A Power-Efficient Duplex Communication System

S.B. Furber†, A. Efthymiou† and Montek Singh‡,

†Department of Computer Science, The University of Manchester,
Oxford Road, Manchester M13 9PL, UK.

‡Department of Computer Science, Columbia University, New York, NY 10027, USA.

Abstract

A delay-insensitive inter-chip communication system is proposed that optimises pin- and power-efficiency. When data is being transmitted bidirectionally, a data symbol passed in one direction is acknowledged by a data symbol passing in the other direction. When communication is unidirectional, a simple acknowledge protocol is used. If no data communication is taking place the system is quiescent. Either end may initiate communication, and the situation where both ends attempt to initiate communication at the same time is addressed and resolved in a way that is insensitive to the inter-chip delays.

1: Introduction

It is a common requirement in a multi-chip system to have an efficient mechanism for passing information between chips. Conventional microprocessor buses are effective for connecting memory and peripheral chips to a processor and a DMA controller, but contention for bus bandwidth limits the scalability of such a system.

Where greater scalability is required, such as in a massively parallel computer system, point-to-point communication channels are often used [1]. As each chip in such a system is connected to several neighbours, each channel should use as few pins as are necessary to deliver the required throughput. As off-chip capacitive loads are high, a low-power communication system will attempt to minimise the switching rate of inter-chip wires. Power-efficiency gains can also be made at the electrical signalling level, for example by reducing voltage swings [2], but these gains are in addition to the switching optimizations considered here.

This paper presents a novel inter-chip communication system that uses delay-insensitive off-chip behaviour to give reliable operation and attempts to optimise power- and pin-efficiency through the use of appropriate protocols and encodings. The control for the system presents an interesting illustration of asynchronous design techniques.

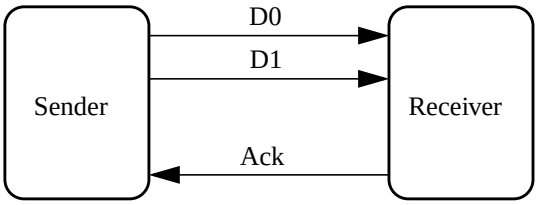

Figure 1: Unidirectional dual-rail channel

2: Delay-insensitive communication

There are many ways that information can be communicated between chips, but a delay-insensitive (‘DI’) communication system may be preferred as it allows a very flexible physical organisation of large systems of chips. A minimal unidirectional DI communication channel comprises two wires from the sender to the receiver and one wire in the return direction (see figure 1). The two transmit wires send data using dual-rail encoding, so a zero is sent by raising the voltage on one of the wires, ‘D0’, and a one by raising the voltage on the other wire, ‘D1’. Each bit of data is acknowledge by a high on the return acknowledge (‘Ack’) wire, and then the wires are restored to their original levels in a return-to-zero (‘RTZ’) cycle: first the data wire that was raised returns to zero, then this is acknowledged by Ack returning to zero. The system is then back in its initial state and ready to send the next bit of data.

2.1: Transition signalling

An alternative to RTZ encoding is to use non-return-to-zero (‘NRZ’) or transition encoding. Here a zero is sent as a transition on the wire D0. This will be a rising transition if the wire was previously low, or a falling transition if the wire was previously high. The information is in the transition, not the voltage level on the wire. Such a scheme has obvious advantages if each off-chip transition has a significant energy cost as it halves the number of transitions required to send a given amount of data. It also improves performance as it only takes one end-to-end cycle to send a bit of data, whereas the RTZ

scheme requires two end-to-end cycles.

A drawback of transition encoding is that it is relatively complex to encode and decode the signals on the wires. This overhead is justifiable only if the power savings are significant, as they are likely to be on inter-chip wiring.

2.2: Related communication schemes

A scheme that appears to combine the best of both RTZ and NRZ approaches is similar to the single-wire handshake proposed by van Berkel and Bink [3]. The D0 or D1 wire is driven high by the Sender to transmit data and then the same wire is returned low by the Receiver to indicate acknowledge. The drawback of this approach is the requirement to have the wire driven from either end, with the concomitant problems of drive hand-over, termination, and the infeasibility of inserting repeaters in long wires.

