Design and Implementation of a High Speed Parallel Architecture for ATM UNI†

Wen-Yu Tseng, Chin-Chou Chen, David S. L. Wei* and Sy-Yen Kuo

Department of Electrical Engineering
National Taiwan University
Taipei, Taiwan, R.O.C.

*School of Computer Science and Engineering
The University of Aizu
Fukushima, 965-80 Japan

Abstract

In this paper, a parallel architecture is proposed to support the operations described in the ITU-T Recommendation I.432 (B-ISDN user-network interface – Physical layer specification). It is rather difficult to perform these operations on a bit serial architecture at a high rate. This paper demonstrates how these tasks can be achieved by means of parallelism. First, we describe the user-network interfaces in general and their physical layer properties. Then a parallel architecture is proposed with a general translation method which converts the serial operation into the parallel one. The application of the parallel architecture on each function is also depicted and the system has been realized in hardware using CMOS technology.

1 Introduction

There is widespread activity in the telecommunications industry to provide broadband video and data services. Broadband Integrated Services Digital Networks enable high data transfer, video conference, video telephony and any other broadband services based on asynchronous transfer mode (ATM). In ATM, all information to be transmitted is packed in fixed-size (53 octets) cells. These cells have a 48 octet information field and a 5 octet header.

User Network Interface (UNI), which is defined in ITU-T Recommendation I.432 [2], defines the functions of interactions between users and networks. It contains error control and scrambling/descrambling functions. Two interface types were standardized in [2]: the SDH-based interface and the cell-based interface. For the SDH-based interface [3, 4, 5], the ATM cell stream is mapped into the byte frame. The SDH-based B-ISDN UNI implementation has the advantage

†This research was supported by the Telecommunication Laboratory, Taiwan, R.O.C., under the contract TL-84-7205.

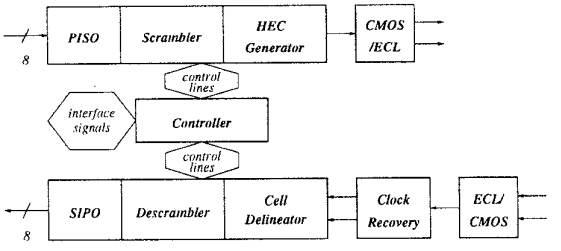

Figure 1: Bit-Serial Architecture of ACUNI Chip.

of full compatibility between the user-network interface and the network-node interface at the expense of the large overhead capacity in the frame. While the cell-based UNI, on which no multiplexing frame structure is imposed, has the advantage of high speed features and efficiency, the cell-based interface itself is not yet fully defined, e.g., coding and allocation of OAM functions are not sufficiently clear. Therefore, the cell-based interface is adopted in the implementation of the chip design. This chip is expected to find important applications in ATM cells multiplexing and demultiplexing prototype as well as in many broadband network systems, such as the link termination of user-network interface, and interconnection interfaces of the ATM multistage switches.

At the physical bit level the B-ISDN user-network interface has a bit rate of 155.52 Mbps or 622.08 Mbps. The original architecture of the ATM cell-based user-network interface (ACUNI) chip in a bit-serial operation is first proposed in [6] as shown in Fig. 1. However, it was shown via simulation and implementation in [6] that this bit-serial architecture can not achieve such high bit rates directly. Thus, a parallel architecture is proposed to accomplish this task.

This paper is organized as follows. The principles of the parallel architecture as well as a general translation method are presented in Sec. 2. The application of the the octet-parallel architecture on each function

1087-4089/96 $5.00 © 1996 IEEE

is depicted in Sec. 3. Sec. 4 shows the parallel architecture and the fabrication data of our ACUNI chip. Finally, the conclusion is given in Sec. 5.

2 Principles of Parallel Architecture

As described in the previous section, two different bit rates are defined; one is 155.52 Mbps and the other is 622.08 Mbps. It is rather difficult to perform those operations on a bit-serial architecture at such a high bit rate of 155.52 Mbps by today's CMOS technology, not to mention the rate of 622.08 Mbps [6]. Therefore, we propose a parallel architecture to support the operations documented in the ITU-T Recommendation I.432.

