システムＬＳＩ設計技術　101－10
（2001. 5. 18）

非同期スイッチの試作

石川 健一郎† 川上 大輔† 柴田 裕一郎‡ 天野 英晴†

†慶應義塾大学 理工学部
〒223 横浜市港北区日吉 3-14-1
(045)-560-1063
E-mail: {ishikawa,kawakami,hunga}@am.ics.keio.ac.jp

‡長崎大学 工学部
〒852-8521 長崎県長崎市文教町 1-14
(095)-847-1111
E-mail: shibata@cis.nagasaki-u.ac.jp

あらまし

完全非同期式の4入力4出力スイッチ用チップを 0.6μm CMOS フルカスタム方式により試作し、回路シミュレーションによってその性能を測定した。独自のアービトレーション回路を導入した結果、試作したチップはスイッチとして正しく動作し、現実的な前提の元では最大約 352MBits/s、チップの限界では最大約 1.6GBits/s の性能を実現した。

キーワード　非同期回路, スイッチ, アービタ, 束データ方式

Trial manufacture of asynchronous switch

K. Ishikawa† D. Kawakami† Y. Shibata‡ H. Amano†

†Dept. of Information and Computer Science, Keio University
3-14-1, Hiyoshi, Kohoku-ku, Yokohama 223, Japan
(045)-560-1063
E-mail: {ishikawa,kawakami,hunga}@am.ics.keio.ac.jp

‡Dept. of technology, Nagasaki University
1-14, Bunkyo, Nagasaki, 852-8521, Japan
(095)-847-1111
E-mail: shibata@cis.nagasaki-u.ac.jp

Abstract

Fully asynchronous 4 inputs/outputs switch is designed on a 0.6μm full-custom CMOS chip, and the performance is evaluated with a circuit level simulator. Using novel arbitration circuits, the chip achieves 1.6Gbits/sec peak throughput and 352Mbits/sec average throughput per one port.

key words　asynchronous circuit, switch, arbiter, bundle data method

1 はじめに

近年のプロセス技術の大幅な進歩により素子の微細化が可能になり、同期式プロセッサの動作周波数は飛躍的に向上してきた。今後もこの傾向は続いていき、2005年には動作周波数10GHzのプロセッサを開発すると発表する企業も現われている。

しかし、プロセス技術の微細化と共に配線が細くなっていくのに加え、チップの面積が広くなっているため、結果として配線遅延が増大していき、ある程度以上微細化されると遅延はゲート遅延ではなく配線遅延の方が支配的になる事が分かっている。

配線遅延の増大によってまず影響を受けるのがクロック線である。クロック線は必要な全ての素子に同時に分配されるのが期待されるが、配線遅延が増大すると共に同時にクロックを分配させることは難しくなっていく。つまり、クロックを使う同期式チップには動作速度に限界がある[1]。

限界が見はじめた同期式チップにかわる新しい方式が盛んに研究されているが、その方式の一つにクロックを一切使わない非同期式のチップが注目を集めている。例えば、非同期式プロセッサは、マンチェスタ大学のAMULET3i[2]、カリフォルニア工科大学のminiMIPS[3]、東京大学のTITAC-2[4]など様々な大学で実際に製作されており、性能や消費電力の面で同期式プロセッサと同程度の性能を上げている。

非同期式チップはプロセッサだけではなく周辺のチップにも広がりはじめている。例えば、マンチェスタ大学のA-MULET3iにはDMAコントローラが内蔵されている[2]。

本発表では非同期式の周辺チップとして束データ方式によりデータを扱う4入力4出力の非同期式スイッチの試作を行い、シミュレーションによる性能評価を行った。

2 スイッチの構造

2.1 転送方式について

同期式においてはデータの表現は簡単にできる。例えば、0をLow、1をHighとすると1ビットを表すのには1つの信号線があれば十分で、クロック線から合図がきた瞬間にそのとき表現したいデータ(0ならLow、1ならHigh)をとっていればよい。

しかし、非同期ではクロック線が存在しないため、仮にデータ線がLowのままであっても、それが1つのLowを表しているのかそれとも時間的に連続した何個かのLowを表しているのか分からない。

そこで非同期式回路でデータを表現するために様々な方法が考えられている[5]。

2.1.1 束データ方式

束データ方式ではデータを表すための線の他にready線を用意しておき、現在データ線に流しているデータが有効な場合はready線をHighに、データが無効な場合はready線をLowにすることによってデータを表現している(図1)。この方法はready線がある他は同期式と全くかわらないため回路の流用が簡単にでき設計しやすい。また、Nビット表現するのにN+1ビットしか必要としないためデータ転送に必要な線が少なくて済む。

