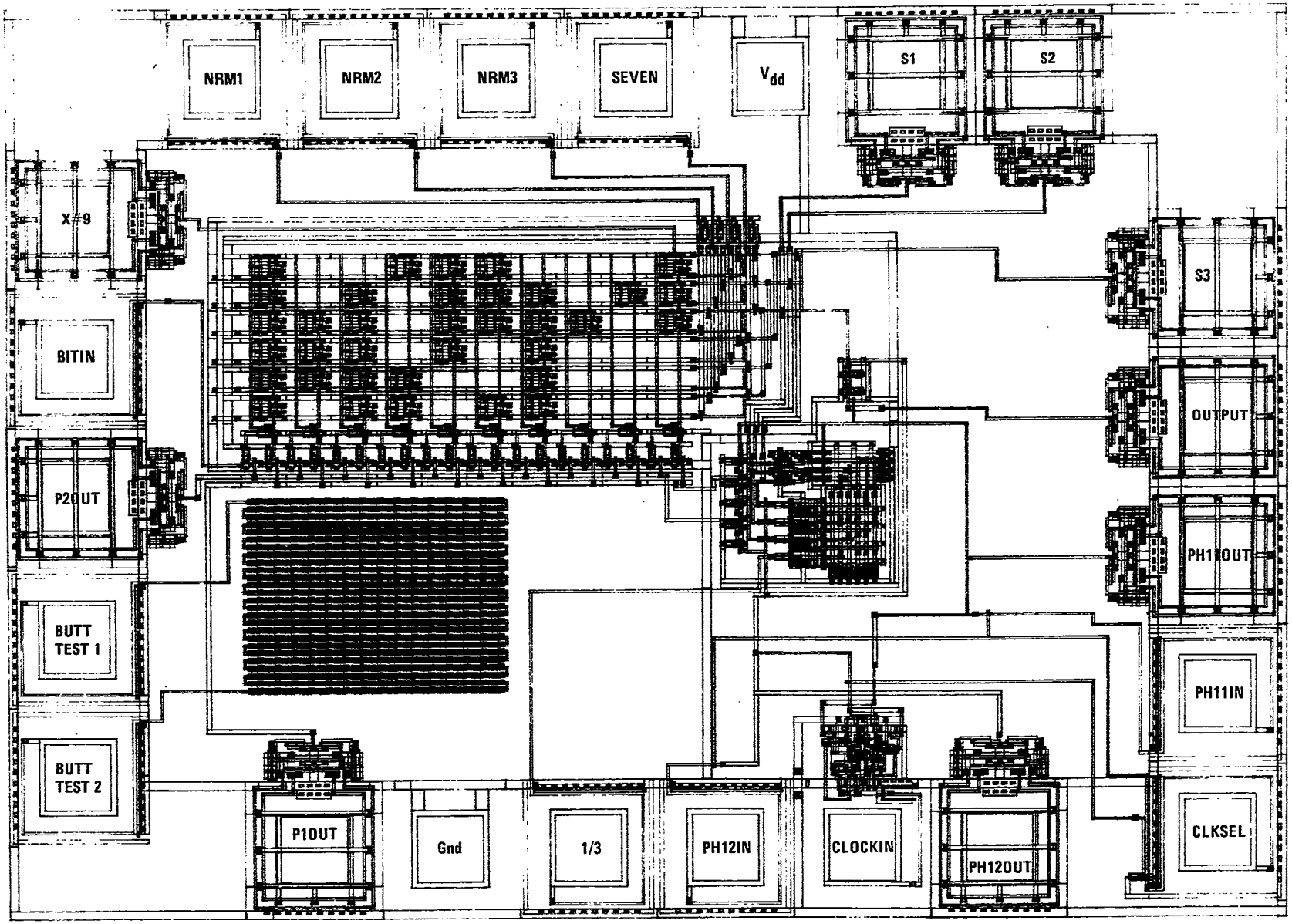

Fig. 6. Layout of the multicode convolutional encoder chip