The parallel architecture is effective in that the decrease in the operating frequency is significant. Take the interface of 622.08 MHz for example, the operating frequency is reduced down to less than 80 MHz if performed octet by octet. As a consequence, this architecture enables the design of the user-network interface by means of field programmable gate arrays or standard cells. In fact, we do use standard cells[‡] to facilitate the design flow.

The only drawback in the parallel architecture is the hardware overhead because more combinational logic gates are required than in the serial architecture. However, since our objective is to achieve the high speed ATM cell-based user-network interface, the additional small area overhead is insignificant and acceptable. Besides, power dissipation may not necessarily increase with the total area as the operating frequency has been lowered down. In most cases, power dissipation can be further reduced with optimization of combinational logic under the parallel architecture.

2.1 Linear Feedback Shift Register

A linear feedback shift register [7] is a circuit consisting of D flip-flops and XOR gates with one or more feedback links. Linear feedback shift registers (LFSRs) are used extensively in many applications. Signature analysis, for example, is a compression technique based upon the concept of cycle redundancy checking (CRC) [8] and is realized in hardware using linear feedback shift registers. In addition, HEC generation, cell delineation, scrambling/descrambling defined in the ITU-T Recommendation I.432 can also be realized with LFSRs. Before we discuss the details of the parallel architecture, some background on the theory and operations of LFSRs are necessary.

[‡]TSMC 0.8 μm CCL standard cell library.

A linear circuit is a logic network constructed from the following basic components:

- unit delays or D flip-flops;
- modulo-2 adders or XOR gates;
- modulo-2 scalar multipliers.

Such a circuit is considered to be linear since it preserves the principle of superposition, i.e., its response to a linear combination of stimuli is the linear combination of the responses of the circuit to the individual stimulus.

In the following, we will first deal with a class of linear circuits, known as autonomous linear feedback shift registers that have no input except for clocks. We define the length l as the number of shift registers in autonomous LFSRs and the state vector Q_k as an l-vector representing the state of the shift registers at time k, i.e.,

$$Q_k = (q_1, q_2, \cdots, q_l)^t, \tag{1}$$

where q_i, $i = 1, 2, \cdots, l$, denotes the value of the i-th register and the superscript means transpose.

The state transition matrix, denoted by $\mathbf{T}$, is defined as an $l \times l$ matrix representing the relationship between the state vectors Q_k and Q_{k+1}, or more specifically,

$$Q_{k+1} = T \cdot Q_k. \tag{2}$$

The extent of parallelism, denoted by n, is defined as the multiple of the data process at one time. Since the autonomous LFSRs has no inputs, it is not necessary to process the data and the autonomous LFSR just proceeds directly to the n-th state. From Eq. (2),

$$Q_{k+n} = T^n \cdot Q_k.$$

From Eq. (1),

$$Q_k = q_1 \cdot e_1 + q_2 \cdot e_2 + \cdots + q_l \cdot e_l,$$

where

$$e_1 = (1, \cdots, 0)^t, e_2 = (0, 1, \cdots, 0)^t, \cdots, e_l = (0, \cdots, 1)^t.$$

Thus,

$$\begin{aligned} Q_{k+n} &= T^n \cdot Q_k \\ &= T^n(q_1 \cdot e_1 + q_2 \cdot e_2 + \cdots + q_l \cdot e_l) \\ &= q_1 \cdot T^n e_1 + q_2 \cdot T^n e_2 + \cdots \\ &\quad + q_{l-1} \cdot T^n e_{l-1} + q_l \cdot T^n e_l \end{aligned} \tag{3}$$

In the above equation, $T^n e_i$, $i = 1, 2, \cdots, l$, can be viewed as the response of the stimulus in each D flip-flop.

Consider the case of LFSR with inputs. We define the sequence $\{x_k\}$ as the input of a LFSR in which x_k is applied at time k. The input can also be regarded as other stimuli, denoted by X, for LFSRs. The stimuli caused by the input in Fig. 2, for example, is '1101'.

