九州大学学術情報リポジトリ
Kyushu University Institutional Repository

アクティブビットを考慮した低電力データパス設計手法

室山，真徳
九州大学　大学院システム情報科学府情報工学専攻

兵頭，章彦
九州大学　大学院システム情報科学府情報工学専攻

大隈，孝憲
シンプレックスソリューションズ株式会社

安浦，寛人
九州大学　大学院システム情報科学府情報工学専攻

https://hdl.handle.net/2324/3714

出版情報：電子情報通信学会技術研究報告. VLD2002, 2002-05. 電子情報通信学会VLD研究会
バージョン：
権利関係：

アクティブビットを考慮した低電力データパス設計手法

室山　真徳[†]　兵頭　章彦[†]　大隈　孝憲[††]　安浦　寛人[†]

† 九州大学 大学院システム情報科学府 情報工学専攻
〒 816–8580 福岡県春日市春日公園 6–1
†† シンプレックス ソリューションズ株式会社
〒 210-0005 神奈川県川崎市川崎区東田町８パレール三井ビルディング８ F

E-mail: †{muroyama,akihiko,yasuura}@c.csce.kyushu-u.ac.jp, ††okuma@simplex.com

あらまし　変数のインスタンスであるデータの特性を考慮してデータパス部分の電力を削減する手法を提案する．データに着目した場合，本質的に必要となる部分は転送されるデータの全ビットが必要となることは少ない．筆者はデータに対し本質的に必要となる部分をアクティブビットと定義し，不必要な部分に対し最適化を行うことでデータパス部分全体の電力を削減する手法を提案する．特に本論文においてはデータパス部分のうちバス部分に着目する．バスはシステムによっては非常に多くの電力を消費する部分である．提案手法はアクティブビットを考慮することでバスの信号遷移を抑えるものである．同時に提案手法の実装方法も紹介する．実験により本手法が有効であることが確認できた．

キーワード　CMOS，アクティブビット，低電力化，スイッチング，データパス

Low Power Datapath Design Technique Considering Active Bits

Masanori MUROYAMA[†], Akihiko HYODO[†], Takanori OKUMA[††], and Hiroto YASUURA[†]

† Department of Computer Science and Communication Engineering Graduate School of Information Science and Electrical Engineering, Kyushu University
Kasuga Koen 6–1, Kasuga-shi, Fukuoka, 816–8580 Japan
†† Simplex Solutions, K.K.
Parale Mitsui Bldg. 8F, 8, Higashida-cho, Kawasaki-ku
Kawasaki-shi Kanagawa 210–0005 Japan

E-mail: †{muroyama,akihiko,yasuura}@c.csce.kyushu-u.ac.jp, ††okuma@simplex.com

Abstract　We popose a low power datapath technique considering characteristics of transfer data. In many cases, not all bits of data have effective information. We call *active bits* which are need for representation of data, unnecessarry parts (except active bits from all part of data) are optimized for reduction of power consumption. Especially, we focus on bus in datapath bacause of large power consumption. In some LSIs main factor of power consumption is transitions of buses. By considering active bits signal transitions of buses are reduced. In addition, an implementation is introduced. Experimental results demonstrate high effects of our technique.

Key words　CMOS, active bit, low-power, switching, datapath

1. はじめに

近年のチップの大規模化，複雑化と携帯機器などの普及に伴い，低電力化がバッテリーの持続時間を長くするため重要な問題となっている．プロセッサのようなシステムはデータパス，制御回路などから構成されており，データパスはバス，メモリ，算術演算器，レジスタなどから構成されている．データパス部分の消費電力はシステム全体に対して大きな影響を与える．システムにおいてバスの幅，あるいは処理単位の幅であるデータパス幅は低消費電力化にとっては重要な設計パラメータである．変数に対して必要となるデータパス幅を解析することで電力を削減することが可能となる．文献 [5] は低電力化のための最適なデータパス幅を決定する方法を紹介している．ただし，これらのデータパス幅を決定する手法は静的な手法である．

我々は低電力データパスのための新たな手法を提案する．データパス上を流れるストリームデータにおいて各々のデータの必要な部分はデータパス全体に流れているとは限らない．ここでデータの本質的に必要なビットをアクティブビットと呼び，不必要な部分をインアクティブビットと呼ぶことにする．我々が提案する手法はデータパス上を流れるデータに対し動的にアクティブビットを解析し利用することで電力を削減するものである．