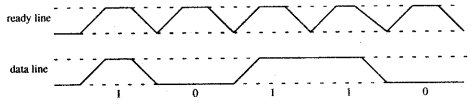

図1: bundle data method

2.1.2 2線2相式

2線2相式では1つのデータを表すのに2本の線を使う。2本のうち1本が1、1本が0を表現しているとして、1を表現している方がHighならば1、0を表現している方がHighならば0、両方とも0ならばデータが流れていないを意味する(両方とも1の場合はエラーを表現する)。各データとデータの間には休止期と呼ばれる両方の線が0の状態が必要である。DIモデル、QDIモデルやSDIモデルで厳密に取り扱うことができるためプロセッサ内部でのデータの表現によく使われる。また、Nビット表現するのには2Nビット必要である。

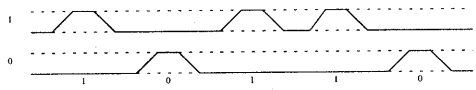

図2: two line two phase method

2.1.3 2線1相式

2線1相式では2線2相式と同じように1つのデータを表すのに2本の線を使う。2本のうち1本が1、1本が0を表現しているとして、1を表している線が遷移を起こしたら1、0を表している線が遷移を起こしたら0、何も遷移が起きていないのならデータは流れていないを意味する。こうすることにより2線2相式では必ず必要だった休止期が必要なくなる。だが、回路が2線2相式を扱う回路より複雑になるという欠点があり、この欠点のため、休止期がないのにも関わらず2線2相式よりもスピードが遅くなってしまう。また、奇数回の転送を行った際には特別な処理をしなければならない。

今回製作したスイッチでは束データ方式を用いてデータを表現している。これは束データ方式なら送りたいデータ幅+1本のデータ線があれば送れるのに対し、2線2相式

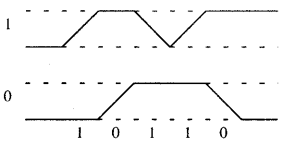

図 3: two line one phase method

や 2 線 1 相式では送りたい幅の 2 倍の本数のデータ線が必要となるからである。

2.2 スイッチの機能

このスイッチは 4 入力 4 出力でビット幅は束データ方式使用時には 4 ビット (つまり実際の出力は 5 ビット) である。クロック線無しで動作し、何段もつなげることができる。ビット幅が狭いのは出力ピンの数の問題であり、同じ設計で更に広いビット幅を持たせることは容易に可能である。

このスイッチは以下のようなポートを持っている。

ポート名	ポートの説明
SEL1	出力先を指定するビットの上位ビット
SEL2	出力先を指定するビットの下位ビット
occupied(in)	出力ポートを確保するための線 (入力)
ready(in)	入力データの ready 線
in1～4	入力データ
occupied(out)	出力ポートを確保するための線 (出力)
ready(out)	出力データの ready 線
out1～4	出力データ
ack(in)	ack の入力線
conflict(in)	衝突が起こったことを示す線 (入力)
ack(out)	ack の出力線
conflict(out)	衝突が起こったことを示す線 (出力)

具体的な使い方としては、まず、SEL1、SEL2 へ信号を送って出力先を指定すると同時に occupied(in) に信号を送り信号の伝達路を確保する。conflict(out) から衝突が起こったという信号が届かないで、ack(out) から信号が届いたら、データを送る (データを送り終わるまでは occupied(in) に信号を送り続ける)。conflict(out) から衝突が起きたという信号が来た場合はしばらく待った後、もう一度 occupied(in) に信号を送る。バッファなどは使用していないためデータ転送路上の入出力ポートはデータを送り終えるまで占領され続ける。

2.3 スイッチの内部

このスイッチは大きく分けて

- same output place detection part
- collision detection part
- output determination part
- output part

の 4 つの部分からなっている。スイッチ 1 ポートの構造を図 4に示す。(1-4) は全てのポートからの信号であることを、(1) は対応するポートからの信号であることを意味する。

