VLSI IMPLEMENTATION OF A SYSTOLIC ARRAY VITERBI DECODER

by

NORLAILI BINTI MOHD. NOH

Thesis submitted in fulfilment of the
requirements for the degree of Master of Science

March 1995

ACKNOWLEDGEMENT

I would like to express my gratitude to my supervisor Dr. Rezaul Hasan for his assistance and guidance throughout the project. He has provided me with helpful information, constructive suggestions, useful material and has been a good supervisor.

I am deeply indebted to many people, in particular Khow Teong Chin and K. Jayabalan. Khow introduced me to Xilinx and Jaya's help in VLSI has benefited me significantly.

I would like to thank those making helpful comments on how to improve the performance of the decoder circuit especially to Chan Hong Heng, Seelan Kumarasamy, and Azmi Said.

In addition, I would like to give special thanks to Abdul Rahman Saad for his many suggestions, assistance, and time. I benefited greatly from the discussions that we always had.

I am grateful to Azlina, Sahlawati, Abdul Latip, Roslan and Saparudin for their technical assistance and for creating a research environment. I am also grateful to my departmental colleagues, Dr. Ali Yeon in particular for their help and tolerance.

I must thank my two good friends Umi Kalthum Ngah and Chean Kooi Chyi for encouraging and supporting me throughout the project. Special thanks to Zaida Mohamath who played an essential role in typing this thesis.

Finally, I acknowledge with gratitude the patience, support and understanding of my family.

Thank you one and all.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

PERLAKSANAAN VLSI BAGI SATU PENYAHKOD VITERBI BERTATASUSUN SISTOLIK

ABSTRAK

Penyahkodan Viterbi adalah kaedah bertenaga bagi pembetulan ralat ke depan dalam pengekodan pelingkaran. Terdapat permintaan bagi perlaksanaan penyahkod Viterbi dalam VLSI bagi penggunaan-penggunaan seperti HDTV (High Definition Television) digit menerusi satelit dalam julat jalur lebar, bagi pengekodan sistem pada saluran-saluran satelit dan angkasa lepas, bagi penghantaran digit halaju tinggi melalui saluran telefon, dalam pengekodan pelingkaran bagi BPSK, dalam pengekodan isyarat PCM dupleks penuh dan perhubungan jarak jauh serta perhubungan angkasa lepas yang lain.

Projek ini adalah berkisar tentang perkembangan satu penyahkod Viterbi yang menggunakan kaedah kesan semula dan strukturnya adalah dalam bentuk tatasusun sistolik. Adalah dipercayai yang rekabentuk ini boleh mengurangkan saiz penyahkod tersebut kerana ia meminimakan sambungan-sambungan di antara modul-modul komponen dan memerlukan ruang storan yang lebih kecil. Kaedah kesan balik boleh mengurangkan bilangan perkakasan yang merupakan masalah bagi penyahkod pertukaran daftar. Ia adalah juga bersesuaian bagi mendapatkan operasi pada kelajuan yang lebih tinggi kerana pengesanan, pembaharuan dan penstoran jujukan maklumat boleh dibuat serentak melalui satu kitaran jam.

Perlaksanaan VLSI penyahkod Viterbi bertatasusun sistolik ini telah dibahagi kepada dua bahagian; bahagian pertama ialah perlaksanaan perkakasan VLSI menggunakan peranti-peranti FPGA VLSI dan bahagian kedua, perlaksanaan VLSI cip tunggal.

Bagi bahagian pertama, cip-cip FPGA VLSI XC3042 adalah digunakan. Xilinx Automatic CAE Tools (XACT) Development System digunakan dalam merekabentuk. OrCAD/SDT digunakan sebagai masukan skematik. Cip-cip FPGA digunakan sebagai satu alat demontrasi. Bagi menguji litar penyahkod, satu isyarat jam dengan frekuensi 20 MHz dikenakan kepada litar jam yang menghasilkan dua isyarat jam, dipanggil "clock 1" dan "clock 2" dengan frekuensi 5 MHz setiap satu. Kedua-dua isyarat ini kemudiannya dikenakan kepada litar penyahkod. Walau bagaimanapun, adalah dipercayai bahawa perkakasan penyahkod tersebut dapat juga beroperasi dengan baik pada frekuensi yang lebih tinggi daripada itu. Perkakasan penyahkod ini menghasilkan keluaran yang mempunyai glic, dan dengan ini penimbal-penimbal telah dimasukkan untuk memberikan perlakuan yang lebih baik. Perlaksanaan perkakasan penyahkod tersebut adalah terdiri daripada hanya IC Xilinx 3042 dan EPROM SG-Thompson 2764. Hasilnya, satu perkakasan yang kemas dan padat dapat dibina menutupi keadaan sebenar litarnya yang agak kompleks. Gambar perkakasan ini ada diberikan dalam tesis ini. Pengujian perkakasan tersebut akan dapat memastikan yang litar tersebut berfungsi dengan sebenarnya, bagi perlaksanaan VLSI yang akan datang sebagai satu langkah dalam menghasilkan cip penyahkod Viterbi.

Bagi bahagian kedua projek ini, Electric VLSI Design System digunakan untuk merekabentuk litar VLSI. Ini adalah merupakan rekabentuk "full custom". Rekabentuk ini mengandungi lebih kurang 20,000 transistor. Walau bagaimanapun, kawasan yang digunakannya adalah agak besar disebabkan oleh rekabentuk secara nod selari/bit-selari.

Keseluruhan bentangan cip penyahkod mempunyai 10 pin. Gambar-gambar bentangan bagi keseluruhan litar penyahkod dan beberapa litar-litar utama adalah ditunjukkan dalam tesis ini.

Bagi penyelakuan beberapa daripada litar VLSI ini, PSpice Circuit Analysis Version 4.04 digunakan. Penyelakuan telah dilakukan ke atas litar-litar utama dengan menggunakan masukan-masukan bit 10 MHz. Program-program penyelakuan dan hasilnya, bagi beberapa daripada litar utama penyahkod tersebut, adalah diberikan dalam tesis ini. Keputusan daripada penyelakuan tersebut telah mendapati bahawa tempoh lengah kritik bagi litar penyahkod tersebut adalah 6 nS.

Daripada kedua-dua jenis perlaksanaan VLSI, didapati bahawa perlaksanaan cip tunggal akan menjimatkan kawasan berbanding dengan perlaksanaan berbilang cip VLSI, serta ia juga dijangka beroperasi pada kelajuan yang lebih tinggi.

ABSTRACT

Viterbi decoding is a powerful method for forward error correction in convolutional coding. There is a growing need for implementing Viterbi decoder in VLSI for applications such as in Digital HDTV (High Definition Television) via satellite in the wide band range, for coding systems on deep space and satellite channels, for high speed digital transmission over telephone channels, in convolutional coding for BPSK, in full duplex PCM signal decoding and other deep space and long distance communication.

This project is on developing a Viterbi decoder which uses the trace-back method structured in a systolic array fashion. It is believed that this architecture can reduce the size of the decoder as it minimizes the connections between component modules and requires a smaller storage space. The trace-back method can reduce the amount of hardware which is normally a problem with register exchange decoder. It is also suitable for achieving a higher speed of operation as tracking, updating and storage of the information sequence can be accomplished simultaneously during a single clock cycle.

VLSI implementation of the systolic array Viterbi decoder is divided into two parts; part one is the hardware implementation using VLSI FPGA devices and part two, single chip VLSI implementation.

In the first part, VLSI XC3042 FPGA chips are used. Xilinx Automatic CAE Tools (XACT) Development system is used in designing. ORCAD/SDT is used for schematic

entry. The FPGA chips are used as a demonstration vehicle. For testing the decoder circuit, a clock signal with frequency 20 MHz is fed into the clock circuit which produces two clock signals, termed clock 1 and clock 2 with frequency 5 MHz each. These two signals are then fed into the decoder circuit. It is believed, however, that the decoder hardware can perform as well at a higher frequency rate. This decoder hardware produces glitches at its output but buffers have been added to give better performance. The hardware decoder implementation consist of only Xilinx 3042 IC's and SG-Thompson 2764 EPROM's. As a result, a very neat and compact hardware is developed masking its rather complex circuitry. A photograph of the circuit hardware is shown in the thesis. Testing the hardware will ensure that the circuit is working, for future VLSI implementation as a step in producing a systolic array Viterbi decoder chip.

In the second part, Electric VLSI Design system is used for the VLSI circuit design. This is a full custom design. The design contains approximately 20,000 transistors. However, the area that it consumed is quite large due to the disadvantage of node parallel/bit-parallel architecture. The total decoder chip layout has a total of ten pins. Photographs of the total decoder circuit layout and some of the layouts of the main circuits in the decoder are shown in the thesis.