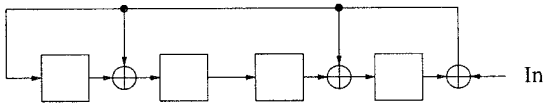

Figure 2: An example input stimulus.

By superposition theorem, the stimuli for the input should be included, or more specifically,

$$\begin{aligned} Q_{k+n} &= q_1 \cdot T^n e_1 + q_2 \cdot T^n e_2 + \cdots + q_l \cdot T^n e_l \\ &\quad + x_k \cdot T^{n-1} X + x_{k+1} \cdot T^{n-2} X + \cdots \\ &\quad + x_{k+n-2} \cdot TX + x_{k+n-1} \cdot X. \end{aligned} \tag{4}$$

In the next section, we will explain the application of these formulas in detail.

Up to this point, it is assumed, for simplicity, that the LFSRs have no output. Now, the output shall be taken into account. We define the sequence $\{z_i\}$ as the output sequence of LFSRs and the generating vector H as an l-vector representing the relationship between the state vector Q_k and the output sequence $\{z_k\}$, or more specifically,

$$z_k = H \cdot Q_k. \tag{5}$$

For a Mealy machine, the output z_k is not only a function of the present state Q_k but also a function of the present input x_k. Since LFSRs preserve the principle of superposition, Eq. (5) can be written as

$$z_k = H \cdot Q_k + x_k. \tag{6}$$

For a complete parallel architecture of LFSR, the output sequence $\{z_k, z_{k+1}, \cdots, z_{k+n-1}\}$ is also necessary in addition to Q_{k+n}.

2.2 Other Sequential Circuits

So far as the nonlinear sequential circuit is concerned, there exists a similar method to convert the nonlinear circuit in serial mode into the one in parallel mode. Instead of deriving explicit equations, we briefly describe the general translation procedure. First, suppose that the state vector of the nonlinear sequential circuit is denoted by Q_k and the input sequence is $\{x_k\}$. The next state Q_{k+1} can be obtained in terms of Q_k and x_k. Then Q_{k+2} can be obtained from Q_{k+1} and expressed in terms of Q_k and $\{x_k, x_{k+1}\}$. Repeat the above procedure until Q_{k+n} represented in terms of Q_k and $\{x_k, x_{k+1}, \cdots, x_{k+n-1}\}$ is obtained. Meanwhile, the output sequence $\{z_k, z_{k+1}, \cdots, z_{k+n-1}\}$ will be calculated directly from the corresponding states of $\{Q_k, Q_{k+1}, \cdots, Q_{k+n-1}\}$. Of course, the output sequence $\{z_k, z_{k+1}, \cdots, z_{k+n-1}\}$ itself should be expressed in terms of Q_k and $\{x_k, x_{k+1}, \cdots, x_{k+n-1}\}$. In one word, the parallel architecture of the general sequential network is constructed based upon the following results in terms of the state Q_k and the input sequence $\{x_k, x_{k+1}, \cdots, x_{k+n-1}\}$:

- the next n-th state Q_{k+n};
- the output sequence $\{z_k, z_{k+1}, \cdots, z_{k+n-1}\}$.

The most important difference from LFSRs is that combination logic circuits after translation often become more complicated than those in LFSRs and optimization is required to simplify the equations.

3. Applications

3.1 Scrambler for Cell-Based UNI

The distributed sample scrambler [9] is recommended for the cell-based user-network interface. It is a class of scrambler in which randomization of the transmitted data stream is achieved by modulo-2 addition of a pseudo random sequence. The pseudo random sequence polynomial is $x^{31} + x^{28} + 1$, as shown in Fig. 3. For our purpose, we choose to skip 7 states and

