Conversions from DPD/GBCD to BCD without Table Lookup

A Case Study on the Intel XScale Architecture

Ping Tak Peter Tang

Computational Software Lab

Intel Corp.

April 21, 2005

DPD to BCD – Algorithm

Goal: Convert 10-bit DPD to 12-bit BCD encoding.

DPD to BCD – Algorithm

Goal: Convert 10-bit DPD to 12-bit BCD encoding.

DPD				BCD		
`pqr`	`stu`	`vwxy`	$\rightarrow$	`abcd`	`efgh`	`ijkm`
				d_2	d_1	d_0

DPD to BCD – Algorithm

Goal: Convert 10-bit DPD to 12-bit BCD encoding.

DPD				BCD		
`pqr`	`stu`	`vwxy`	$\rightarrow$	`abcd`	`efgh`	`ijkm`
				d_2	d_1	d_0

DPD encoding is based on cases of d_j small or large:

DPD to BCD – Algorithm

Goal: Convert 10-bit DPD to 12-bit BCD encoding.

DPD				BCD		
`pqr`	`stu`	`vwxy`	$\rightarrow$	`abcd`	`efgh`	`ijkm`
				d_2	d_1	d_0

DPD encoding is based on cases of d_j small or large:

Cases	p	q	r	s	t	u	v	w	x	y
all $d_j \leq 7$	b	c	d	f	g	h	0	j	k	m
only $d_0 > 7$	b	c	d	f	g	h	1	0	0	m

These cases are `[vwx]` $\leq$ `[100]`.

DPD to BCD – Algorithm

Goal: Convert 10-bit DPD to 12-bit BCD encoding.

DPD				BCD		
`pqr`	`stu`	`vwxy`	$\rightarrow$	`abcd`	`efgh`	`ijkm`
				d_2	d_1	d_0

DPD encoding is based on cases of d_j small or large:

Cases	p	q	r	s	t	u	v	w	x	y
only $d_1 > 7$	b	c	d	j	k	h	1	0	1	m
only $d_2 > 7$	j	k	d	f	g	h	1	1	0	m

These cases are `[vwx]` $<$ `[111]`.

DPD to BCD – Algorithm

Goal: Convert 10-bit DPD to 12-bit BCD encoding.

DPD				BCD		
`pqr`	`stu`	`vwxy`	$\rightarrow$	`abcd`	`efgh`	`ijkm`
				d_2	d_1	d_0

DPD encoding is based on cases of d_j small or large:

Cases	p	q	r	s	t	u	v	w	x	y
only $d_0 \leq 7$	j	k	d	0	0	h	1	1	1	m
only $d_1 \leq 7$	f	g	d	0	1	h	1	1	1	m
only $d_2 \leq 7$	b	c	d	0	0	h	1	1	1	m

These cases are indicated by `[st]` $<$ `[11]`.

DPD to BCD – Algorithm

Goal: Convert 10-bit DPD to 12-bit BCD encoding.

DPD				BCD		
`pqr`	`stu`	`vwxy`	$\rightarrow$	`abcd`	`efgh`	`ijkm`
				d_2	d_1	d_0

DPD encoding is based on cases of d_j small or large:

Cases	`p`	`q`	`r`	`s`	`t`	`u`	`v`	`w`	`x`	`y`
all $d_j > 7$	`0`	`0`	`d`	`1`	`1`	`h`	`1`	`1`	`1`	`m`

DPD to BCD – Implementation

Goal: Convert 10-bit DPD to 12-bit BCD encoding.

DPD				BCD		
`pqr`	`stu`	`vwxy`	$\rightarrow$	`abcd`	`efgh`	`ijkm`
				d_2	d_1	d_0

DPD to BCD – Implementation

Cases	p	q	r	s	t	u	v	w	x	y
all $d_j \leq 7$	b	c	d	f	g	h	0	j	k	m
only $d_0 > 7$	b	c	d	f	g	h	1	0	0	m

These cases are [vwx] $\leq$ [100].