For simulation of these VLSI circuits, PSpice Circuit Analysis Version 4.04 is used. The main circuits were simulated using 10 MHz bit inputs. Simulation programs and results of some of the main circuits in the decoder are shown in the thesis. From the simulation results, it is found that the critical delay period for the decoder circuit is 6 nS.

From the two types of VLSI implementations, it is found that the single chip implementation will save space compared to the multiple VLSI chips implementation and also expected to work at a higher speed.

CHAPTER 1

INTRODUCTION

1.1 VITERBI ALGORITHM

The Viterbi algorithm was first developed for convolutional codes, and naturally it has its greatest impact here. The algorithm is used in decoding convolutional codes in the presence of intersymbol interference. Convolutional codes can be viewed as a special class of linear block codes. However, we shall find that the additional convolutional structure endows a linear code with superior properties which both facilitate decoding and improve performance.The term "convolutional" applies to this class of codes because the output symbol sequence can be expressed as the convolution of input (bit) sequence with the generator sequences. The nature of the Viterbi decoding algorithm is demonstrated by the use of a $\left(2, \frac{1}{2}\right)$ convolutional code. Here 2 and $\frac{1}{2}$ represent the constraint length, k and code rate $\frac{b}{n}$ respectively.

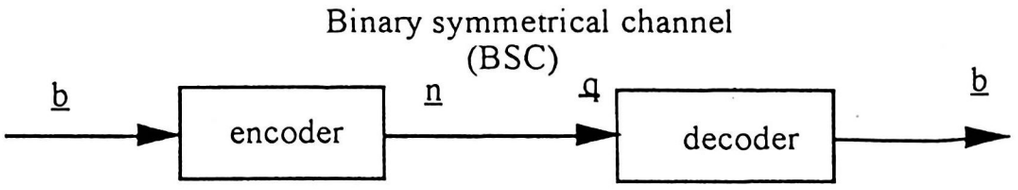

Figure 1.1 - Transmission of information bit

An example of a fixed convolutional code with k = 2 and $\frac{b}{n} = \frac{1}{2}$ is shown as Figure 1.2.

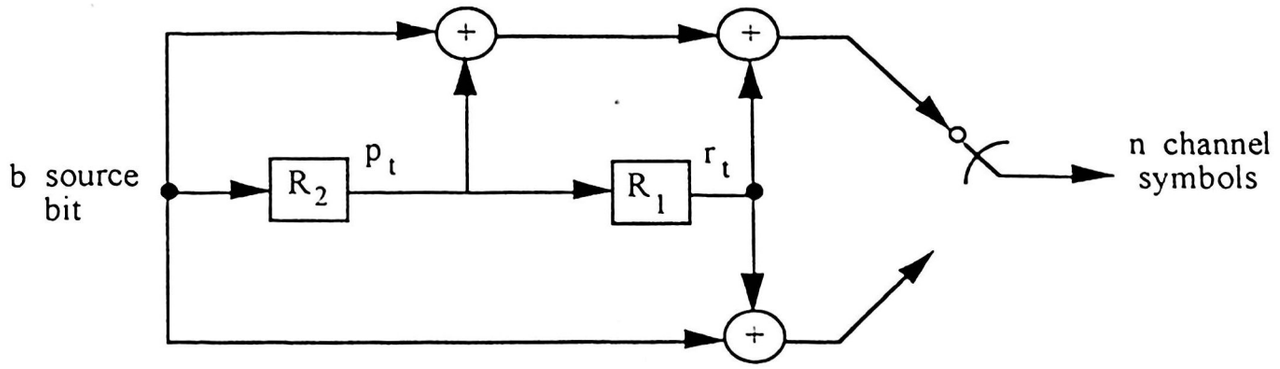

Figure 1.2 - A $\left(2, \frac{1}{2}\right)$ convolutional code encoder

The encoder is composed of a 2 stage shift register with three modulo 2 adders and a multiplexer for converting a parallel output to a serial output. During each frame time, a new information bit stored in R1 is shifted out of the shift register. At the end of each frame, the encoder stores the next state in the shift register and output the next piece n of the codeword.

A fixed convolution coder may be regarded as a linear time invariant finite-state machine whose structure can be exhibited with the aid of any one of several diagrams. One of these is the trellis diagram.

The trellis diagram of the above convolutional code is shown in Figure 1.3.

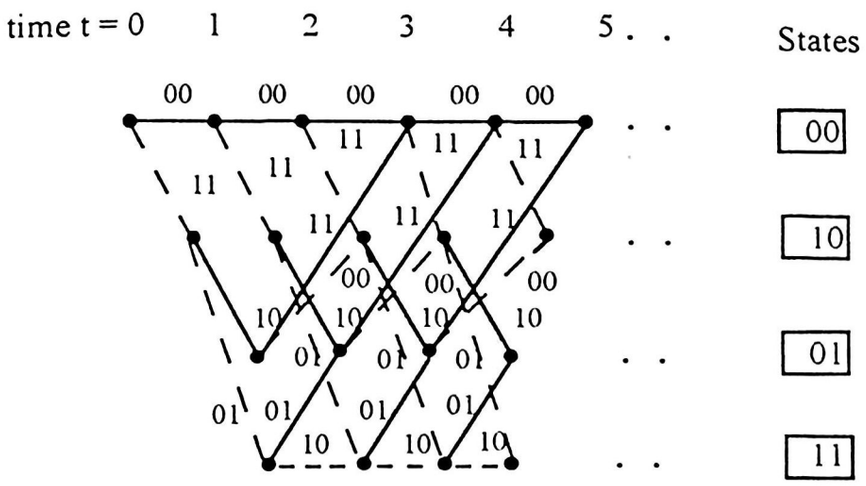

Figure 1.3 - Trellis diagram for a (2, 1/2) convolutional code

The code branches produced by a '0' input bit are shown as solid lines and code branches produced by a '1' input bit are shown dashed. Outputs are indicated by symbols along the branches.

The completely repetitive structure of the trellis diagram suggests a further reduction of the code to the state diagram of Figure 1.4. The states of the state diagram are labeled according to the nodes of the trellis diagram.

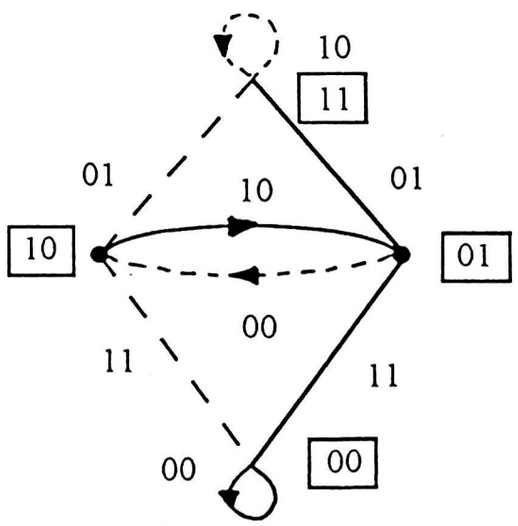

Figure 1.4 - State diagram for encoder of Figure 1.2

A good explanation on convolutional code, trellis diagram and state diagram can be found in [1], [5], [6], [12], [13], and [14].

To generalize a rate $\frac{b}{n}$ convolutional code, we note simply that a trellis of 2^{bk} states will occur with all branchings and mergings occuring in groups of 2^b branches. Consequently, the state diagram will also have 2^{bk} states, with each state having 2^b output branches emanating from it and 2^b input branches arriving into it. Therefore for a $\left(2, \frac{1}{2}\right)$ convolutional code, there are four states of shift register with 2 input branches and 2 output branches entering and emanating from each state.

Assume that state vector at time t is

$$m_t = (p_t, r_t) \qquad 1(a)$$

$$= (b_{t-1}, p_{t-1}) \qquad 1(b)$$

Note that r_{t-1} of the state vector m_{t-1} at time unit t-1 is shifted out of the encoder register when the state vector changes to m_t.

If codeword $n = (n_0, n_1, n_2, ...)$ is transmitted over a binary symmetrical channel (BSC) and the received sequence is $q = (q_0, q_1, q_2, ...)$(refer to Figure 1.1), then the branch metric from state $S_{m_{t-1}}$ at time unit t-1 to S_{m_t} at time unit t is defined by

$$d_{t-1,t}(m_{t-1}, m_t) = \| q_{t-1} - n_{t-1} \| \qquad (2)$$

where $d_{t-1, t}(m_{t-1}, m_t)$ denotes Hamming path distance from state $S_{m_{t-1}}$ at time unit t-1 to S_{m_t} at time unit t. $\| q_{t-1} - n_{t-1} \|$ is the Hamming weight of the difference of the two binary vectors q_{t-1} and n_{t-1}.