本論文では CMOS 論理回路を対象として議論する．CMOS LSI における消費電力は静的な電力と動的な電力とに分けて考えることができる．本論文においては静的な電力は考慮しないこととし，動的な電力のみを考える．CMOS 回路における動的な電力は以下の式によって近似できる [8]．

$$P = \sum_{k=1}^{N} CL_k \cdot Swit_k \cdot V_{DD}^2 \qquad (1)$$

ここで Nは回路の総ゲート数，CL_kはゲート kの負荷容量，$Swit_k$ はゲート kでのスイッチング回数，そして V_{DD}は電源電圧である．消費電力はゲートの総スイッチング回数に比例することを式 (1) は示しており，総スイッチング数を削減するとこで消費電力を削減できる．バス部分においては負荷容量が大きく信号の多重化などによって信号遷移が多くなると消費電力が大きくなる（オンチップにおいて約 40%の電力を消費する場合もある [7]）．

上記の 3 つのパラメータを削減することで電力を小さくできるが，我々はスイッチング回数を削減することで消費電力を削減する手法を提案する．これまでに信号の符号化によるスイッチング回数削減手法が提案されている．信号がランダムに遷移する場合の手法として bus-invert 手法 [9]，on-line adaptive scheme [4] がある．特に bus-invert 手法は広く知られている手法であり提案手法の比較対象とした．この手法はバスのスイッチング回数が少なくなるようにデータの値を反転させる方法である．以降この手法を BI 法と呼ぶ．この手法では付加ビットとして 1 ビット用意し（INV ビット）付加ビットが 1 の場合には転送データを反転していることを表す．一方，信号に相関関係がある場合のスイッチング回数削減手法として Gray code [2]，T0 手法 [1] そして working-zone coding [3] がある．しかしこれらの手法は各々のデータの特徴を考慮しておらず信号に相関関係の現れやすいアドレスバスのみを考慮している．我々の提案する手法はデータの特徴を考慮しておりデータバスを対象としている．

本論文の構成は以下のとおりである．2 章ではアクティブビットを考慮した低電力バスを提案し，実験により提案手法の効果を確認する．3 章ではアプリケーションを固定した場合にオーバーヘッドを削減する手法を提案する．4 章で実装方法を紹介し，5 章で本論文をまとめる．

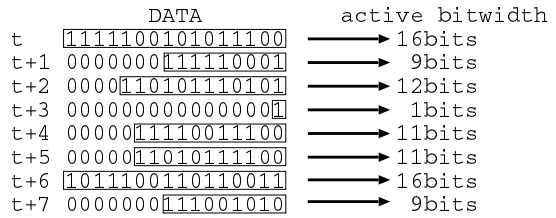

図 1　アクティブビットの例

2. アクティブビットを考慮した電力削減手法

2.1 基本アイデア

32 ビットプロセッサでは通常は 32 ビット長の命令が使用される．実際のアプリケーションにおいてはアクティブビットはしばしば 32 ビットよりも少なくてすむ．実行時にバス上には動的にデータが転送されるのでアクティブビット幅もまた動的に変化する．ここでアクティブビット幅を以下のように定義する．

a) **上位ビットから連続する 0 を消去して残ったビットの数**

この場合にはアクティブビットは符号なしの固定小数点表記の数のみを扱っていることになる．他には符号付き固定小数点表記の数の場合や浮動少数点表記の数などがあるが本論文では取り扱わない．符号付きの場合は次のように定義することができる．

b) 上位ビットから連続する 1 を消去して残ったビットの数

提案手法においてはアクティブビット幅の定義として a) を採用する．またデータが 0 の場合をアクティブビット幅 0 とする．

バスにはアドレスバス，データバスなどがあるが，これ以降データバスに限定して説明をする．理由はアドレスバスはアクティブビットがほとんど一定だからである．したがって本論文では我々は特にアクティブビット幅がよく変化するデータバスを対象とする．データバスを通して様々なデータが転送されるが，データにおいて実際に必要となる部分はデータごとに様々である．つまりデータのアクティブビットはデータごとに異なる．図 1を例として説明する．時刻 t から $t+7$ まで 8 つの 16 ビットのデータを転送する場合を考える．図中の四角で囲まれたビットラインがアクティブビットであり時刻 t におけるデータのアクティブビット幅は 16 となる．このデータ群の場合アクティブビット幅の最大値は 16 となりデータパス幅は 16 となる．図 2は mpeg2player と 116k byte のデータにおいてアクティブビット幅の頻度を示す．図よりデータパス幅 32 ビット全てを使用するデータは少なく特に 8 ビット幅以下のデータが多いことが分かる，つまりアクティブビット幅は様々であり偏ることが分かる．