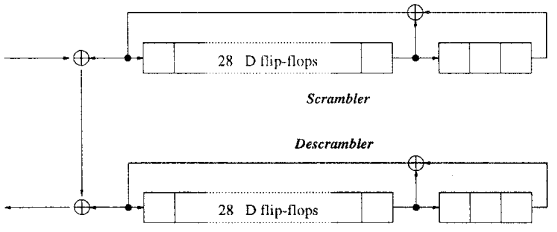

Figure 3: Distributed Sample Scrambler / Descrambler.

jump to every 8th state since the scrambler operation is desired to be performed in parallel on an octet of 8 bits. According to Eq. (6), the output sequence is

$$z_{k+i} = q_{28-i} + q_{31-i} + x_{k+i}, \quad i = 0, 1, \ldots, 7,$$

where q_i is the value in the corresponding D flip-flop.

While Q_{k+n} in Eq. (4) can be obtained by the direct calculation of T^n, another short-cut method is adopted to simplify the calculation of T^n. The key

point behind this method is that each flip-flop in LFSRs can be treated separately and be combined later by the superposition theorem. For example, the value of the first (left-most) D flip-flop in Fig. 3 will be *moved* to the 9th D flip-flop after 8 clock cycles, i.e., q_9^+ is a function of q_1. Another example is that the value of the 28th D flip-flop will be *distributed* to both the 5th and the 8th D flip-flops. The next state transition logic is:

$$\begin{aligned} q_i^+ &= q_{i+20} + q_{i+23} + x_{k+8-i}, \quad i = 1, 2, \ldots, 8, \\ q_i^+ &= q_{i-8}, \qquad\qquad\qquad\quad i = 9, 10, \ldots, 31, \end{aligned}$$

All situations are listed in the form of the stimulus-response pair in Table 1. Note that in the parallelized distributed sample scrambler, the correction vector should be modified to align on the octet boundary.

Table 1: Next State of Distributed Sample Scram bler in Parallel.

Stimulus	Response
q_1	00000000100000000000000000000000
q_2	00000000010000000000000000000000
q_3	00000000001000000000000000000000
q_4	00000000000100000000000000000000
q_5	00000000000010000000000000000000
q_6	00000000000001000000000000000000
q_7	00000000000000100000000000000000
q_8	00000000000000010000000000000000
q_9	00000000000000001000000000000000
q_{10}	00000000000000000100000000000000
q_{11}	00000000000000000010000000000000
q_{12}	00000000000000000001000000000000
q_{13}	00000000000000000000100000000000
q_{14}	00000000000000000000010000000000
q_{15}	00000000000000000000001000000000
q_{16}	00000000000000000000000100000000
q_{17}	00000000000000000000000010000000
q_{18}	00000000000000000000000001000000
q_{19}	00000000000000000000000000100000
q_{20}	00000000000000000000000000010000
q_{21}	00000000000000000000000000001000
q_{22}	10000000000000000000000000000100
q_{23}	01000000000000000000000000000010
q_{24}	00100000000000000000000000000001
q_{25}	10010000000000000000000000000000
q_{26}	01001000000000000000000000000000
q_{27}	00100100000000000000000000000000
q_{28}	00010010000000000000000000000000
q_{29}	00001001000000000000000000000000
q_{30}	00000100100000000000000000000000
q_{31}	00000010010000000000000000000000

3.2 Scrambler for SDH-Based UNI

The self synchronizing scrambler with generating polynomial $x^{43} + 1$ has been identified for the SDH-based physical layer. The scrambler and descrambler are slightly different from each other and are shown in Fig. 4.

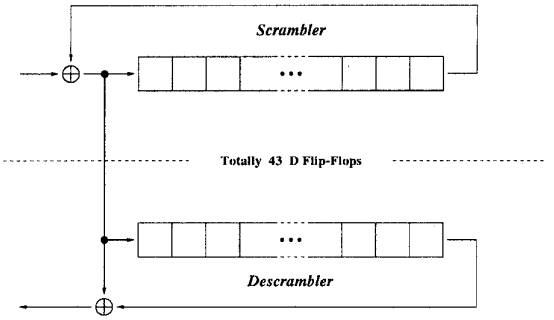

Figure 4: Self Synchronous Scrambler/Descrambler.