A BSC is the most common type of discrete memory less channel with $\mathcal{X} = \mathcal{Y} = \{0, 1\}$ and conditional probabilities of the form

$$p(1/0) = p(0/1) = p$$

$$p(0/0) = p(1/1) = 1\text{-}p$$

These are the probabilities for outputs given each of the input. $\mathcal{X}$ and $\mathcal{Y}$ are the discrete input alphabet and discrete output alphabet respectively. A BSC is represented by the diagram of Figure 1.5.

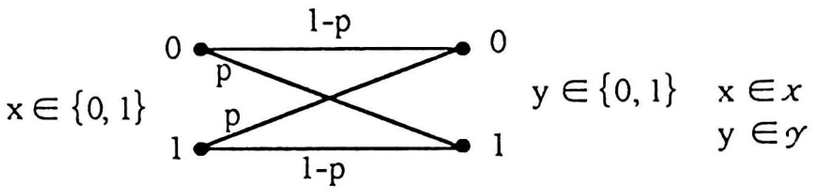

Figure 1.5 - Binary symmetrical channel

The partial path metric of a state S_{m_t} up until the end of the first t branches of a path is defined by

$$M_t(m_t) = \sum_{j=1}^{t} d_{j-1,j}(m_{j-1}, m_j) \qquad (3)$$

where m_{j-1} and m_j are the state numbers on the path, m_{j-1} is the state number at time unit j-1 and m_j is the state number at time unit j for j = 1, 2, ... t. The partial path metric can also be expressed by the difference equation,

$$M_t(m_t) = M_{t-1}(m_{t-1}) + d_{t-1,t}(m_{t-1}, m_t) \qquad (4)$$

Let $Y_t(m_t)$ to be the set of all paths that start S_0 at time unit 0 and end at state S_{m_t} at time unit t. The smallest metric for all paths terminating at state S_{m_t} at time t is given by

$$P_t(m_t) = \min_{Y_t(m_t)} M_t(m_t) \qquad (5)$$

$P_t(m_t)$ is the weight (Hamming weight) of what is called the survivor path to state S_{m_t} at time unit t. The viterbi decoder for a convolutional code finds the survivor path, which is the path of minimum metric, which reaches a given state at time unit t.

The Viterbi algorithm implemented in a convolutional code encoder is summarised as follows:

(1) Begin at t = 1

Compute Mj(mj)

Store survivor and its metric for each state.

(2) Increase t by 1. Compute Mj(mj) for all the paths entering a state by adding the branch metric entering that state to the metric of the connecting survivor at the preceding time unit. For each state store the survivor together with its metric, and eliminate all other paths.

(3) If j < length of codeword, repeat step (2). Otherwise, stop. The final survivor with the smallest metric can be used to decode the information bit along this path.

(4) Define data window. The decoding window length, L, should be in the order of several times k. At time unit L, the decoder chooses the surviving path which reaches the state with the smallest metric among the 4 nodes within the decoding window. The first branch in this window can be

decoded as the first information bit. This decoding window shifts out the decoded data of the first branch and accepts data of a new frame at time L + 1 of the received codeword for the next iteration. In a similar manner the decoding window W is used then to decode the second and succeeding decoded information bits. The process continues in this fashion indefinitely in order to decode the received information bits.

[1], [4], [5], [9], [12], [13], and [14] are good references in understanding Viterbi algorithm. The decoding depth is strongly dependent on the merging characteristics of the code [1]. At short decoding depths there are many low-weight unmerged paths which will significantly degrade performance because they have a reasonably high probability of being selected by the output decision device. Satisfactory performance is achieved only at decoding depths at which the large number of low weight unmerged paths dissapear. For convolutional code with code rate $\frac{1}{2}$, the decoding depths need to be about 5k [1].

1.2 Trace-back Method

At high speeds, the technique described above requires all of the shifting to be done in parallel and means that each individual symbol storage device must be equipped with the necessary gates to accept inputs from one of two different locations. For even moderately long decoding spans, this necessitates an enormous amount of hardware and is normally not practical. At very low speeds one may usually design the decoder such that the sequence can be interchanged in a serial fashion through a single routing device.

Fortunately, there is a second method which can be used for high-speed decoders. This is referred to as the trace-back method [11]. Using this technique one does

not store the actual information sequences but instead stores the results of each comparison (the trellis connections). After several branches have been processed, the trellis connections are recalled in the reverse order in which they were stored and a path is traced back through the trellis diagram. Instead of decoding a single branch of information symbols, this technique requires that one decode several branches at a time at high speeds.

In the trace-back method, a mapping for tracing a state at time unit t back to its previous state at time unit t-1 is developed. The trellis connections between the states at time unit t-1 and time unit t for the $\left(2, \frac{1}{2}\right)$ connectional code is shown in Figure 1.6.

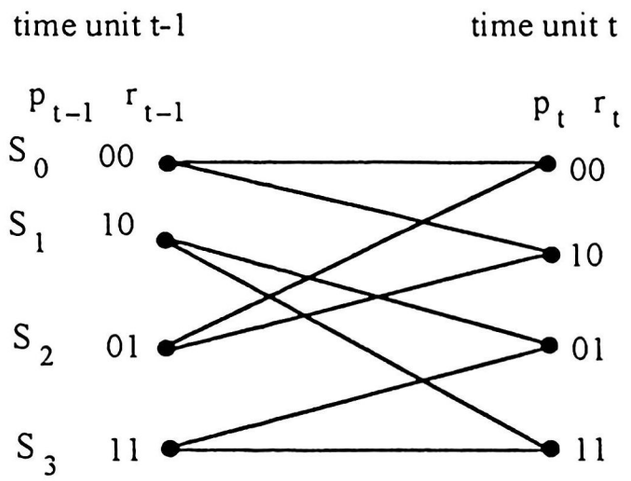

Figure 1.6 - Trellis connections between states

For state S_0 at time unit t, there is an upper branch which comes from state S_0 and a lower branch which come from state S_2 at time unit t-1. Figure 1.6 can also be represented by a logic table such as in Figure 1.7. The upper branch and the lower branch in Figure 1.6 is assigned as "zero" bit "0" and "one" bit "1" respectively in Figure 1.7. The dash notation "__" in Figure 1.7 means there is no connection between the horizontal and vertical states in the diagram.

p_{t-1} r_{t-1} \ p_r r_t	00	10	01	11
00	0	0	-	-
10	-	-	0	0
01	1	1	-	-
11	-	-	1	1

Figure 1.7 - Table of trace-back changes

Let $x_t(m_t)$ denote the above described one-bit function of $S_{m_{t-1}}$ and S_{mt} of the table of Figure 1.7. Since the r_{t-1} of the state vector m_{t-1} is shifted out of the register when state vector changes to m_t (refer to equation 1) then this component needs to be stored for tracing back along the survivor path. $x_t(m_t)$ is actually r_{t-1}. This can be verified by the logic table in Figure 1.7.

Therefore, $p_{t-1} = r_t$ (6a)

$r_{t-1} = x_t(m_t)$ (6b)

Let us define the trace-back mapping function (TB) as follows:-

$$TB : (p_t, r_t, x_t(m_t)) \rightarrow (p_{t-1}, r_{t-1})$$

From equation 6,

$$TB : (p_t, r_t, x_t(m_t)) = (p_{t-1}, r_{t-1}) = (r_t, x_t(m_t)) \qquad (7)$$

The trace-back algorithm is outlined as follows:

Step 1) Let $P_0(0) = 0$ and $P_0(t) = \infty$ for $t \neq 0$

Step 2) Find partial path metric

Store the smallest metric

Determine xt(mt). This step can be represented by the following diagram, Figure 1.8.