アクティブビットには計算に必要な情報が含まれているがインアクティブビット部分はどのような論理値（信号値）であっても問題ない．そこで**インアクティブビット部分は前のデータの該当ビットラインの論理値を保存する**ようにする．つまりインアクティブビットのスイッチングが起らないようにする．スイッチングにより電力は消費されるので電力の消費が抑えられることになる．

図 1の例にある転送データ群にこの手法を適用した結果が図 3である．図 1のデータを転送する際のスイッチング回数の合計は 51 回であるのに対し，提案手法を用いた場合である図 3のデータのスイッチング回数の合計は 29 回となる．ただしアクティブビット幅の情報を転送するための付加ビットが必要となる．アクティブビット幅を表現するのにワンホッ

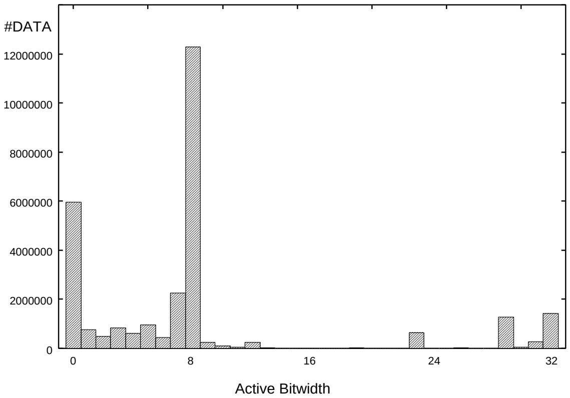

図 2　アクティブビット幅ごとの転送データ数（mpeg2payer:116k byte data）

```
         DATA
t    1111100101011100
t+1  1111100111110001
t+2  1111110101110101
t+3  1111110101110101
t+4  1111111110011100
t+5  1111111010111100
t+6  1011100110110011
t+7  1011100111001010
```

図 3　符号化の例

ト符号化と 2 進符号化の 2 つの方法を考えることができる．ワンホット符号化はある数を表現するのに 1 ビット必要となる符号化であり n 個の数を表現するのに n ビットが必要であり，ワンホット符号化は他の符号化（2 進符号化を含む）より多くのビットを必要とするがスイッチングは小さくすることができる．低電力化のためにワンホット符号化を使用すると，必要となる付加ビットの本数は n ビットのバス幅においてビット幅は 0 ビット幅から n ビット幅までの（$n+1$）通りであるので合計 $n+1$ 本必要となる．

2.2　付加ビットを転送データ外部に持つ場合

オンチップバスの場合には付加ビットが増えてもそれほど問題にならない．しかしチップ間のデータ転送に使用されるバスでは付加ビット用に I/O ピンを使用するために付加ビットを削減する方法を考える必要がある．そこで付加ビット削減手法を提案する．まず以下に変数を定義する．

- N_{seg} ： セグメント数（$1 \leqq N_{seg} \leqq n$）
- bw_i ： 各セグメントのビット幅（$1 \leqq bw_i$）

つまりバスを N_{seg}に分割する．また以下の条件を満足する必要がある．

$$\sum_{i=1}^{N_{seg}} bw_i = n \tag{2}$$

式 2 は各セグメントのビット幅の合計はバス幅に等しいことを表す．本章では特に各セグメントのビット幅はすべて等しい場合のみを考える．このときセグメントのビット幅は以下の式から導くことができる．図 3 の例は $N_{seg} = 16, bw = 1$ の場合である．

$$\forall i, bw_i = n/N_{seg}\,(= bw)$$

これ以降ではアクティブビットは以下のように考える．ここで 0 セグメントとはセグメント内の全ビットが 0 の場合のことである．

```
         DATA                     active bitwidth
t    1111100101011100  ──────▶  8bits
t+1  0000000111110001  ──────▶  5bits
t+2  0000110101110101  ──────▶  6bits
t+3  0000000000000001  ──────▶  1bits
t+4  0000011110011100  ──────▶  6bits
t+5  0000011010111100  ──────▶  6bits
t+6  1011100110110011  ──────▶  8bits
t+7  0000000111001010  ──────▶  5bits
```

図 4　分割例（$N_{seg} = 8, bw = 2$）

```
         DATA
t    1111100101011100
t+1  1111100011110001
t+2  1111010101110101
t+3  1111110101110100
t+4  1111101110011100
t+5  1111111010111100
t+6  0011100110110011
t+7  1011100011001010
```