The data rate of any delay-insensitive scheme is limited by the end-to-end cycle time of the system. Where very high data rates are required timing information can still be embedded in the data, for example in the balanced 3-wire system proposed by Røine [4]. Such schemes achieve high performance by abandoning the constraint of delay-insensitive operation, and as such require more careful engineering than the approaches presented here.

Stan and Burleson have conducted a detailed analysis of various low-power encodings for global communications [5], particularly looking at broad bus-based communication. Again, there is no attempt to ensure delay-insensitive operation.

2.3: N-of-M codes

For higher communication rates, other delay insensitive codes can be used such as an ‘N-of-M’ code [2]. For example, in a 3-of-6 code six wires are used. A transition on any three of the six wires represents a symbol, and there are 20 possible symbols (6x5x4/3x2). These symbols can be used to represent a 4-bit binary value (0-15), Start, End, Ack, etc. A 2-of-7 code can similarly offer 21 symbols, giving the same data rate for less power (two instead of three transitions to send four bits of data) and more pins (7 instead of 6). Pin-efficiency can therefore be traded-off against power-efficiency.

(The ‘End’ symbol referred to above can be used to indicate the end of a data packet, and data packets can therefore have a variable length. This contrasts with the dual-rail system described previously where, because there are no spare symbols available during data transmission, the end of a data packet must be implicit. A fixed-length data packet is the simplest, though not the only, way to implement an implicit packet length.)

A 1-of-4 code has much simper encoding and decoding than a 3-of-6 or 2-of-7 code, but requires more pins. Like the dual-rail system, it has no spare symbols to indicate the end of a data packet. It is likely to be a good choice for on-chip communication where wiring is a less scarce resource. It is also likely that for many on-chip applications RTZ coding will give better performance than NRZ coding.

3: Proposed bidirectional communication scheme

In many applications bidirectional communication is required between chips. In what follows we begin by constructing a bidirectional scheme based upon dual-rail NRZ encoding; later we will return to NRZ N-of-M codes.

Bidirectional communication following the dual-rail system uses six wires. However, this number of wires appears excessive, and an optimization of this scheme uses only four wires by exploiting the data wires in one direction to carry acknowledges for communication in the other direction using the following approach:

- Four wires are used for bidirectional communication. A0 and A1 carry data symbols in one direction; B0 and B1 carry data symbols in the other direction.
- During true bidirectional communication a transition on A0 or A1 is acknowledged by a transition on B0 or B1, and vice versa.
- During unidirectional communication the same protocol applies, but the returned data is void.
- The start of valid data is indicated by preceding it with a ‘Start’ symbol. For example, a void response could be zero and a Start symbol a one. A predefined number of bits following a Start symbol represent valid data.

The above protocol gives good wiring efficiency and causes minimum synchronisation between the two directions. There is no requirement that bidirectional communications start at the same time; the second can start at any time before, during or after the first.

3.1: Protocol power-efficiency

For a low-power communications system we want to minimize the number of transitions on wires used to send a given data value. In particular, we want to avoid sending transitions when there is no data to send. This leads to a problem with the above proposed bidirectional communications system: in the DI communication model there is, in effect, a single token being passed from end to end determining whose turn it is to send a symbol. In order to enable either end to initiate communication this token must pass back and forth, requiring at least one wire transition in each direction and consuming power when there is no data being passed. To save power, the token should be discarded once neither end has data available to transmit. This leaves the system in

a zero-power quiescent state.

However, once the system is in a quiescent state either end may wish to initiate communication at any time, and inevitably from time to time both ends will attempt to initiate communication at the same time (to within some tolerance). Initiating communication requires the generation of a token, and if both sides attempt to initiate communication at the same time two tokens will be introduced into the loop. This is not allowed within the DI model as there is no way to prevent it from leading to symbol interference [2].

To resolve this problem the following mechanism is proposed:

- Both ends of the communication channel 'know' there is a problem, as both will issue a Start symbol and receive a Start symbol instead of an Ack symbol.
- One end of the channel must 'defer' to the other. We will call the end that defers the Slave, and the other the Master.
- The Slave defers by retracting its Start symbol and replacing it by an Ack.

In a DI system true retraction is not possible since, once the sender has made a transition on a wire, it cannot make another transition on the same wire until it has had confirmation (in the form of some sort of acknowledgement) that the first transition has been received at the far end. If two transitions were sent on a wire they might arrive as two distinct transitions, coalesce so that nothing is received, or partly coalesce into a runt pulse that may cause unpredictable behaviour in the receiver. The actual behaviour depends on the delays to which the two transitions are subject, and the delay-insensitive model starts from the assumption that these are unknown. Reliable retraction is therefore infeasible.