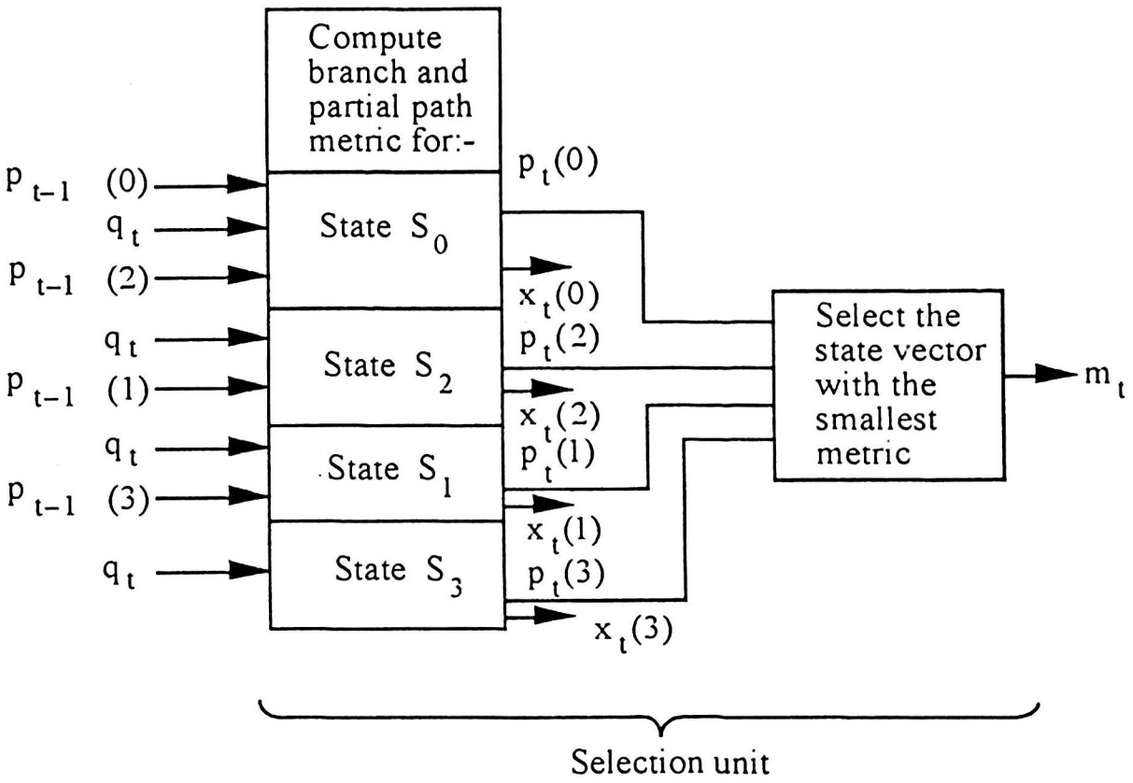

Figure 1.8 - Structure of selection unit

Step 3) Choose the state vector with survivor of smallest metric and trace-back the survivor by using mapping function, TB, to find the decoded bit.

If $j < 5k$ go to step 2)

Otherwise, find an m_t such that $p_t(m_t)$ is minimum.

Then

$$m_t \equiv (p_t, r_t)$$

and $$(p_{t-1}, r_{t-1}) = (r_t, x_t(m_t))$$

Step 4) If j = length of codeword +5K, stop.

If j ≥ length of codeword, then t = t + 1 and go to step 3).

If j < length of codeword, then t = t + 1 and go to step 2).

Once the trace-back procedure is completed at each iteration the decoded bit is identified.

One drawback of this technique is that the decoding delay is 2 to 3 times longer than with register exchange since one cannot initiate the trace-back until several branches past the minimum decoding depth (in the above mentioned convolutional code, 5k).

1.3 A Systolic Viterbi, Trace-back Decoder

A systolic array decoder can provide a continuous trace-back operation in an array of registers in a pipe-line fashion, instead of waiting for the entire trace-back procedure to be completed at each iteration. For this type of decoder, step (3) of the trace-back algorithm mentioned above can be implemented using 10 ie. 5k trace-back units which are connected in the pipeline manner. Each trace-back unit performs the trace-back mapping function on the branches of the survivor path. The structure of the systolic Viterbi decoder is shown in Figure 1.9(a) and the register in the trace-back unit required to process the trace-back algorithm is shown in Figure 1.9(b).

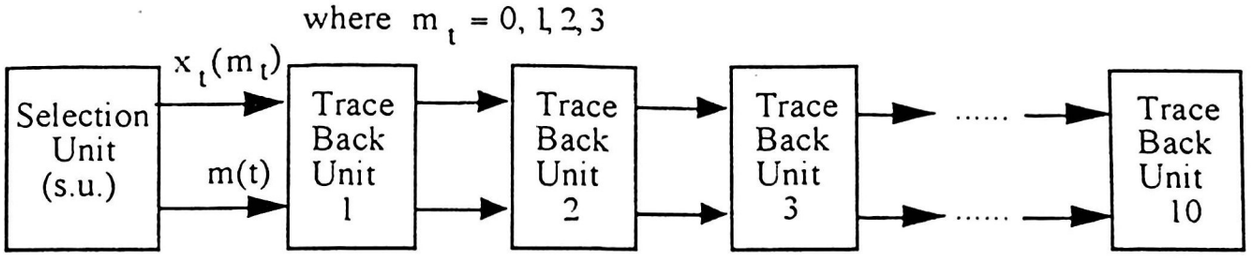

Figure 1.9(a)

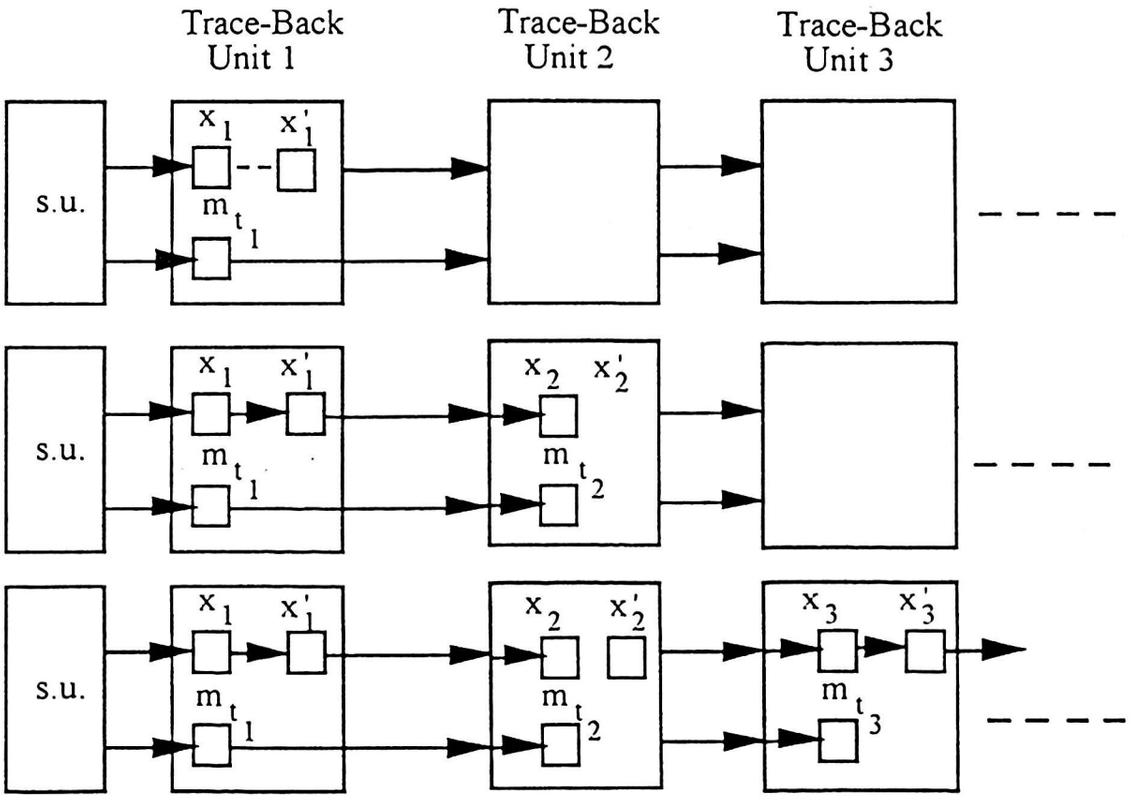

Figure 1.9(b)

Figure 1.9(a) - The structure of the systolic Viterbi decoder

Figure 1.9(b) - The registers in the trace-back unit

Let's say that the decoder is operating at iteration t. Following the trace-back algorithm, the trace-back units will perform sequentially the trace-back mapping for i=t, t-1, t-2, ..., t-5k+1. An extra storage register (x^1 in Figure 1.9(b)) is

needed to store vector x_t as the tracing back of survivor from time t to t-1 is done simultaneously with selecting the survivor forward in time from time t to t+1. Let us explain this in more detail.

Assume that data x_t and m_t are stored in Trace-back Unit 1 in the first row of Figure 1.9(b). This trace-back unit will then produce the trace-back mapping of vectors x_t and m_t in order to find m_{t-1}. These m_{t-1} and x_{t-1} vectors will later be shifted to Trace-back Unit 2 in the second row of Figure 1.9(b). Therefore, an extra storage register is needed to simultaneously store x_{t-1} before being shifted to the second trace-back unit. As each x_t is shifted from the selection unit at each iteration, two registers x_t and x'_t are used for this data shifting. Therefore besides the x_t and m_t registers which are used for storing x_t and the state vector, there is also register x'_t to store the vector x_t of the previous time unit. Thus, each trace-back unit consists of a two 4 bit (2^k bit) registers and one 2 bit (k bit) register.

Finally, at t = 10K = 20, one obtains the first decoded information bit (this bit is equivalent to the first bit in m_{t10}). The remaining bits are decoded one by one after each time unit.

For more detailed explanation on the trace-back systolic array Viterbi decoder, please refer to [11].