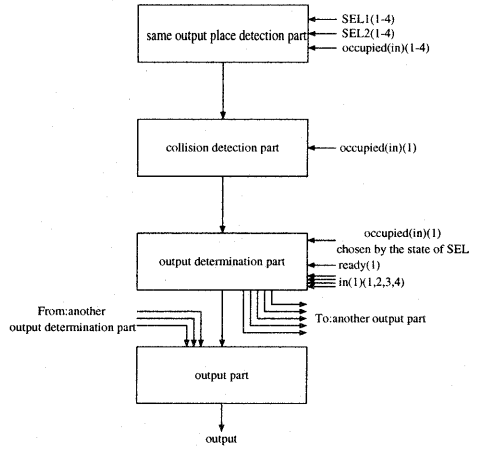

図 4: switch structure(one port)

まず、same output place detection part で同じところを使う他の入力が無いか調べ、collision detection part で、その結果を使って実際にデータを流すかどうか決め、output determination part で、各出力に流すデータを決め、output part で実際に出力を行う。

2.4 アービタ

非同期式のアービタとしては図 5に示す回路がすでに提案されている [6]。

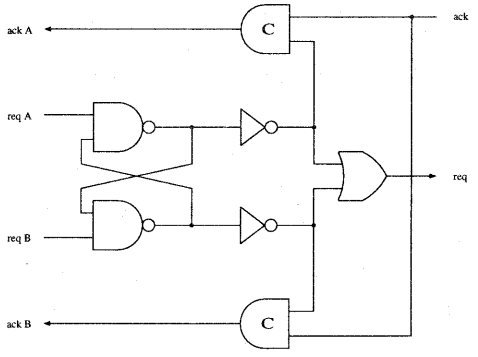

図 5: arbiter

この回路で、どちらかの req が 0 から 1 になればもう一方の req が 1 になっても反応しないという、アービタの基本的な動作を実現できる。しかし、同時に信号が入ると NAND 素子の出力が不安定になりうまく動かないという難点がある。また、C エレメントが 2 入力の場合でも 2 つ必

要でありトランジスタの数が多くなってしまう。そこで、このチップでは以下に示す方法でアービトレーションを行っている。

2.5 same output place detection part

図6に示される same output place detection part では他の入力と出力先が重なっていないかどうか確認している。この回路では XNOR 素子を使って、SEL 線により指定されるそれぞれの出力先が同じで、かつ、データを送ろうとしている (occupied 線が High になっている) 場合には collision detection part に High を、そうでない場合は Low を出力する。出力先が重なっているかどうかのテストなので本来なら occupied 線の比較は AND 素子によって行うべきであるが、XNOR 素子を使っている SEL 線の比較とタイミングがずれてしまうのでここでは XNOR 素子によって行っている。このため、occupied 線が両方とも Low の場合でも SEL 線さえあっていれば collision detection part に High が出力されてしまうが、occupied 線が High にならない限り collision detection part の出力は High にならないため問題はない。

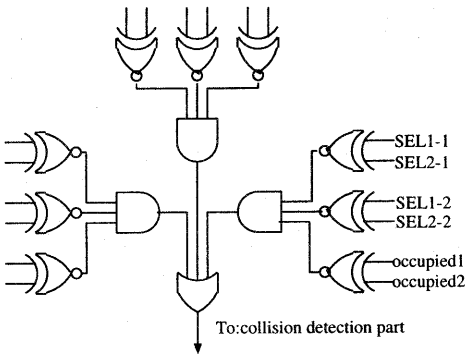

図 6: same output place detection part

2.6 collision detection part

図7に示される collision detection part では same output place detection part の結果と現在の状況をふまえて、現在入力されているデータを出力先に出力すべきなのかどうかを判断している。具体的には same output place sdetection part からの信号が Low の場合、および、same output place sdetection part からの信号が High かつすでに output determination part に High の信号を出している場合、次の output determination part に High の信号を出力している。このような設計により、他のポートが同じ出力先に信号を出していない限りデータを出力ポートに出すことが出来、データを出力している間に他のポートが同じ出力先に出力しようとしても影響を受けない。delay は same output place detection part からの信号と occupied(in) から来た信号が同時に AND 素子に入るように調整するためにある。

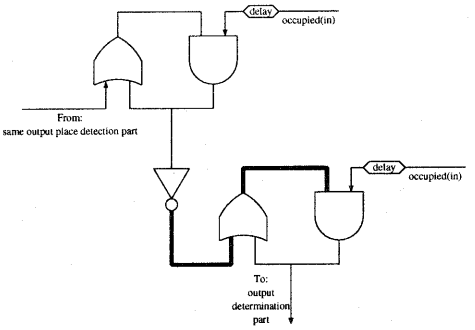

図 7: collision detection part

図 7、図 8、図 9、図 10で細い線で表されている配線は Low の信号が流れている配線で、太い線で表されている配線は High の信号が流れている配線である。図/reffig:cdp ではインバータ素子から 2 つ目の OR 素子までと、2 つ目の OR 素子から 2 つ目の AND 素子までが High の状態である。この場合 occupied 線が Low なので output determination part への出力も Low である。