図 5　Embeded 手法の例

c) **上位のセグメントから連続する 0 セグメントを消去して残ったセグメントの数**

この場合の最大アクティブビット幅は N_{seg} となる．したがって付加ビットは $N_{seg}+1$ 本となる．

図 4は図 1のデータに対し $N_{seg} = 8, bw = 2$ と分割した場合の例である．最大アクティブビット幅は 8 となる．つまり付加ビットは 9 本必要である．しかしスイッチング回数は 31 回となり，$N_{seg} = 16, bw = 1$ の場合よりスイッチング回数は若干多くなる．

2.3　付加ビットを転送データ内部に持つ場合

ここではアクティブビット幅情報を転送データ内部に持つ場合を考える．つまり付加ビットを転送データ内部に埋め込む．この方法は 2 つの段階から成り立つ．第 1 の段階ではインアクティブビット部分には前のデータを保持するようにし，第 2 段階ではアクティブビット部分の最上位ビットは前の値の反転値をとるようにする．こうすると前のデータと今のデータを上位ビットから比較した場合にはじめて値が異なるビットがアクティブビットの最上位ビットとなる．以下この手法を embeded と呼ぶことにする．図 5 は embeded 手法の例を示している．

2.4　実　　験

2.4.1　実 験 方 法

本手法を用いて実験を行った．実験用ツールとしてプロセッサシミュレータ SimpleScalar を用いた．SimpleScalar のデータバスの幅は $n = 32$ である．アプリケーションとして 4 種類（mpeg2play,go,gzip,perl）を用意した．また mpeg2play においてはデータの依存関係を調べるためにデータを 3 種類用意した（データサイズは 116k byte,245k byte, 649k byte である）．比較対象にはよく知られる bus-invert coding(BI) を選択した．

まず SimpleScalar 用にベンチマークプログラムをコンパイルし SimpleScalar 上でベンチマークプログラムを走らせる．その際のメモリの読み込み，書き込みの際のデータバスの値をトレースファイルとして保存する．トレースファイルのデータに本手法を適用してスイッチング回数とアクティブビット幅の情報を得た．

2.4.2　実 験 結 果

符号化なしの場合，提案手法を使用した場合（$N_{seg} \in$ {32,16,8,4,2,1},embeded）および BI 法を使用した場合のス

表 1　各アプリケーションのスイッチング回数

App.	総データ数	スイッチング回数
mpeg2play		
116k	29008616	116570540
245k	32448512	196764223
649k	195225786	683169457
go	195225786	960143319
gzip	195225786	1324939651
perl	683908	4712544

イッチング回数を表 1,2,3, 4にそれぞれ示す．結果には付加ビット部分におけるスイッチング回数も含めている．mpeg2play (649k)，go，gzip はトレースファイルが非常に大きかったためファイルが扱うことができる量を越えていた．そこでファイルが扱うことができる最大のパターン数分のみに対して提案手法を適用し実験を行った．

データの依存性を調べるために mpeg2play においては入力データを 3 種類用いて実験を行った（これは入力データのファイルサイズを変えることによる影響を調べることが目的である）がそれぞれ約 50%前後のスイッチング回数が削減できた．したがって入力データのサイズにあまり関係なく大幅なスイッチング回数が削減できると思われる．

mpeg2play では 3 種類の入力データのサイズがを与えたときすべての場合においてアクティブビット幅 8 ビット以下のデータが約 80%であった．同じアプリケーションにおいては入力データに依存せずアクティブビット幅の出現頻度は同じような傾向を示すことが分かった．go では 6 種類のアクティブビット幅が大半を占めており各アクティブビット幅の出現頻度に大きな偏りが存在する．特にアクティブビット幅が 0 であるデータの出現が約 50%を占める．gzip は他のアプリケーションと比較し各アクティブビット幅の出現頻度に大きな偏りはない．perl はアクティブビット幅が 29 ビットであるデータが約 35%存在する．アクティブビット幅が大きいデータが多いことを示している．