CHAPTER 2

ARCHITECTURAL AND LOGIC DESIGN

Below is the block diagram for a general trace-back systolic array Viterbi decoder.

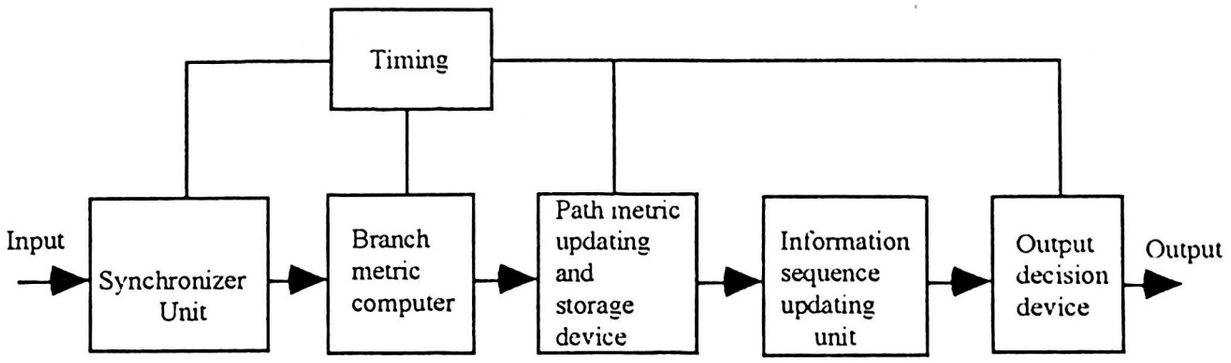

Figure 2.1 - Architectural block diagram for a general trace-back systolic array Viterbi decoder

The decoder consist of:-

1. A synchronizer which provides branch timing information to the decoder;
2. A branch metric computer which determines the apropriate branch metrics for each received code branch;
3. A path metric updating, comparison and storage device in which the branch metrics are added to the previously stored path metrics, the appropriate comparison made, and the new path metrics stored.
4. An output decision device which determines the decoder outputs.

2.1 Path Metric Storage and Updating

Consists of two adders, followed by a comparator and this followed by a storage element. This is called the add-compare-select (ACS) function. For a constraint

length k code, this circuit must be duplicated 2^k times. Therefore, for a code with a constraint length of 2, there are four of such circuit in the decoder [1].

In this project, the codeword length, L_c, of the $\left(2, \frac{1}{2}\right)$ convolutional code is taken as 20. By looking at step (4) of the trace-back algorithm, the decoding process will not stop until $j = L_c + 5k$. In this case, it will only stop at 30. The maximum Hamming weight for each code branch is 2. Therefore if the maximum case is considered, the final path metric will be 2 x 30 = 60. Because of this, 6 bit adders and 7 bit comparators are used in the ACS unit.

The decoder is taken as a parallel input device where for a codeword with a code rate of $\frac{1}{2}$, 2 bits will enter the decoder in a parallel fashion. For example, if the received sequence is q(refer to Figure 1.1) = (00, 01, 11, ...) then for the case of q_1 = 01, 0 and 1 are termed as MSB and LSB respectively.

Figure 2.2 is the block diagram of the ACS unit for one state.

In Figure 2.2, the 4 D flip-flops act as the synchronizer of the circuit. EXCLUSIVE-OR is to determine the difference between the output codeword $\underline{n}$ with the input codeword of the decoder $\underline{q}$. As the branch metris is either 0, 1 or 2, a half-adder is sufficient for determining the branch metric. 6 bit adder is used to produce the path metric and 7 bit comparator is to compare which of the branches is of minimum metric. 7 AND and 7 OR are for selecting the minimum metric 7 FD (7D flip-flops) are used for storing the selected weight. Figure 2.2 is the ACS unit of only one state. Therefore a $\left(2, \frac{1}{2}\right)$ convolutional code decoder consist of four of the above. The logic diagrams for the 6 bit adder and 7 bit comparator are as shown in Figure 2.3 and Figure 2.4 respectively.