The output sequence $\{z_k, z_{k+1}, \cdots, z_{k+7}\}$ of the scrambler/descrambler is

$$z_{k+i} = q_{43-i} + x_{k+i}, \quad i = 0, 1, \ldots, 7,$$

The next state transition logic is:

$$\begin{aligned} q_i^+ &= q_{i+35} + x_{k+8-i}, \quad i = 1, 2, \ldots, 8, \\ q_i^+ &= q_{i-8}, \qquad\qquad\quad i = 9, 10, \ldots, 43, \end{aligned}$$

The next-state table of the self synchronous scrambler in the form of stimulus-response pair, as depicted in the above subsection, is omitted for clarity.

3.3 HEC Generation

The header error control (HEC) is an 8-bit sequence. It will be the remainder of the division (modulo 2) by the generator polynomial $x^8 + x^2 + x + 1$ of the product x^8 multiplied by the content of the header excluding the HEC field, to which the fixed pattern '01010101' will be added. The HEC generator is shown in Fig. 5.

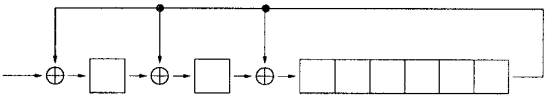

Figure 5: HEC Generator.

Based on the proposed translation method mentioned above, the next state of the octet-parallel HEC generator is shown in Table 2. An observation on Table 2 reveals that the responses are in regular sequence if each row is regarded as a state of the HEC generator. It is easily seen that each row is the very next sate of the above row. This can be verified by the following equations. For the HEC generator, $X = e_1$ and $e_{i+1} = Te_i$. From Eq. (4), we can obtain

$$\begin{aligned} Q_{k+n} &= q_8 \cdot T^{15}e_1 + \cdots + q_1 \cdot T^8 e_l + x_k \cdot T^7 e_1 \\ &\quad + \cdots + x_{k+6} \cdot Te_1 + x_{k+7} \cdot e_1. \end{aligned} \tag{7}$$

Table 2: Next State of HEC Generator in Parallel.

Stimulus	Response	Stimulus	Response
x_{k+7}	10000000	q_1	11100000
x_{k+6}	01000000	q_2	01110000
x_{k+5}	00100000	q_3	00111000
x_{k+4}	00010000	q_4	00011100
x_{k+3}	00001000	q_5	00001110
x_{k+2}	00000100	q_6	00000111
x_{k+1}	00000010	q_7	11100011
x_k	00000001	q_8	10010001

Table 1 is slightly different due to the feedback link between the 28th and the 29th D flip-flops.

3.4 Cell Delineation

Although based on the same polynomial x^8+x^2+x+1 theoretically, cell delineation is somewhat different from HEC generation. Before performing the division, The LFSR computing the remainder shall reset its register to all 0s. Then 40 bits of header are applied in serial and the remainder is left in the register. After that, the LFSR will be reset again in order to compute another remainder. Fig. 6 illustrates this situation[§]. On the other hand, cell delineation will be

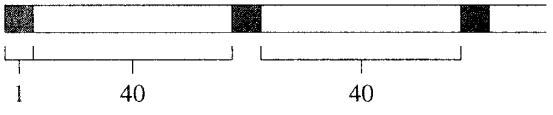

Figure 6: Fragmental Cell Delineation – Serial.

performed bit by bit as shown in Fig. 7.

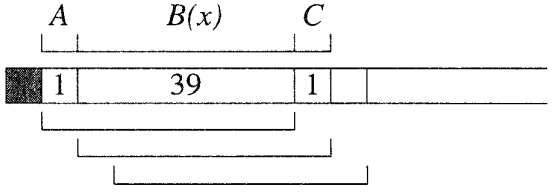

Figure 7: Continuous Cell Delineation – Serial.