Instead of attempting retraction we define a special SlaveAck symbol which subsumes the Start symbol. In other words, every wire that makes a transition in the Start symbol also makes a transition in the SlaveAck symbol, and there must be at least one additional wire making a transition in the SlaveAck symbol to differentiate it from the Start symbol. When the Slave needs to 'retract' its Start symbol, it sends a transition on the wire that did not make a transition for the Start symbol, effectively morphing the Start symbol already sent into a SlaveAck symbol.

A corollary of this subsumption is that SlaveAck must be outside the code set used for normal communication.

- Once the Slave has deferred and the Master has received the SlaveAck symbol, the Master has the token and communication can begin. The Master can send a data symbol and the Slave can acknowledge this with a Start symbol, followed by full bidirectional data transmission.
- The final Ack symbol before communication ceases must use a set of wire transitions that does not overlap with the Start symbol, otherwise the end of the channel that issues the final Ack cannot subsequently issue a Start without risk of symbol interference.

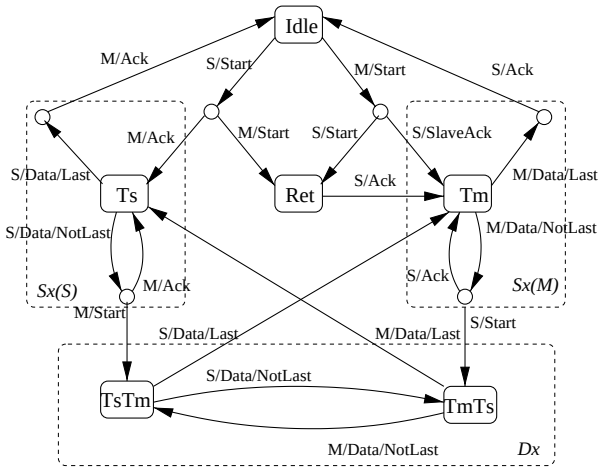

Figure 2: Protocol diagram

A sensible solution is to make Start and Ack non-overlapping codes and SlaveAck the union of the two codes (see the dual-rail example below).

The full protocol is illustrated in figure 2 (which was created by Alex Yakovlev and is used with permission). This illustrates the signal sequences seen on the wires connecting the Master (M) and the Slave (S). The six major states are Idle, Slave transmit (Ts), Master transmit (Tm), Retract (Ret) and two duplex states (TsTm and TmTs). The marking on the arcs illustrates the next transmission in the protocol, for example 'M/Data/Last' indicates that the Master sends its last data value.

3.2: Dual-rail example

In a dual-rail DI system 01 can be used for Start, 10 for Ack and 11 for SlaveAck. A state machine controls the transition from Start to data communication, but the return to idle must be controlled by implicit knowledge at both ends of the channel of the packet length.

3.3: Master and Slave identification

Identifying the Master and Slave in a symmetric system could be problematic. One idea, in a dual-rail system, would be for the zero wire to be held at logic 0 and the one wire at logic 1 at reset. If the wires from Master to Slave are cross-connected, the Slave can detect this at reset and cross them back internally, but remembering that it is the Slave (see figure 3). Alternatively, Master and Slave status could be configured at system start up by an external agency.

3.4: N-of-M example

A similar scheme can be constructed using N-of-M codes, with 2M wires used to form the bidirectional link,

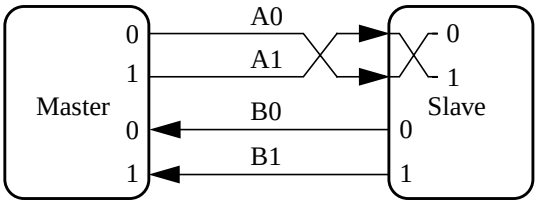

Figure 3: Identifying Master and Slave

M in each direction. As before, the SlaveAck symbol must be outside the N-of-M code set in order that it can subsume the N-of-M Start symbol; it could, for example, be the Start symbol plus one extra transition.