各アプリケーションにおけるバスラインごとの総スイッチング回数について説明する．実験より全てのアプリケーションにおいて上位のバスラインのスイッチング回数の削減が大きいことが分かった．mpeg2play では入力データに関係なくバスライン 8 ビットより上位のバスラインのスイッチング回数が大幅に削減されていた．go はもとの場合においてバスライン 14 ビット目以下のスイッチングが大部分であり，本手法を用いることで大幅なスイッチング回数を削減できていることが分かった．削減効果が高いのはアクティブビット幅 0 および 8 の出現頻度が高いことが原因である．gzip でもスイッチング回数は削減できているが削減効果は mpeg2play, go に比べそれほど高くなかった．これはアクティブビット幅がそれほど変化しないデータが連続してバス上に流れているためだと推測される．最後に perl であるが，これも gzip と同様に削減効果があまり高くなかった．これはアクティブビット幅が大きいデータが多いために本手法の特性を生かした削減ができないためである．

我々の手法はアクティブビット幅がよく変化するアプリケーションに対して効果が高い．また各バスラインごとのスイッチング回数をみた場合，上位ビットの方がスイッチング回数が少なくなる．このことを利用してスイッチング回数の多い下位ビットを上位ビットに挿入することでカップリング容量を小さくすることができると思われる [6].

3. アプリケーションに特化した手法

本章ではアプリケーションを特定した場合に付加ビット数の制限のもとで消費電力が最小となる手法を説明する．まずアクティブビット幅 i が出現する確率を $P_{act}(i)$ とする．

表 3　提案手法適用後のスイッチング回数の削減結果（アクティブビット幅情報を転送データに埋め込む場合）

App.	スイッチング回数	削減率（%）
mpeg2play		
116k	71493568	38.7
245k	99143546	49.6
649k	449070546	34.3
go	439428153	54.2
gzip	1156630291	12.7
perl	3545468	24.8

表 4　既存の低電力バス化手法（BI 法）を適用した場合のスイッチング回数の削減結果

App.	スイッチング回数	削減率（%）
mpeg2play		
116k	68537403	41.2
245k	102407712	48.0
649k	423384781	38.0
go	959710528	0.0451
gzip	1216382035	8.19
perl	4216642	10.5

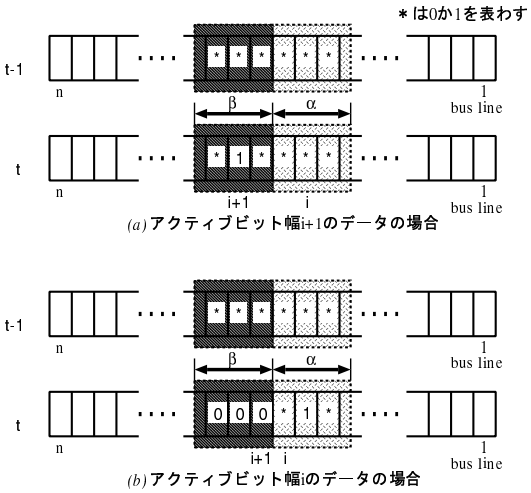

図 6　付加ビット削減手法 ($bw_i = \alpha, bw_{i+1} = \beta$)

この情報は事前にシミュレーションによって得ることが可能である．初期状態は $N_{seg} = n, bw = 1$ とし，これから $N_{seg} = C, 1 \leqq C \leqq n-1$ にすることを考える．つまり付加ビットを $n+1$ 本から $C+1$ 本に削減する．セグメント $i, i+1$ についてセグメント幅がそれぞれα, βとし，図 6の (a) のようにアクティブビット幅が $i+1$ の場合と図 (b) のように i の場合を考える．すると (a) の場合はセグメントを 1 つにすることによるスイッチング回数の増加はない．そこで (b) の場合のみを考える．時刻 $t-1$ においてセグメント $i+1$ 内の j ビット目（$1 \leqq j \leqq \beta$）の 1 の出る確率を $P_1(j)$ とし全パターン数を PT とすると，全体のスイッチング回数の増加分の合計Δは

$$\Delta = P_{act}(i) \cdot (\sum_{j=1}^{\beta} P_1(j) - 2P_{act}(i+1)) \cdot PT$$