Obviously, the original HEC generator should be modified so as to fit our need. Let $P(x)=x^8+x^2+x+1$ and A, $B(x)$, C as shown in Fig. 7. Assume that the remainder of 40 bits of data after the initialization is $R(x)$, i.e.,

$$\left[Ax^{39}+B(x)\right]_{P(x)}=R(x),$$

[§]The shaded box is the time slot with reset.

in which $[\cdot]_{P(x)}$ denotes the remainder of the division by the polynomial $P(x)$. At the next clock cycle, the values $R^+(x)$ left in the register will be

$$\left[Ax^{40}+x\cdot B(x)+C\right]_{P(x)}.$$

Clearly, only $x\cdot B(x)+C$ is desired and Ax^{40} should be discarded. Therefore,

$$\begin{aligned}&[x\cdot B(x)+C]_{P(x)}\\ &\quad=\ R^+(x)+\left[Ax^{40}\right]_{P(x)}\\ &\quad=\ R^+(x)+A(x^6+x^5+x).\end{aligned}\qquad(8)$$

From Eq. (8), we can obtain the continuous HEC checker (bit by bit) shown in Fig. 8.

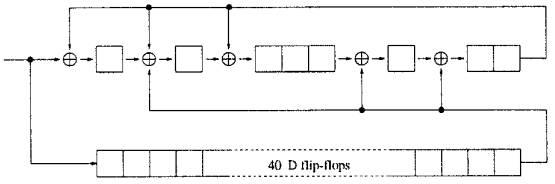

Figure 8: Continuous HEC Checker.

For completeness and later use, consider the continuous HEC checker in a *backward* way. Suppose that

$$[x\cdot B(x)+C]_{P(x)}=R(x).$$

The remainder of B(x) can be calculated using the formula

$$[B(x)]_{P(x)}=x^{-1}\left[R(x)+C\right],$$

where x^{-1} denotes the reverse operation of LFSR. For example, the reverse operation of the original HEC generator is shown in Fig. 9.

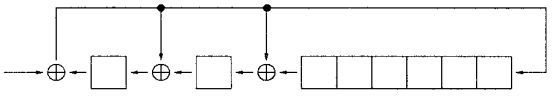

Figure 9: Reversed HEC Generator.

In the previous clock cycle, the remainder left in the register was

$$\begin{aligned}&\left[Ax^{39}+B(x)\right]_{P(x)}\\ &\quad=\ x^{-1}\left[R(x)+C\right]+\left[Ax^{39}\right]_{P(x)}\\ &\quad=\ x^{-1}\left[R(x)+C\right]+A(x^5+x^4+1).\end{aligned}\qquad(9)$$

The continuous HEC generator/checker shown in Fig. 8 is in serial mode and it is desired to convert it

into an octet-parallel HEC generator/checker. While the translation method depicted in the previous section can be applied to it directly, we describe another method which achieves the the same goal with reduced complexity and shorter delays instead.

First, convert the original HEC generator in bit-serial mode into an octet-parallel one depicted in Sec. 3.3. After translation, the HEC generator will be operated in parallel as shown if Fig. 10. Remember

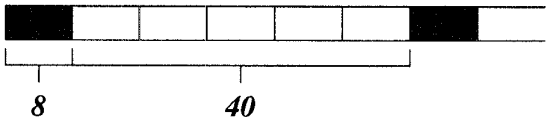

Figure 10: Fragmental Cell Delineation – Parallel.

that cell delineation will be performed bit by bit. The above octet-parallel HEC generator should be further modified as shown in Fig. 11 for the first step. Below

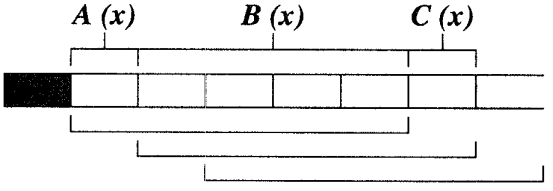

Figure 11: Continuous Cell Delineation – Parallel.