For minimum power in an N-of-M system it seems excessive to use N transitions for Ack. Instead, Start and Ack use a 1-of-M code. A state machine then moves the send channel to N-of-M for data communication and returns to 1-of-N after End. In this case SlaveAck can be the 2-of-M code formed by the union of Start and Ack.

In a 3-of-6 system this reduces the number of transitions for sending four bits of data in unidirectional communication from six to four, a 33% power saving.

4: Metrics

When comparing various alternative DI communication systems there are two metrics that seem most relevant:

- Power-efficiency: the number of bits of data sent per wire transition. This will be different for unidirectional and bidirectional communication.
- Pin-efficiency: the number of bits sent per send-receive cycle per pin.

(Clearly the above power-efficiency metric is based on the assumption that off-chip activity dominates the power consumption, so the power overhead of N-of-M encoding and decoding logic is negligible.)

Table 1 summarizes the performance of various systems using these metrics when sending very large data packets. For shorter packets the start-stop overheads will reduce the efficiency of all systems somewhat.

The codes listed in the table cover 2-rail (the 1-of-2 code) which sends one bit at a time in both RTZ and NRZ forms, the NRZ 1-of-4 code which sends 2 bits at a time, and the NRZ 3-of-6 and 2-of-7 codes which send 4 bits at a time. The codes listed as '...+ Ack' use separate acknowledge wires. The other codes use a symbol on the return set of wires for Ack. The 1/3-of-6 code uses 1-of-6 for Start and Ack and 3-of-6 for data; 1/2-of-7 likewise.

The 'cost' figure gives an overall figure of merit (smaller is better) based on a combination of the factors in the table as: 1/[(ave. bits/transition)*(ave. bits/cycle/pin)], where the averages are taken between unidirectional and bidirectional figures with equal weighting. This is an arbitrary cost function, and other combined

	Power-efficiency		Pin-efficiency		
	bits/transition		bits/cycle/pin		'cost'
DI style	uni	bi	uni	bi	
2-rail RTZ + Ack	1/4	2/8	1/12	2/12	32
2-rail RTZ	1/4	2/4	1/8	2/8	14.2
2-rail NRZ + Ack	1/2	2/4	1/6	2/6	8
2-rail NRZ	1/2	2/2	1/4	2/4	3.6
1-of-4 + Ack	2/2	4/4	2/10	4/10	3.3
1-of-4	2/2	4/2	2/8	4/8	1.8
3-of-6 + Ack	4/4	8/8	4/14	8/14	2.3
3-of-6	4/6	8/6	4/12	8/12	2
1/3-of-6	4/4	8/6	4/12	8/12	1.7
2-of-7 + Ack	4/3	8/6	4/16	8/16	2
2-of-7	4/4	8/4	4/14	8/14	1.6
1/2-of-7	4/3	8/4	4/14	8/14	1.4

Table 1: comparison of protocols

cost functions that give different weightings to power- and pin-efficiency and uni- and bidirectional communication can be postulated to suit a particular application.

It can be seen that those communication systems that do not use a separate acknowledge wire have at least 25% better power-efficiency and 12.5% better pin-efficiency in bidirectional communication than those that do. For the dual-rail codes the benefits are 50% in both power-efficiency and pin-efficiency. The 1/N-of-M codes only affect power-efficiency in the unidirectional case, where the benefit is to cancel the power-efficiency loss of the new N-of-M code system relative to the equivalent with a separate acknowledge wire.

In a system that makes extensive use of bidirectional communication the best codes are eight times more power-efficient than a basic RTZ dual-rail with acknowledge system, and four time more power-efficient than the NRZ dual -rail system.

On the basis of the cost function used in Table 1, the NRZ 2-of-7 with separate acknowledge code is 16 times better than the basic RTZ dual-rail scheme; the use of the reverse channel offers a further 20% improvement, and the 1/2-of-7 scheme a further 10% on top of that, making a factor 23 improvement overall.