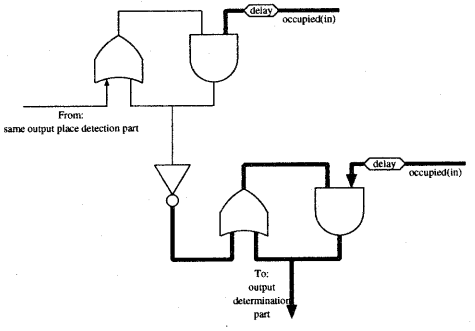

図 8: collision detection part(normal)

図 8は出力先がどことも重ならない状態で出力要求信号が来た場合の回路の状態を示す。図中に示すように output determination part には High の信号が出力される。

図 9と図 10は対になっており、2 つ以上のポートが同じときに同じ出力ポートに書き込もうとすると、一番先に書き込もうとしたポートが図 9の状態となり、それ以外のポートが図 10の状態となる。

図 9で表されているのがデータを送っている途中に他のポートが同じ出力ポートに出力しようとした場合の回路の状態である。図中に示す通り、output determination part への High の出力は維持されたままであり、データを出力し続けることができる。

図 10で表されているのが他のポートがデータを送っている途中に同じ出力ポートに出力しようとした場合の回路の

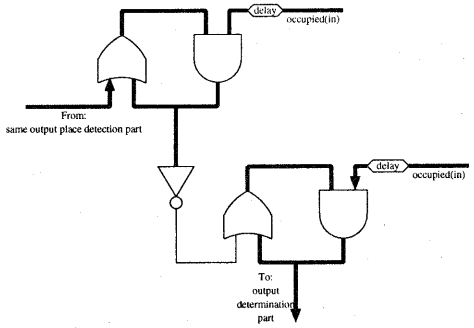

図 9: collision detection part(be interrupted)

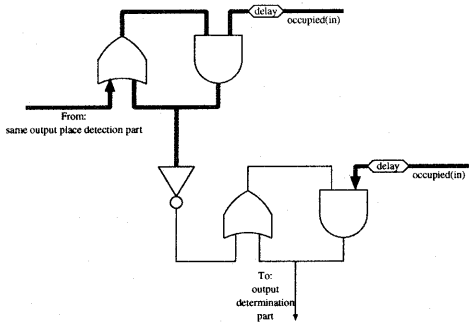

図 10: collision detection part(interrupt)

状態である。図中に示す通り、output determination part へは Low しか出力されない。つまり、データを出力することは出来ない。

以上は素子や配線による遅延が無いと考えた場合であり、実際にはほぼ同時に 2 つ以上のポートが同じポートに出力しようとすると全てのポートが出力に失敗する (Low を output determination part に出力する)。

2.7 output determination part

output determination part を図 11に示す。実際にはこれと同一回路を 4 つを用いる。それぞれが各出力ポートに対応しており、occupied 線が High で SEL1、SEL2 が条件と一致したときだけその出力ポートに信号を送る。

一番右の 2 入力 NAND 素子には collision detection part からの信号と occupied(in) chosen by the state of SEL が入力され、両方とも High(つまり、出力先があいており、その出力先に出力したがっている) なら Low(active) を output part にある occupied(out) へ出力する。

残りの 3 入力 NAND 素子は collision detection part からの信号と occupied(in) chosen by the state of SEL に加えて、ready や in1-4 からの信号も NAND 素子への入力に加わる。

collision detection part からの信号は 24 個の NAND 素子に分配されるため、信号の立上りに時間がかかってしまい、レイテンシが増大する。このため、collision detection part の出力する部分だけを 4 つ用意し、1 つで 6 つの NAND 素子を受け持つようにしてレイテンシを低くしている。

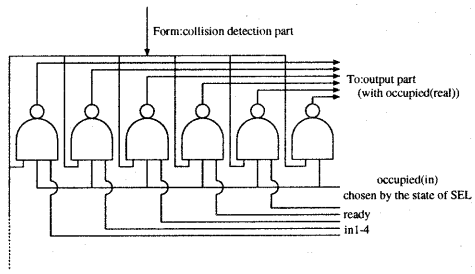

図 11: output determination part

2.8 output part

output part ではそれぞれの方向から来た信号を単純に NAND 素子を使って 1 つにしている (この部分だけ active Low である点に注意)。アービトレーションが取れていることは保証されているため、このようにしても問題はない。