DPD to BCD – Implementation

Cases	p	q	r	s	t	u	v	w	x	y
all $d_j \leq 7$	b	c	d	f	g	h	0	j	k	m
only $d_0 > 7$	b	c	d	f	g	h	1	0	0	m

These cases are [vwx] $\leq$ [100].

```
!On input R0 is [p q r s t u v w x y]
and     R1,R0,#0x0E       !R1 is [v w x 0]
cmp     R1,#0x0A          !compare [v w x 0] with [1 0 1 0]
bge     Cont_1
and     R1,R0,#0x380      !R1 is [pqr0000000]
bic     R0,R0,#0x380      !R0 is [000stuvwxy]
orr     R0,R0,R1,LSL #1   !R0 is [0pqr0stuvwxy]
RETURN
```

DPD to BCD – Implementation

Cases	p	q	r	s	t	u	v	w	x	y
only $d_1 > 7$	b	c	d	j	k	h	1	0	1	m
only $d_2 > 7$	j	k	d	f	g	h	1	1	0	m

These cases are $[\mathtt{vwx}] < [\mathtt{111}]$.

DPD to BCD – Implementation

Cases	p	q	r	s	t	u	v	w	x	y
only $d_1 > 7$	b	c	d	j	k	h	1	0	1	m
only $d_2 > 7$	j	k	d	f	g	h	1	1	0	m

These cases are [vwx] $<$ [111].

```
Cont_1:
cmp     R1,#0x0E          !compare [v w x 0] with [1 1 1 0]
bge     Cont_2
!Now, either d1 or d2 alone is large
cmp     R1,#0x0C          !compare with [1 1 0 0]
beq     d2_large
```

DPD to BCD – Implementation

Cases	p	q	r	s	t	u	v	w	x	y
only $d_1 > 7$	b	c	d	j	k	h	1	0	1	m
only $d_2 > 7$	j	k	d	f	g	h	1	1	0	m

These cases are [vwx] $<$ [111].

```
d1_large:
and     R1,R0,#0x60       !extract [s t] into R1
bic     R0,R0,#0x6E       !clear s t v w x
orr     R0,R0,R1,LSR #4   ![p q r 0 0 u 0 s t y]
mov     R1,R0,LSR #7
add     R0,R0,R1,LSL #7   ![p q r 0 0 0 u 0 s t y]
orr     R0,R0,#0x80       ![p q r 1 0 0 u 0 s t y]
```

DPD to BCD – Performance

Case	Latency (cycles)	Comments
$d_2, d_1 \leq 7$	6.3	no branch misprediction
only $d_1 > 7$	16.3	
only $d_2 > 7$	17.8	
only $d_0 \leq 7$	20.8	
only $d_1 \leq 7$	22.4	
only $d_2 \leq 7$	18.2	
all large	18.2	
sequential 000 to 999	11.2	small penalty
random	14.13	normal penalty

GBCD to BCD – Algorithm

Goal: Convert 10-bit GBCD to 12-bit BCD encoding.

GBCD to BCD – Algorithm

Goal: Convert 10-bit GBCD to 12-bit BCD encoding.

GBCD		BCD
$x = 10^2 d_2 + 10 d_1 + d_0$	$\rightarrow$	$y = 16^2 d_2 + 16 d_1 + d_0$

GBCD to BCD – Algorithm

Goal: Convert 10-bit GBCD to 12-bit BCD encoding.

GBCD		BCD
$x = 10^2 d_2 + 10 d_1 + d_0$	$\rightarrow$	$y = 16^2 d_2 + 16 d_1 + d_0$

$$
\begin{aligned}
z_1 &\stackrel{\text{def}}{=} 10 d_2 + d_1 = \lfloor x/10 \rfloor \\
z_2 &\stackrel{\text{def}}{=} d_2 = \lfloor x/100 \rfloor \\
y &\leftarrow x + 3(2 z_1 + 32 z_2)
\end{aligned}
$$

GBCD to BCD – Implementation

!On input R0 and has value x = 100 d_2 + 10 d_1 + d_0

```
mov  R1,#205          (205*x)>>11 is floor(x/10)
mul  R1,R0,R1         205*x
mov  R2,#41           (41*x)>>12 is floor(x/100)
mul  R2,R0,R2         41*x
mov  R1,R1,LSR #10    (205*x)>>10
bic  R1,R1,#1         this is 2*(10 d_2 + d_1)
mov  R2,R2,LSR #12    this is d_2
add  R1,R1,R2,LSL #5  this is 32 d_2 + 2(10 d_2 + d_1)
add  R1,R1,R1,LSL #1  this is 156 d_2 + 6 d_1
add  R0,R0,R1         !this is 256 d_2 + 16 d_1 + d_0
```

DPD/GBCD to BCD – Performance

	Latency (cycles)	
Cases	DPD	GBCD
Best case	6.3	10.2
Worst case	20.8	10.2
Sequence average	11.2	10.2
Random average	14.1	10.2
Code size	55 inst.	11 inst.