The duplex communication scheme does, of course, introduce a coupling between the send and receive chan-

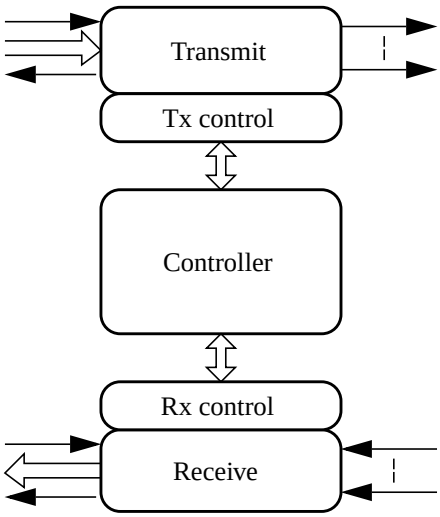

Figure 4: Communication system structure

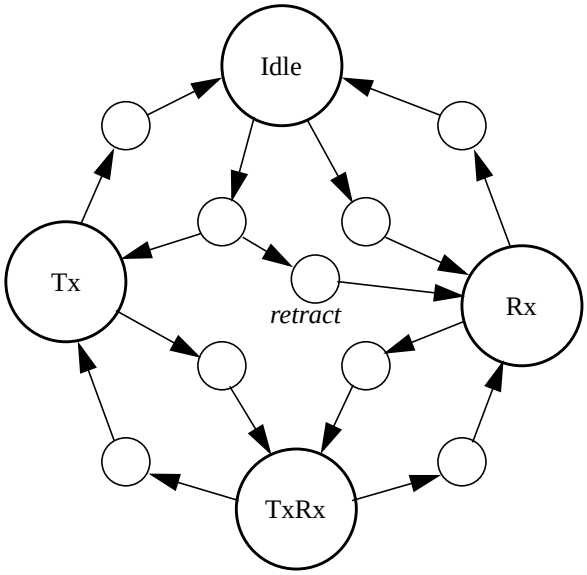

Figure 5: Controller state machine

nels that is avoided when separate acknowledge wires are used. If one channel has a slow data source this will adversely affect the performance of the reverse channel. The extent to which this amounts to a serious problem depends on the application, but a higher-level protocol may be necessary in some cases to minimise interference effects. For example, data transmission could be delayed until it is known that a full packet is available and the receive buffer is empty.

5: Controller design

The design of an inter-chip communication system based on the above techniques presents some interesting challenges. The principle is fairly straightforward. The design is first broken down into three subsystems as shown in figure 4:

- the transmit logic - this handles the outgoing data stream;
- the receive logic - this handles the incoming data stream;
- the controller - this handles the interactions between the two data streams required to implement the protocol.

The transmit and receive subsystems are then further broken down into:

- a control section that is largely independent of the data encoding scheme;
- a datapath section that performs the data encoding or decoding.

The high-level specification of the controller reflects the four possible modes of operation: idle, transmission (Tx), reception (Rx) and bidirectional communication (TxRx). The state machine also requires states to handle transitions between these top-level states, some of which include arbitration processes. For example, consider the case when transmit data becomes available just as the controller is about to acknowledge received data. Should it send an acknowledge or a Start symbol? The relative timing of the transmission request and the data reception is arbitrary, so arbitration is required to ensure reliable operation.

In addition, the Slave controller requires state transitions to handle the Start retract case discussed in section 3.1. The full state machine structure is shown in figure 5.

6: Controller implementation

A detailed implementation of the controller was carried out using Petrify [6], an STG-based synthesis tool that produces speed-independent circuits. The design of the STG for this circuit turned out to be a complex task, as is evidenced by the result shown in figure 6.

The controller implementation required 83 gates, and the full communication interface using 3-of-6 encoding required a total of just under 400 gates. The breakdown of these gates is shown in figure 7. Just over half of the gates are in the transmit and receive data blocks, and these figures would be most subject to change if a different DI code were used.

The interface was designed using the 0.35 µm AMS CMOS gate library available through Europractice and simulated using Verilog with typical gate speeds. The simulation used two interfaces communicating through modelled interconnect with 5 ns delay in each direction, under which conditions it had a cycle time of 24 ns. The inter-chip 3-of-6 encoding sends 4-bits in each direction per cycle for a total throughput of over 320 Mbits/s.

7: Conclusions

Delay-insensitive communication is attractive for inter-chip point-to-point communication as it will give

reliable operation over short or long connections. The wires carry unidirectional signals, so repeaters can easily be inserted in long connections and the DI operation will not be affected. N-of-M codes using transition encoding offer significant (factor 8 for a 2-of-7 code) power savings over the simplest dual-rail return-to-zero codes due to the greatly reduced number of transitions on off-chip wires that are required to send a given quantity of data.