is a list of necessary equations to support continuous cell delineation in parallel depicted in Fig. 11. Suppose that

$$\left[A(x)\cdot x^{32}+B(x)\right]_{P(x)}=R(x).$$

At the next clock cycle (in parallel), the values $R^{+}(x)$ left in the register will be

$$\left[A(x)\cdot x^{40}+x^{8}\cdot B(x)+C(x)\right]_{P(x)}.$$

Thus, we can obtain the remainder of $x^{8}\cdot B(x)+C(x)$ from

$$\left[x^{8}\cdot B(x)+C(x)\right]_{P(x)}=R^{+}(x)+\left[A(x)\cdot x^{40}\right]_{P(x)}.$$

There is still seven remainders left blank between the octet boundary. In order to carry it out, more computations are necessary. In Fig. 12, assume that $r(x)$ and $R(x)$ are the remainders of the corresponding data stream, respectively. It is desired that the seven remainders between the octet boundary should be expressed in terms of $r(x)$ and $R(x)$ together with $\{x_{47},x_{46},\cdots,x_{40},x_{7},x_{6},\cdots,x_{0}\}$. From Eq. (8), four of the remainders are obtained and listed in Table 3.

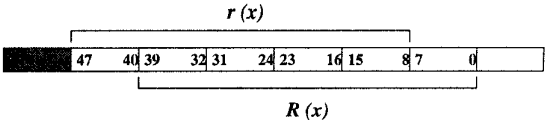

Figure 12: Full Cell Delineation – Parallel.

By Eq. (9), three other remainders are obtained and listed in Table 4. It is seen that the number of variables increases rapidly downward both in Table 3 and 4. The increase in the number of variables means more fan-ins and fan-outs in hardware, which should be avoided in implementation. That's the reason why we use the forward and the backward method simultaneously to reduce the complexity of the hardware and to achieve better performance.

4 Implementation

The parallel architecture of the transceiver chip for the ATM cell-based UNI is shown in Fig. 13. The operations of the scrambler/descrambler, HEC generator, and cell delineator are all explained in the previous section. PISO (Parallel In Serial Out) and SIPO (Serial In Parallel Out) perform the serial-parallel conversions. The clock recovery circuit [10] retrieves the correct clock signal from the bit-stream of data. The ECL/CMOS and CMOS/ECL pads [11] perform CMOS/pseudo-ECL voltage level conversions, which are necessary in order to satisfy the high-speed requirement. Table 5 summarizes the features of our parallel ACUNI chip.

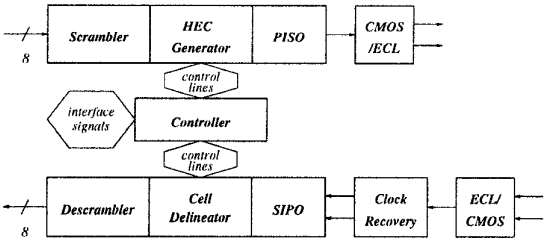

Figure 13: Parallel Architecture of ACUNI.

5 Conclusions

We have proposed a parallel architecture to accomplish the operations stated in the ITU-T Recommendation I.432. The principles of each function are explained and parallelized by translation method. According to the experimental results, we know that parallel architecture is really effective and reduces the

Table 3: Full Cell Delineation – Forward Method.

r_0	r_1	r_2	r_3	r_4	r_5	r_6	r_7
r_7	$r_7 + r_0$	$r_7 + r_1$	r_2	r_3	r_4	r_5	r_6
x_7	x_{47}				x_{47}	x_{47}	
r_6	$r_7 + r_6$	$r_7 + r_6 + r_0$	$r_7 + r_1$	r_2	r_3	r_4	r_5
x_6	$x_{46} + x_7$	x_{47}			x_{46}	$x_{47} + x_{46}$	x_{47}
r_5	$r_5 + r_6$	$r_7 + r_6 + r_5$	$r_7 + r_6 + r_0$	$r_7 + r_1$	r_2	r_3	r_4
$x_{47} + x_5$	$x_{47} + x_{45} + x_6$	$x_{47} + x_{46} + x_7$	x_{47}		x_{45}	$x_{46} + x_{45}$	$x_{47} + x_{46}$
r_4	$r_5 + r_4$	$r_4 + r_6 + r_5$	$r_7 + r_6 + r_5$	$r_7 + r_6 + r_0$	$r_7 + r_1$	r_2	r_3
$x_{47} + x_{46} + x_4$	$x_{46} + x_{44} + x_5$	$x_{46} + x_{45} + x_6$	$x_{47} + x_{46} + x_7$	x_{47}	x_{44}	$x_{45} + x_{44}$	$x_{46} + x_{45}$

Table 4: Full Cell Delineation – Backward Method.