となる．ここで例えば $\forall j, P_1(j) = 1/2, (1 \leqq j \leqq \beta)$ とすると

$$\Delta = P_{act}(i) \cdot (\beta/2 - 2P_{act}(i+1)) \cdot PT$$

となる．このことを利用してアクティブビット幅情報のためのビット削減の手続きを考える．手続きの概略は以下のとおりである．

表 2　提案手法適用後のスイッチング回数の削減結果（アクティブビット幅情報を転送データ外部に持つ場合）

App.	スイッチング回数						削減率
	$N_{seg}=32$	$N_{seg}=16$	$N_{seg}=8$	$N_{seg}=4$	$N_{seg}=2$	$N_{seg}=1$	(%)
mpeg2play							
116k	62565155	65013280	67504813	71705992	85307285	109015604	46.3
245k	91869628	95909363	100214872	107067319	136289899	18656175 3	53.3
649k	381402768	400521703	414056202	441363330	506485789	636190 005	44.2
go	488982827	537244344	537286039	488654693	667593634	76531383 5	49.1
gzip	1120586026	1123376607	1169074717	1195639475	1271015529	1 309485465	15.4
perl	3566507	3617192	3719750	3820318	4318665	4490912	24.3

表 5　スイッチング回数削減率の比較

	N_{seg}				
	32	16	8	4	2
分割サイズ一定	53.3	51.3	49.1	45.6	30.7
アプリケーションを考慮	53.3	50.6	50.4	49.2	42.9

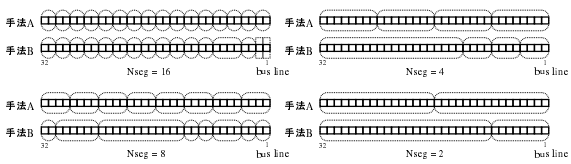

図 7　分 割 結 果

（1）　全ての隣り合うセグメントのペアに対してΔの値を計算する．

（2）　Δの値が最小となるペアを 1 つのセグメントにし，複合化後のアクティブビット幅の出現確率を計算．

（3）　総セグメント数が Cとなるまで 1. と 2. を繰り返す．

提案手法を用いて実験を行った．アプリケーションとして mpeg2pay，データにサイズ 245k byte の動画を用いた．実験によるスイッチング回数削減の結果を表 5に示し分割結果を図 7に示す．表中にはアプリケーションを考慮した付加ビット削減手法を用いた場合と分割サイズ一定（$bw = 32/N_{seg}$）の場合のスイッチング回数の削減率を表示している．ここで削減率の単位は%である．図中の手法 A は分割サイズ一定の場合であり，手法 B はアプリケーションを考慮した場合である．表よりアプリケーションを考慮した分割手法を用いた場合は分割サイズ数を小さくしてもスイッチング回数の削減効果が高いままであることが分かる．しかし $N_{seg} = 16$ の場合は分割サイズ一定の場合の方が削減効果は高い．これはΔの値を計算をする際に $P_1(j)$ の値を 1/2 と近似したが実際の $P_1(j)$ の値は異なっており，その結果Δの真の値からの誤差が大きかったためであると思われる．分割数を小さくすればするほどアプリケーションを考慮した場合と分割サイズ一定の場合との削減率の差は大きくなっている．図 7において破線で囲まれているバスラインの集まりがセグメントである．図より $N_{seg} \neq 16$ 以外は手法 A と手法 B の分割方法は大きく異なることが分かる．また分割方法によってスイッチング回数が変わることが分かった．

本論文では同時に 2 つのペアを複合化しているが，今後 3 つ以上のペアを複合化することを考えることが課題となる．また $P_1(j)$ $(1 \leqq j \leqq \beta)$ の値を正確に得た場合，どの程度のスイッチング回数削減効果が得られるのか調べることも課題である．削減効果が高い場合には $P_1(j)$ の値を考慮すべきである．また，mpeg2play のみでなく他のアプリケーションを試す必要がある．

4.　回路実装方法

次に実際に回路として実装した場合について考える．実装する回路は符号化部と復号化部に分けることができる．それぞれについて実装方法を以下に述べる．

4.1　符 号 化 部

以下の議論では変数は論理変数（値は 0 または 1 のみとりうる）とし関数は論理関数とする．

もとのデータを $A_t = (a_t^0, a_t^1, ..., a_t^{n-1})$ とし符号化後のデータを $Z_t = (z_t^0, z_t^0, ..., z_t^{n-1})$ とする．またアクティブビット幅データを $W_t = (w_t^0, w_t^1, ..., w_t^{N_{seg}})$ とする．ここで t は時間をあらわし a_t^i, z_t^i, w_t^iはそれぞれもとのデータ，符号化後のデータ，アクティブビット幅データの時刻 t における論理値を表す．符号化回路はアクティブビット幅検出回路と転送データ生成回路から構成されている（図 8参照）．