2.9 その他の構造

2.9.1 conflict(out)

conflict(out) への出力は図 12の回路の出力と conflict(in) の入力を処理したものとを OR 素子であわせたものである。conflict(in) は図 13のように、それぞれのポートからこのポートへ出力する時に現在使用中であることを示す occupied(real) を反転させたものと AND を取ったものを用いる。occupied(real) が Low(active) であるということはその方向から信号が来ていることを意味するので、この方法により信号が来ている方向を知り、逆に辿ることができる。

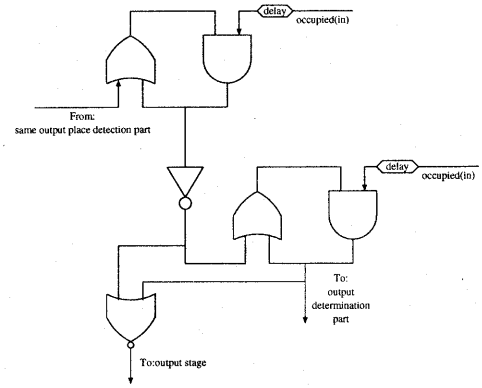

図 12: conflict(in)

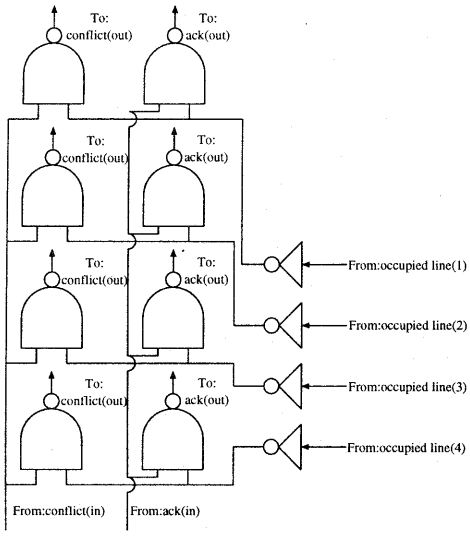

図 13: conflict line

2.9.2 ack(out)

ack(out) への出力は conflict(out) とあまり変わらない。違いはチップ内で ack(out) に出力するような信号が発生することがないだけである。図 13のように ack(out) が決定され、それが出力される。

3 シミュレーションによる評価

実チップはまだデータインしていないためシミュレーションによって評価する。

3.1 シミュレーションの方法

シミュレーションの手順を以下に示す。

- layoutPlus を使ってパッド部分以外の回路のレイアウトを決定
- PDRACULA を使って回路の結線情報を抽出
- hspice を使ってシミュレーション
- AvanWaves を使って表示

各ツールは VDEC[1]によって提供されたものを用いた。また、シミュレーションの条件は以下の通りである。

- ROHM 社の 0.6 μ m プロセスを使用
- 素子はフルカスタム方式
- 信号の立ち上がり、立ち下がりには 0.2ns かかる
- 入力パッドの delay は立ち上がり、立ち下がりともに 0.135ns
- 出力パッドの delay は立ち上がりが 1.748ns、立ち下がりは 1.572ns

3.2 シミュレーションの結果

チップの特性として以下のようなことが分かった。

- チップのゲート数は NAND 換算で 1533 個
- occupied 線が Hi の時、ready 線、data 線は最低 1.3ns の長さが無いと認識されない
- ready 線、data 線の Hi の部分と次の Hi の部分との間には (0V の部分が) 最低 0.8ns ないと融合してしまう
- 0.5ns 以内に別の入力ポートが同じ出力ポートに出力しようとすると両方とも conflict あつかいになり、出力ができない

またチップの性能としては以下のようなことが分かった。

- data 線、ready 線の delay はパッドの delay を考えなければ約 3.4ns、パッドの delay を考えれば約 5.3ns になる
- occupied 線の delay はパッドの delay を考えなければ約 3.4ns、パッドの delay を考えれば約 5.3ns になる
- ack 線の delay はパッドの delay を考えなければ約 1.1ns、パッドの delay を考えれば約 3.0ns になる
- conflict が起きるとそれぞれ 3.4ns、3.6ns、3.85ns、2.65ns 後に conflict を起こした各出力ポートの conflict 線に Hi が出力される。
- data 線の信号は 1 回チップを通る毎に信号の長さが-0.020ns～+0.252ns 変化する (パッドの delay を考えない場合)
- data 線の信号は 1 回チップを通る毎に信号の長さが-0.122ns～+0.150ns 変化する (パッドの delay を考える場合)