R_0	R_1	R_2	R_3	R_4	R_5	R_6	R_7
$R_1 + R_0$	$R_2 + R_0$	R_3	R_4	R_5	R_6	R_7	R_0
$x_{40} + x_0$	x_0			x_{40}	x_{40}		x_0
$R_1 + R_2$	$R_3 + R_1 + R_0$	R_4	R_5	R_6	R_7	R_0	$R_1 + R_0$
$x_{41} + x_{40} + x_1$	$x_{41} + x_1 + x_0$		x_{40}	$x_{40} + x_{41}$	x_{41}	x_0	$x_{40} + x_1 + x_0$
$R_3 + R_2 + R_0$	$R_4 + R_2 + R_1$	R_5	R_6	R_7	R_0	$R_1 + R_0$	$R_2 + R_1$
$x_{42} + x_{41} + x_2 + x_0$	$x_{41} + x_{40} + x_2 + x_1$	x_{40}	$x_{41} + x_{40}$	$x_{42} + x_{41}$	$x_{42} + x_0$	$x_{40} + x_1 + x_0$	$x_{41} + x_{40} + x_2 + x_1$

Table 5: Specifications and Features of Parallel ACUNI Chip.

IC Process	0.8 μm SPDM CMOS
Data Rate	155.52 Mbps (STM-1)
Chip area	$4331\mu m \times 4277\mu m$
Transistor count	13317
Package	PGA
Pin count	68
Operation frequency	190 MHz

operating frequency significantly, with the hardware overhead.

References

[1] R. Händel and M. N. Huber, *Integrated Broadband Networks – An Introduction to ATM- Based Networks,* Addison-Wesley, 1991.

[2] ITU-T Recommendation I.432, "B-ISDN User-Network Interface – Physical Layer Specification," 1991.

[3] ITU-T Recommendation G.707, "Synchronous Digital Hierarchy Bit Rates," *Blue Book,* 1989.

[4] ITU-T Recommendation G.708, "Network Node Interface for the Synchronous Digital Hierarchy," *Blue Book,* 1989.

[5] ITU-T Recommendation G.709, "Synchronous Multiplexing Structure," *Blue Book,* 1989.

[6] Ing-Yi Chen, Chin-Chou Chen, Wen-Jay Lin and Sy-Yen Kuo, "ACUNI: A 155.52Mbit/s Transceiver Chip for ATM Cell-Based User-Network Interface," *Proc. of the Fifth VLSI Design/CAD Symposium,* 1994.

[7] S. W. Golomb, *Shift Register Sequences,* rev. ed., Aegean Park Press, Laguna Hills, Californa, 1982.

[8] W. W. Peterson and E. J. Weldon, Jr., *Error-Correcting Codes,* 2nd ed. MIT Press, Cambridge, Massachussetts, 1972.

[9] Seok Chang Kim, Byeong Gi Lee, "Synchronization of Shift Register Generators in Distributed Sample Scramblers," *IEEE Transactions on Communications,* Vol. 42, pp. 1400-1408, 1994.

[10] Banu, M., A. E. Dunlop, "Clock Recovery Circuits with Instantaneous Locking," *Electronics Letters,* Vol.28, pp. 2127-2130, Nov, 1992.

[11] M. S. J. Steyaert, W. Bijker, P. Vorenkamp and J. Sevenhans, "ECL-CMOS and CMOS-ECL Interface in $1.2\mu m$ CMOS for 150MHz Digital ECL Data Transmission Systems," *IEEE Journal of Solid-State Circuits,* Vol 26, pp. 18-23, January, 1991.