Bidirectional communication can be implemented using a pair of unidirectional channels, one in each direction. However, an improvement to this approach is to replace the acknowledge wires with acknowledge symbols sent down the return data channel. This reduces the pin requirement and further improves the power-efficiency. Where the channel sometimes carries unidirectional data, it is more power-efficient to use a 1-of-M code for the acknowledge and to switch dynamically between this and N-of-M for data transmission.

Achieving zero quiescent power is possible without breaking the delay-insensitivity of the inter-chip connections, but a special protocol is required to handle the case where both ends of the communication start to send data at (approximately) the same time.

The full protocol controller is complex, but has been

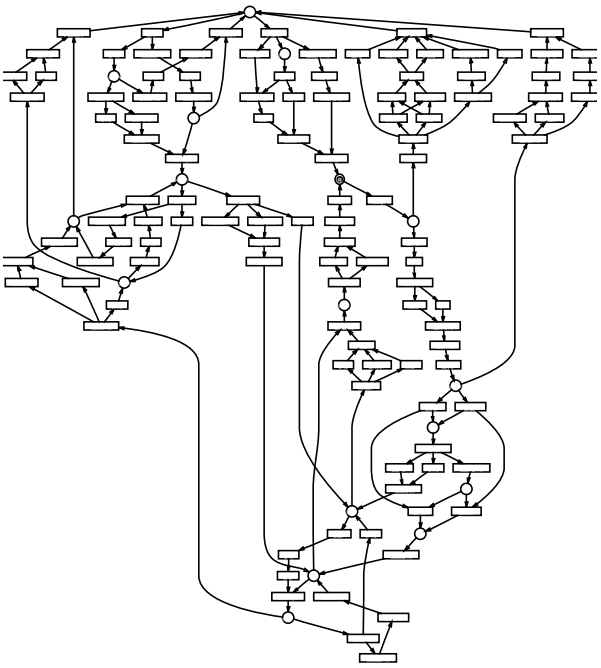

Figure 6: Controller STG

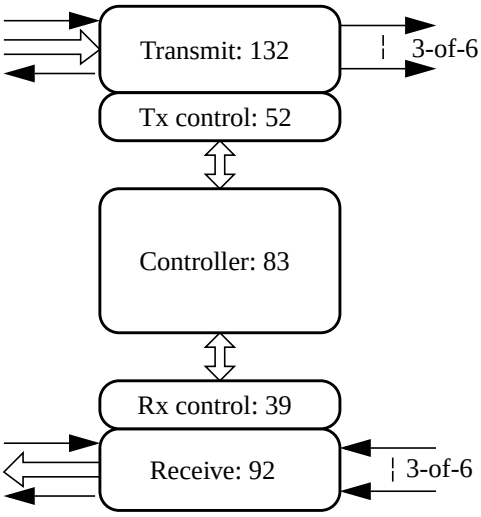

Figure 7: Communication system gate counts

synthesized as a speed independent circuit using Petrify [6] and simulated at the gate level. The circuit operates correctly and achieves acceptable performance, though further improvements may be possible.

Greater power-efficiency is also achievable at the electrical level [2], but this is in addition to the power benefits of reduced switching activity that are the outcome of the techniques presented here.

8: References

[1] IEEE P1355(1995) Standard for Heterogeneous Interconnect (HIC).

[2] Dally, W.J. and Poulton, J.W., “Digital Systems Engineering”, Cambridge University Press, 1998. ISBN 0 521 59292 5.

[3] van Berkel, K and Bink, A, “Single-Track Handshake Signalling with Application to Micropipelines and Handshake Circuits”, Proc. Async’96, March 1996, pp. 122-133.

[4] Røine, P.T., “A System for Asynchronous High-Speed Chip to Chip Communication”, Proc. Async’96, March 1996, pp. 2-10.

[5] Stan, M.R. and Burleson, W.P., “Low-Power Encodings for Global Communication in CMOS VLSI”, IEEE Trans. VLSI, Dec 1997, pp. 444-455.

[6] Cortadella, J., Kishinevsky, M., Kondratyev, A., Lavagno, L., Yakovlev, A. “Petrify: a tool for manipulating concurrent specifications and synthesis of asynchronous controllers” IEICE Transactions on Information and Systems, Vol. E80-D, No. 3, March 1997, pp. 315-325.