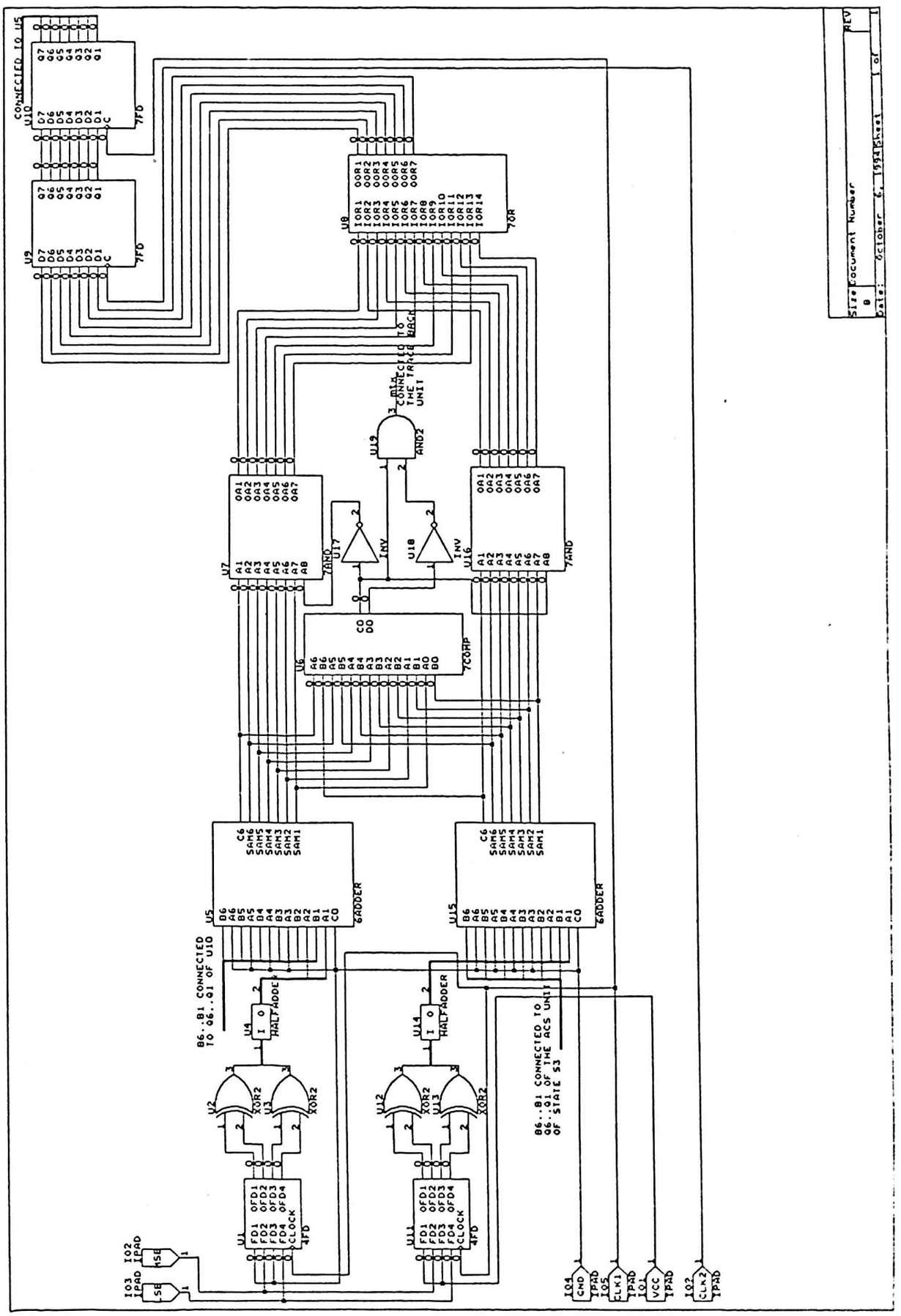

Figure 2.2 - Block diagram of the add-compare-select unit for one state

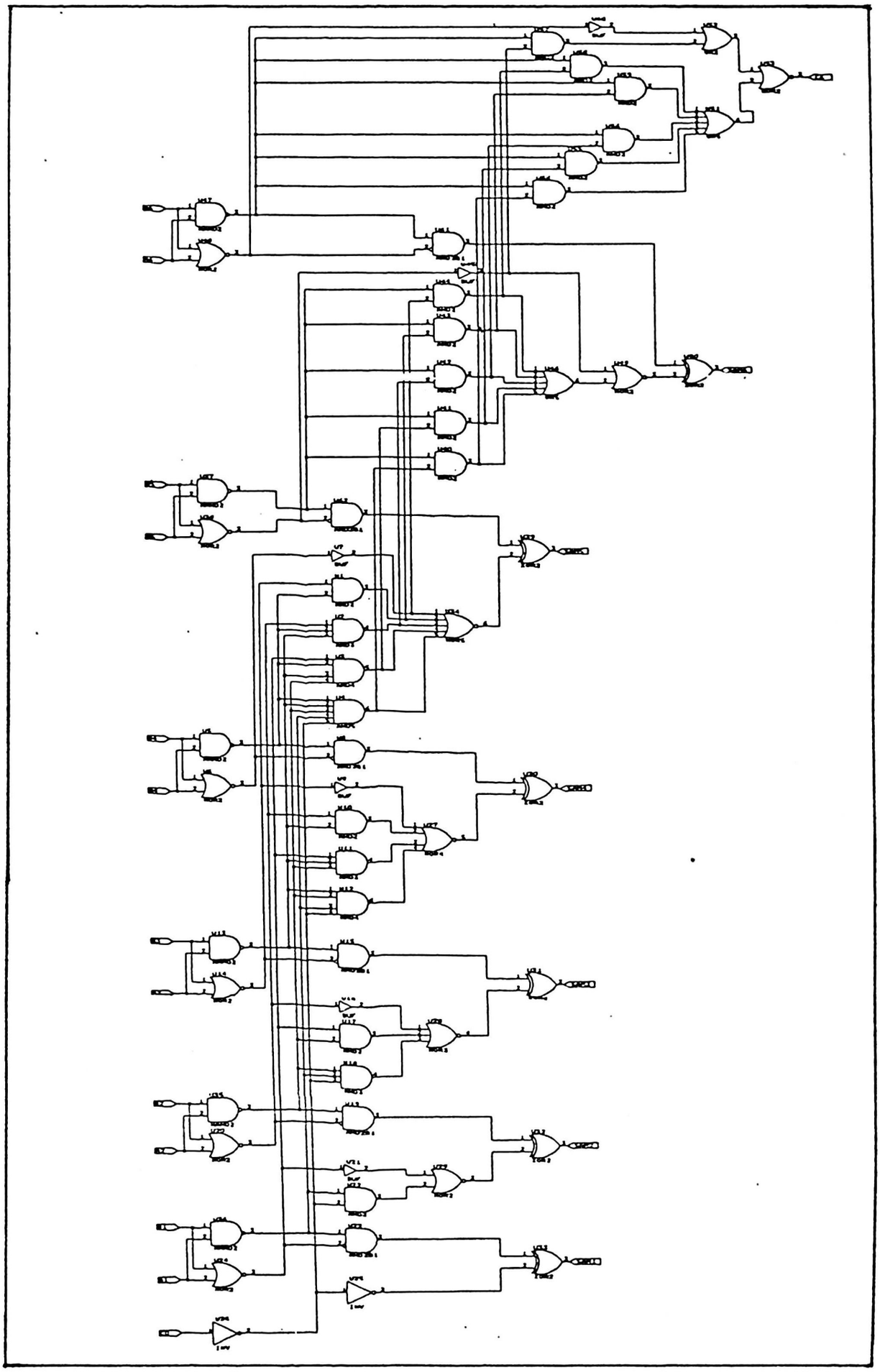

Figure 2.3 - 6 bit adder

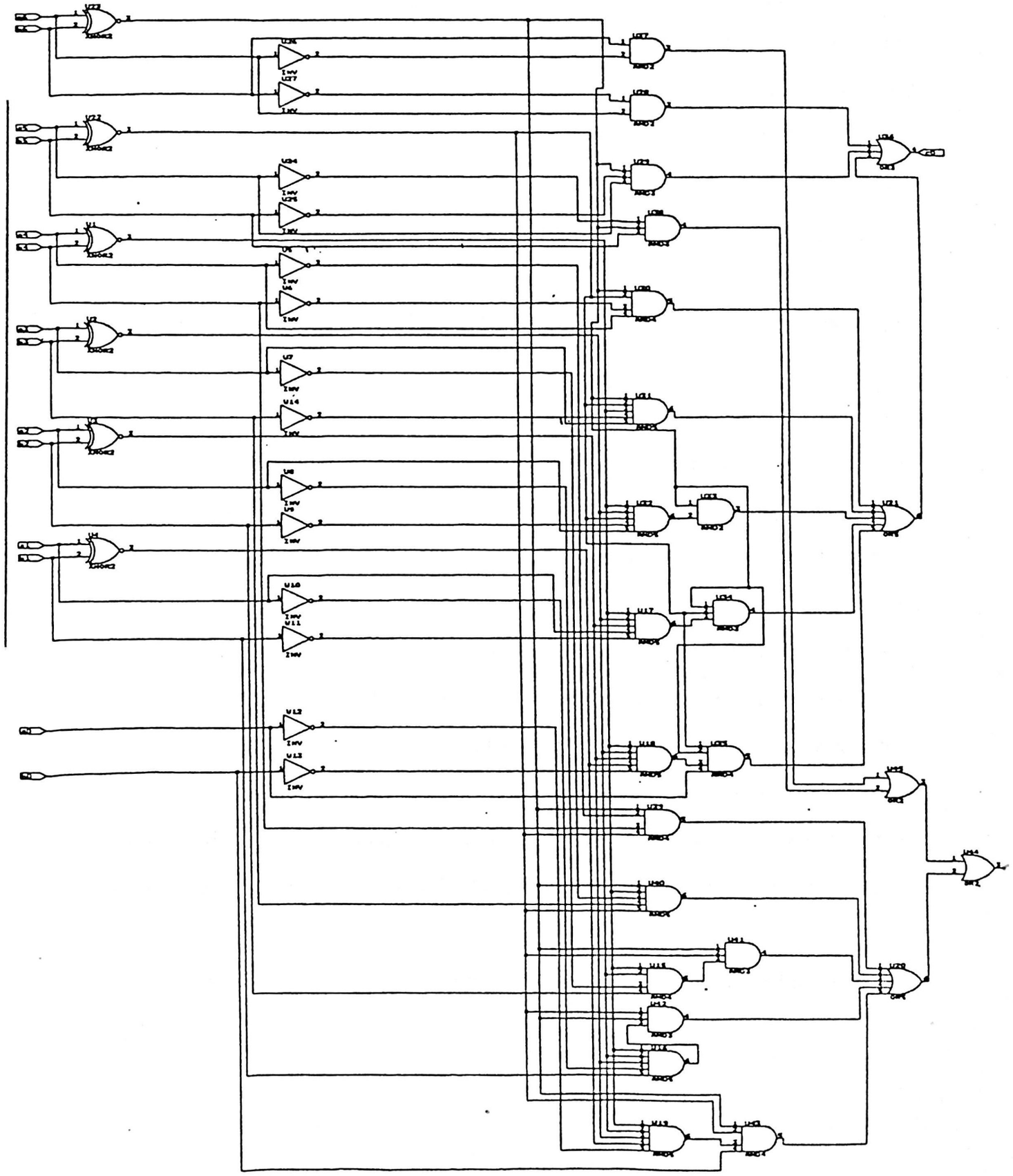

Figure 2.4 - 7 bit comparator

Between two states, there will be another comparator and selecting unit (7 OR and 7 AND) to determine which path of the two states that has a lesser weight. The selected weight will then be compared (by another 7 bit comparator) to the selected weight produced by the selecting unit of the other two states. Besides the weight of the minimum path, the state that produces this minimum metric will be determined and passed on to the trace-back unit. The block diagram of the ACS unit for the whole decoder circuit is given in Figure 2.5.

2.2 Storage and Updating of Hypothesized Information Sequences

Using the trace-back method, the results of each metric comparison (the trellis connections) are stored, which is different from the register exchange technique where a complete hypothesized information sequence are stored for each state. After several branches have been processed, the trellis connections are recalled in the reverse order in which they were stored and a path is traced back through the trellis diagram. Instead of decoding a single branch of information symbols, several branches are decoded at a time at high speeds. The drawback of this technique is that the decoding delay is two to three times longer than the register exchange since the trace-back can only be initiated after several branches past the minimum decoding depth have been processed. The 'rule of thumb' is that the decoding depth needs to be about 5k[1]. In this case, the decoding depth is 10.

To produce the trace-back mapping functin $x_t(m_l)$ the circuit as in Figure 2.6 is used.