まずアクティブビット幅検出部について考える．セグメント i における j ビット目を s_{ij}とし

$$X_i = \bigcap_{j=1}^{bw} s_{ij}$$

とする．すると W_tは以下の式で表現できる．

$$w_t^{N_{seg}} = a_t^{n-1}$$

$$w_t^k = \overline{\bigcup_{l=k+1}^{N_{seg}} X_l \cdot X_k (0 \leqq k \leqq (N_{seg}-1), X_0 = 1)}$$

次に転送データ生成回路について考える．

$$Y_i = \bigcup_{j=i+1}^{N_{seg}} w_t^j \ (\text{ただし } N_{seg}/2 < i)$$

$$Y_i = \bigcup_{j=0}^{i} w_t^j \ (\text{ただし } i \leqq N_{seg}/2)$$

とする．A_tのセグメント i におけるデータを S_t^iとすると Z_t のセグメント i におけるデータ T_t^iは以下のようになる．

$$T_t^i = if\ Y_i = 1\ then\ S_t^i\ else\ S_{t-1}^i \ (\text{ただし } N_{seg}/2 < i)$$

$$T_t^i = if\ Y_i = 0\ then\ S_t^i\ else\ S_{t-1}^i \ (\text{ただし } i \leqq N_{seg}/2)$$

ここで T_tを 2 つの部分に分けた理由は実装した際に回路の段数を削減するためである．またアクティブビット幅データの $w_t^{N_{seg}/2+1}$ビット目は使用しない．したがってアクティブビット幅データのための付加ビットの $N_{seg}+1$ 本のうち実際に必要なビットは N_{seg}本となる．(ただし embeded の場合には $a_t^\alpha = 1 (1 \leqq \alpha \leqq n)$ とすると $T_t^\alpha = \neg T_{t-1}^\alpha$となる．$\neg$は論理否定を表わす.)

4.2　復 号 化 部

復号化回路について考える．復号化回路の入力は符号化

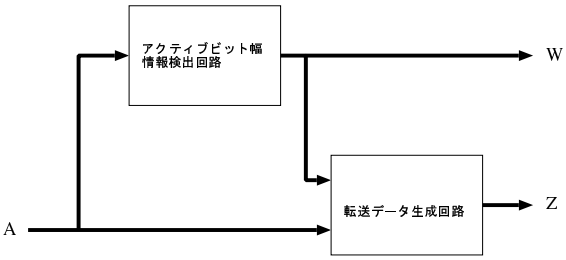

図 8　符号化回路

したデータ Z_t とアクティブビット幅情報 W_t である．出力を $B_t = (b_t^0, b_t^1, ..., b_t^{n-1})$ とする．復号化回路は転送データ生成回路とほぼ同じになる．

符号化回路と同様に

$$Y_i = \bigcup_{j=i+1}^{N_{seg}} w_t^j \text{ (ただし } N_{seg}/2 < i)$$

$$Y_i = \bigcup_{j=0}^{i} w_t^j \text{ (ただし } i \leqq N_{seg}/2)$$

とし，Z_t のセグメント i におけるデータを S_t^i とする．B_t のセグメント i におけるデータ $\overline{T}_t^i$ は以下のようになる．

$$\overline{T}_t^i = if\ Y_i = 1\ then\ S_t^i\ else\ 0 \text{ (ただし } N_{seg}/2 < i)$$

$$\overline{T}_t^i = if\ Y_i = 0\ then\ S_t^i\ else\ 0 \text{ (ただし } i \leqq N_{seg}/2)$$

embeded の場合はまずデータからアクティブビット幅情報の抽出を行う必要がある．まず Z_t のセグメント i におけるデータを S_t^i とし，各セグメントごとの前の値と今の値の排他的論理和をとり，それを $diff_i$ $(1 \leqq i \leqq N_{seg})$ とする．

$$diff_i = S_{t-1}^i \oplus S_t^i$$

これをもとにしてアクティブビット幅情報を抽出する．抽出方法は符号化のときの抽出方法と同じである（$diff_i$ を X_i に置き換える）．

これらの式を満足するように回路を設計する．復号化回路はアクティブビット幅情報検出回路が必要ないので符号化回路に比べ構成が単純になる．

4.3 評　　価

実際に符号化，復号化回路を verilog-HDL で記述し，Synopsys 社の論理合成ツールである DesignCompiler で面積最小の制約を与えて合成した．プロセステクノロジは日立 $0.18\mu m$ を使用した．クロック周波数は 100MHz とした．結果は表 6および表 7に示す．面積の単位としている BC は 1BC=$53.76\mu m^2$ である．