転送元と転送先が 1 つのこのチップによって結合されており、以下のような手順で信号をやりとりすると仮定する。

1. 送り元が送り先に向かって occupied 線で信号を送り、伝送路を確保する
2. occupied 線の信号を受け取ると、送り先はデータを受け取る用意をした後、ack 線で信号を送る
3. 送り元が送り先にデータを送る

[1]東京大学大規模集積システム設計教育研究センター

stable な時間	処理時間	burst 転送	転送性能
0.6ns	18.2ns	2.5ns	1.54GBits/s
10ns	57.4ns	11.35ns	312MBits/s

表 1: シミュレーションの結果

4. 送り先が送り元に ack 線で信号を送る

結果は表 1のようになった。

現実的な数字として信号の認識、処理には 10ns かかると仮定してシミュレーションすると受け取り先が信号を受け取りきるまでに約 57.4ns かかる。

また、バースト転送が可能だとする場合、1 つのデータを送るごとに 11.35ns 余計にかかる。

このまま、送り先を変えないで送り続けるとすると一秒間に約 88.1MBits の情報を 1 つの線で送ることができる。このスイッチの一ポート当たりのビット幅は 4 ビットなので、1 つのポート当たりのデータ転送能力は最大約 352MBits/s になる。

しかし、ここでシミュレーションした、バースト転送において 1 回にかかっている時間のうち、約 8 割は信号そのものを表現するために使われており、チップの性能を表しているとは言いがたい。

そこで、信号がチップを通るのを 1 回に限定して信号が短くなってしまうのを無視するとし、受け取る側はどんなに短い信号でも認識できると仮定することにより、チップそのものの性能を導くとすると、受け取り先が信号を受け取るまでに 18.2ns かかるとわかる。この場合、全ての信号がステーブルなのは 0.6ns である。

また、この仮定の元で、バースト転送が可能だとすると 1 つのデータを送るごとに 2.5ns 余計にかかる。

このまま、送り先を変えないで送り続けるとすると一秒間に約 400MBits の情報を 1 つの線で送ることができる。このスイッチの一ポート当たりのビット幅は 4 ビットなので、1 つのポート当たりのデータ転送能力は最大約 1.60GBits/s になる。

4 おわりに

研究の結果、束データ方式において信号の長さの誤差が ±0.1ns 程度の単機能スイッチを作ることができた。そして、十分に余裕を持った環境で最大約 312MBits/s の転送能力を持つことをシミュレーションによって確かめることができた。また、スイッチとして最低限機能する環境においては最大約 1.54GBits/s の転送能力を持つことをシミュレーションによって確かめることができた。

現在、データイン可能なデータが完成しており、データインの受付が始まり次第データインする予定である。

参考文献

[1] 南谷 崇: “非同期技術の潮流と実践のアプローチ-最新鋭「TITAC-2」の挑戦” エレクトロニクス,1998 年 8 月号,pp.65–69.

[2] J.D. Garside, W.J. Bainbridge, A. Bardsley, D.M. Clark, D.A. Edwards, S.B. Furber, J. Liu, D.W. Lloyd, S. Mohammadi J.S. Pepper, O. Petlin, S. Temple, J.V. Woods: “AMULET3i - an Asynchronous System-on-Chip” Proceedings Async 2000,April 2000,pp.162–175.

[3] 小沢 基一, 南谷 崇: “MIPS R3000 マイクロプロセッサの挑戦” エレクトロニクス,1998 年 8 月号,pp.72–74.

[4] A.Takamura, M.Kuwako, M.Imai, T.Fujii, M.Ozawa, I.Fukasaku, Y.Ueno, T.Nanya: “TITAC-2: An asynchronous 32-bit microprocessor based on Scalable-Delay-Insensitive model” International Conference on Computer Design(ICCD),October 1997 ,pp.288–294.

[5] 亀田 義男, 南谷 崇: “パルス理論による非同期データパス回路の構成” 電気情報通信学会論文誌,D-I Vol.J83-D-I No.1,pp.163–170.

[6] 小栗 清: “布線論理による新しい汎用情報処理アーキテクチャPCA(完)” Bit,July 2000 Vol.32 No.7,pp.51–59.