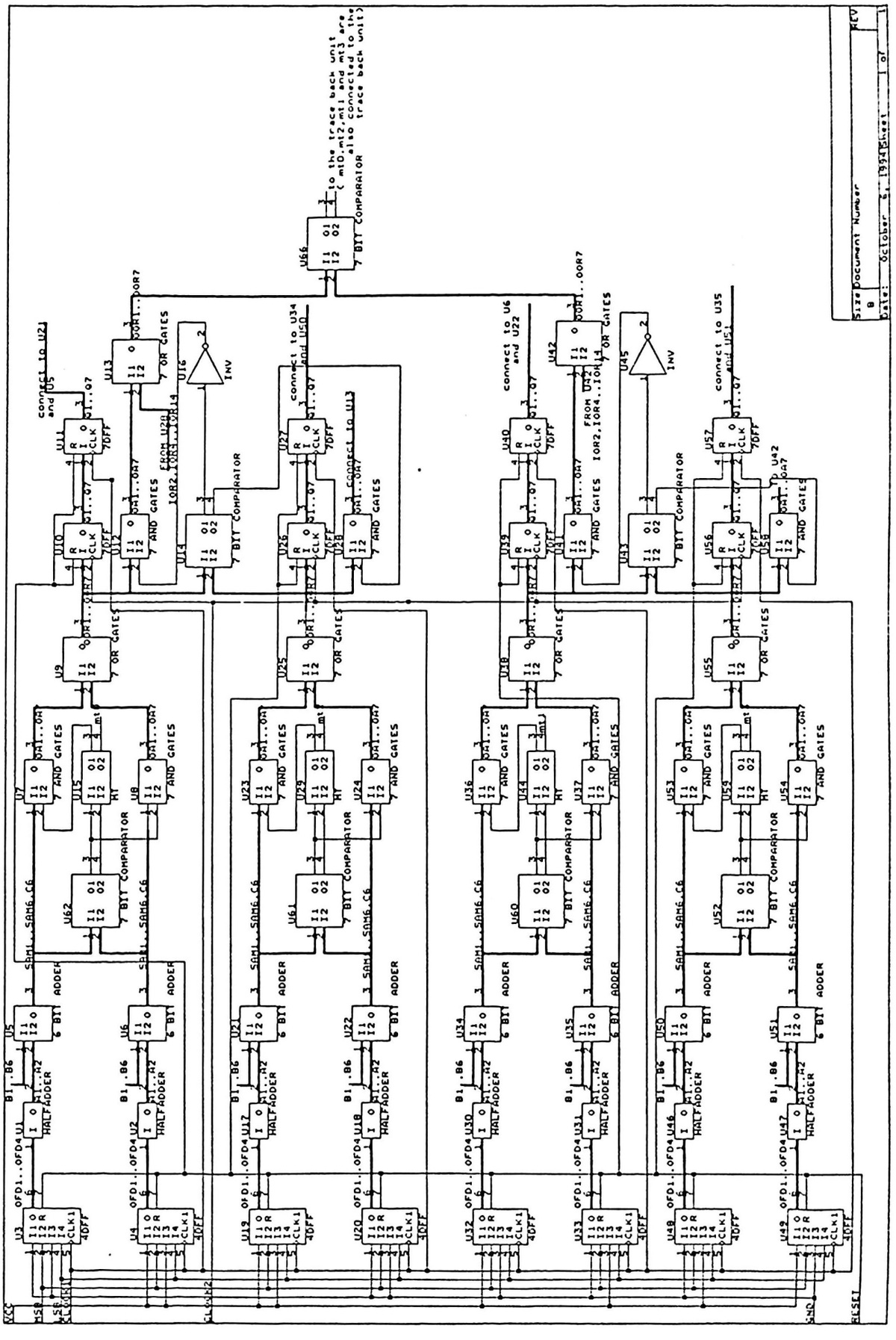

Figure 2.5 - The add-compare-select unit of the whole decoder circuit

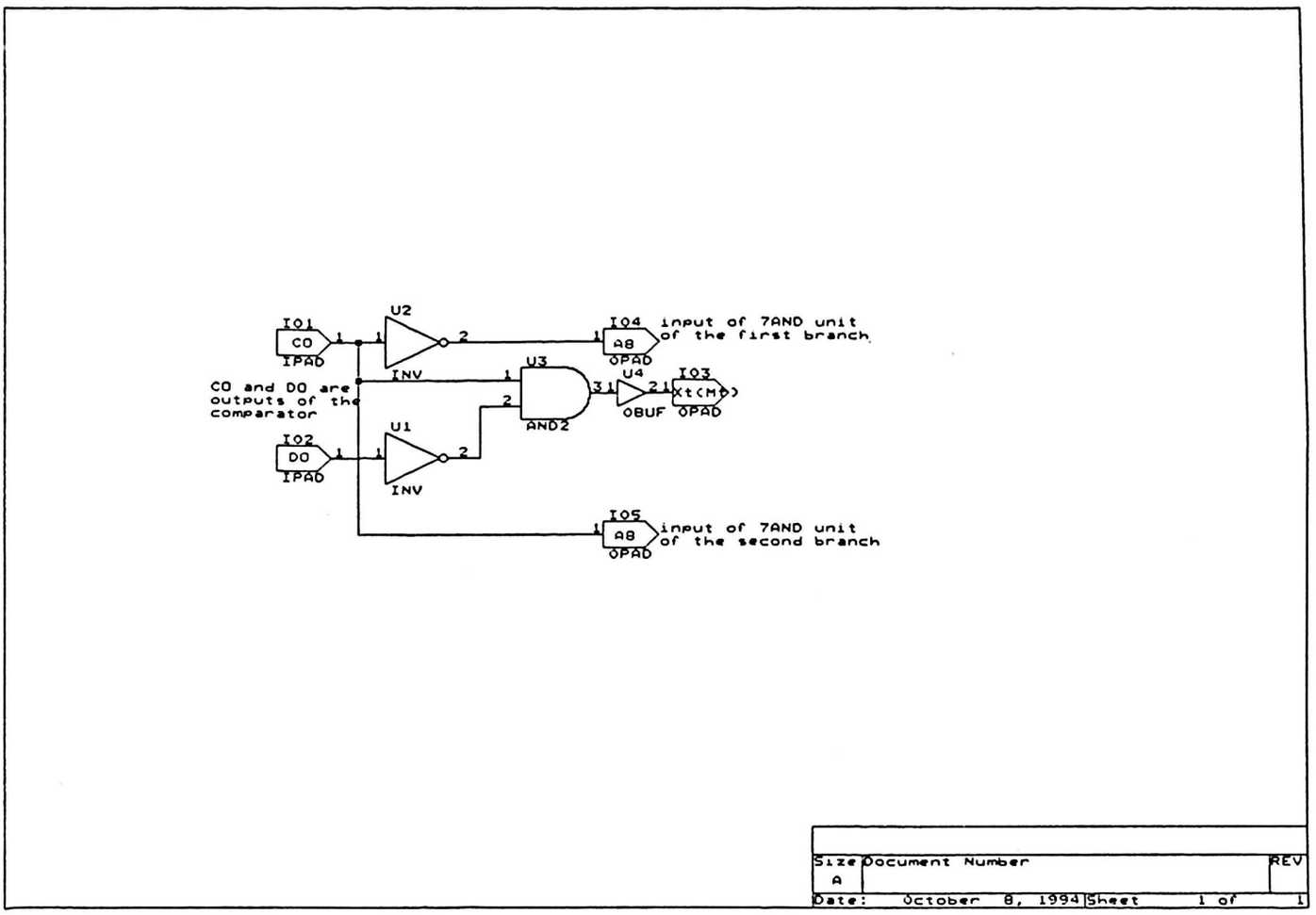

Figure 2.6 - Circuit to produce $x_t(m_t)$ for the trace-back process

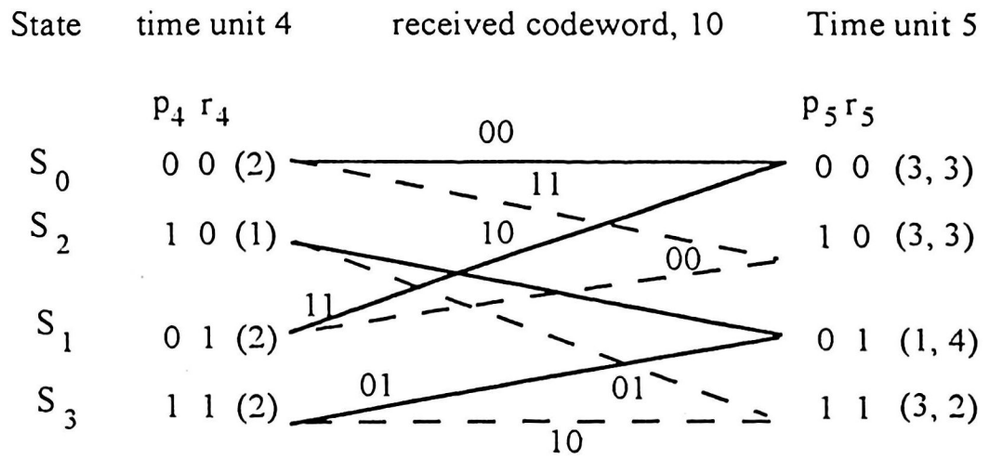

Figure 2.7 - Survivor connections between time units 4 and 5

To explain the circuit, we take an example such as in Figure 2.7. In this example we consider the survivor connections between time unit 4 and 5 (refer to Figure 1.3).