符号化回路は全ての場合，復号化回路は embeded 以外の場合において提案手法の回路の方が消費電力が小さい．これは BI 法が消費電力が大きい排他的論理和 XOR を多く使用しているためである．また embeded は復号化回路が大きくなる．遅延では符号化回路は全ての場合において提案手法の方が小さい値が出た．一方，復号化回路では $N_{seg} = 1, 2, 4$ 以外では BI 法の方が小さい値が出ている．BI 法の復号化回路は簡単な構成となっているためである．既存の手法よりもよい結果が得られた．

5. おわりに

本論文では動的にデータのアクティブビットを考慮したバスの電力削減手法を提案した．バスのインアクティブ部分は前のデータを保持し不要なスイッチングを起こさせないことで電力を削減した．オーバーヘッド削減手法も併せて紹介した．実験によりスイッチング回数は最大 53.3%削減でき，本手法が有効であることが確認できた．アプリケーションが固定の場合には付加ビット数の制約のもとでスイッチング数が最小となるセグメント幅を求めることができる．

同時に提案手法の回路実装の方法も紹介した．設計者は回路コストや遅延，電力，付加ビットの本数を考慮して最適なバスの構成方法を選択できる．

また，アクティブビットを考慮してデータパス全体の低電力化が可能であると考えられる．その場合より効果的に電力を削減できると思われる．

表 6　符号化回路の比較

	N_{seg}							
	32	16	8	4	2	1	emb	BI
付加ビット（本）	32	16	8	4	2	1	0	1
Cell 数	216	200	159	147	141	141	231	282
面積（BC）	305	291	237	221	216	215	346	525
消費電力（μW）	655	569	511	501	499	500	636	1671
遅延（$n\ sec$）	3.19	3.44	2.17	1.71	1.33	0.89	3.72	3.98

表 7　復号化回路の比較

	N_{seg}							
	32	16	8	4	2	1	emb	BI
付加ビット（本）	32	16	8	4	2	1	0	1
Cell 数	54	46	41	35	33	32	231	32
面積（BC）	71	53	41	35	33	32	410	64
消費電力（μW）	192	158	152	149	143	140	736	267
遅延（$n\ sec$）	2.39	1.34	0.67	0.41	0.40	0.24	3.89	0.48

謝辞

本チップの設計は東京大学大規模集積システム設計教育センターを通し，株式会社日立製作所の協力および Synopsys ツールを用いて行われたものである．

文　　献

[1] L. Benini, G. De Micheli, E. Macii, D. Sciuto and C. Silvano, “Asymptotic zero-transition activity encoding for address buses in low-power microprocessor-based systems,” Proc. *of the Great Lakes Symp. VLSI*, pp.77–82, Mar. 1997

[2] H. Mehta, R. M. Owens and M. J. Irwin, “Some issues in Glay code addressing,” Proc. *of the Great Lakes Symp. VLSI*, pp.178–180, Mar. 1996

[3] E. Musoll, T. Lang and J. Cortadella, “Workng-zone encoding for reducing the energy in microprocessor address buses,” IEEE Trans. *on VLSI Sytems*, vol 6, pp.658–672, Dec. 1998

[4] L. Benini, A. Macii, E. Macii, M. Poncino and R. Scarsi, “Synthesis of low-overhead interfaces for power-efficient communica tion over wide buses,” Proc. *of Design Automation Conf.* , pp.128–133, Mar. 1999

[5] H. Yamashita, H. Tomiyama, A. Inoue, F. N. Eko, T. Okuma and H. Y asuura, “Variable Size Analysis for Datapath Width Optimization,” Proc. *of Fifth Asian Pacific Conference on Hardware Description Language*, pp.69–74, Mar. 1998

[6] K. W. Kim, K. H. Baek, N. Shanbhag, C. L. Liu and S. M. Kang, “Coupling-Driven Signal Encoding Scheme for Low-Power Interface Des ign,” Proc. *of International Conference on CAD*, pp.318–321, Nov. 2000

[7] Dake Liu, Christer Svensson, “Power Consumption Estimation in CMOS VLSI Chips”, IEEE Journal *of SOLID-STATE CIRCUITS*, vol. 29, No. 6, pp.663–670, Jun. 1994

[8] 榎本 忠儀: CMOS 集積回路, 培風館. 1996

[9] Mircea R. Stan, Wayne P. Burleson, “Bus-Invert Coding for Low-Power I/O”, IEEE Trans. *on VLSI SYSTEMS*, vol. 3, pp.49–59, Mar. 1995