The number in bracket (at the right of p_4r_4 column) indicates the path metric at time unit 4 for each state. Let us assume that the received codeword during this time is 10. The new path metric coming in to each state from both branches are indicated by the numbers in bracket at the right of column p_5r_5. Therefore the new selected path metric at time unit 5 for each state are as follows:-

State	**Selected path metric (at t = 5)**	
S_0	3	$x_5(0) = 1$
S_2	3	$x_5(2) = 1$
S_1	1	$x_5(1) = 0$
S_3	2	$x_5(3) = 1$

$x_5(m_t)$ is determined from the table of trace-back changes (Figure 1.6).

Therefore for state S_1 at time 5, the circuit in Figure 2.7 will produce $x_5(1) = 0$.

The complete trace-back, systolic array Viterbi decoder circuit is shown in Figure 2.9. Only the main circuits that build up the decoder are mentioned and shown in this chapter. Other circuits in the decoder can be found in Appendix A.

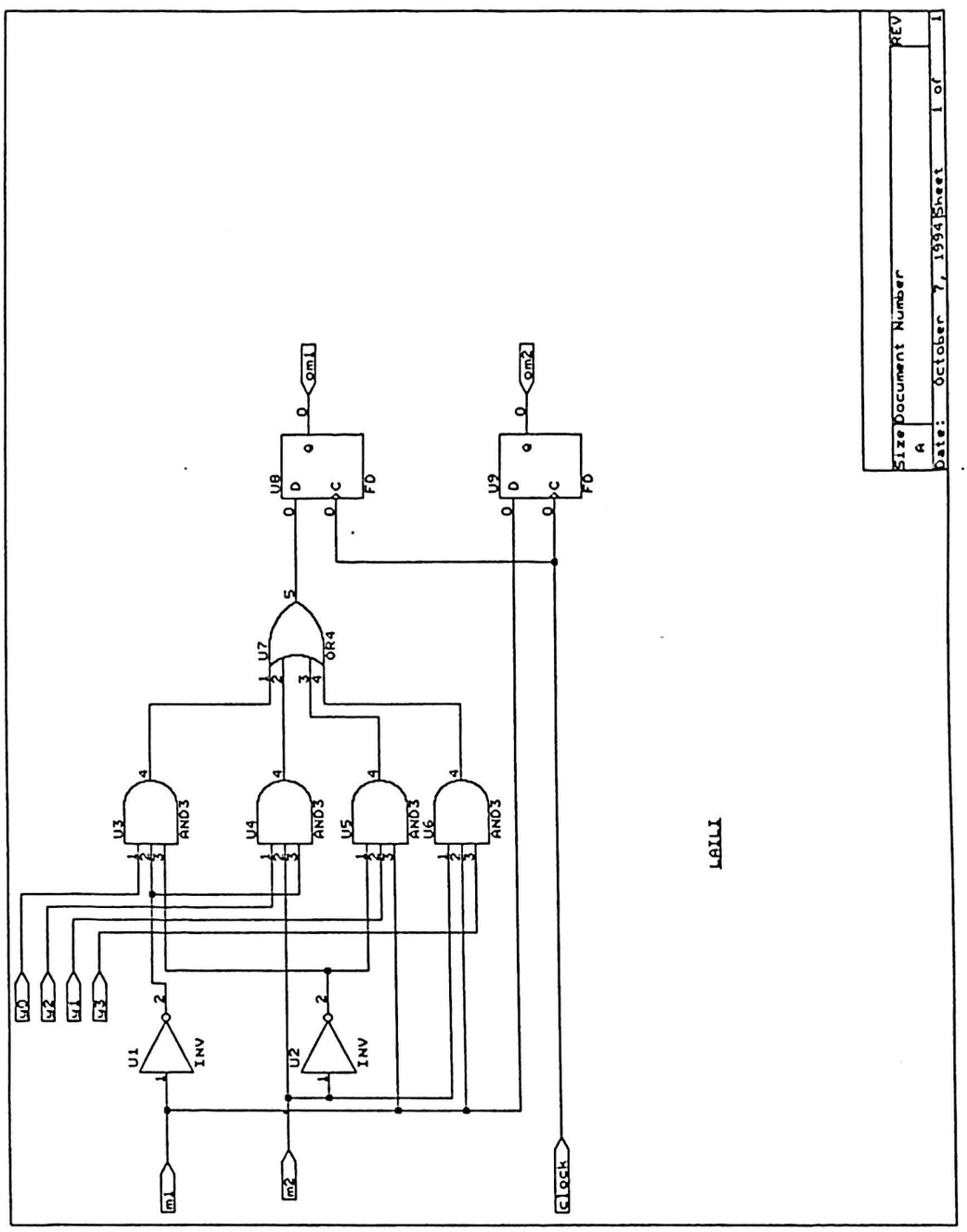

Figure 2.8 - Circuit to produce trace-back mapping function (r_t, $x_t(m_t)$

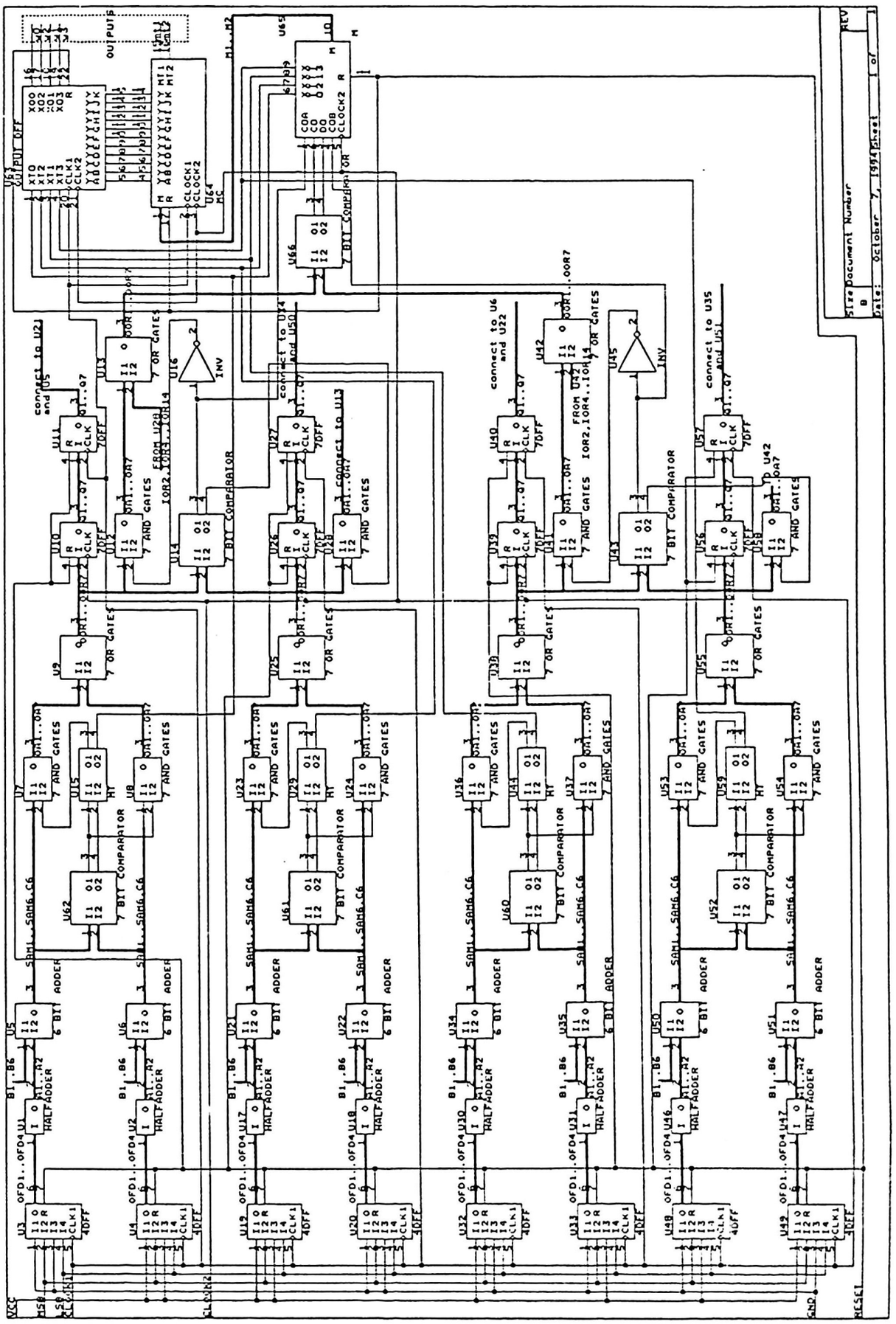

Figure 2.9 - Trace-back, Systolic Array Viterbi decoder circuit