SHARC Internal Power Consumption Measurements

Last Modified: 6/6/96

There have been many customer inquiries concerning typical power consumption of the SHARC processors. As you all know we only supply a maximum Pint which is based on experimentation. Iddin is measured while executing a radix-2 FFT butterfly with instruction in cache, one data fetch from each block of memory and a DMA transfer from internal memory to internal memory. A similar method of experimentation to try to determine "typical power".

The first issue was to determine what "typical" instructions would be so "typical" power can be determined. A guess was taken. The following is a description of the test cases:

Test Case 1 executes an addition, a subtraction, a PM data access and a DM data access.

Test Case 2 executes a multiplication, an addition, a PM data access and a DM data access.

Test case 3 executes a multiplication, an addition, a subtraction, a PM data access and a DM data access.

Test Case 4 executes a multiplication, an addition and a subtraction.

Test Case 5 executes an addition and a subtraction.

Test Case 6 executes a PM data access and a DM data access.

Test cases were selected assuming "typical" instructions would be associated with number crunching. A jump statement was used to sustain these instructions.

The experiments where performed on an ADSP 21062 rev 2.0 using 3 separate clock rates, 25MHz, 33 MHz, and 40 MHz. (A rev 0.6 part with a 24 MHz clock was also tested. The results were almost identical to those of the rev 2.0). Vddin was fixed at 5.25v. The following table describes the results:

<table>
<thead>
<tr>
<th>Test Case</th>
<th>Iddin @ 24 MHz</th>
<th>Iddin @ 33 MHz</th>
<th>Iddin @ 40 MHz</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>380 mA</td>
<td>410 mA</td>
<td>470 mA</td>
</tr>
<tr>
<td>2</td>
<td>400 mA</td>
<td>440 mA</td>
<td>500 mA</td>
</tr>
<tr>
<td>3</td>
<td>400 mA</td>
<td>440 mA</td>
<td>510 mA</td>
</tr>
<tr>
<td>4</td>
<td>280 mA</td>
<td>330 mA</td>
<td>370 mA</td>
</tr>
<tr>
<td>5</td>
<td>280 mA</td>
<td>320 mA</td>
<td>360 mA</td>
</tr>
<tr>
<td>6a (50% Switching)</td>
<td>320 mA</td>
<td>380 mA</td>
<td>420 mA</td>
</tr>
<tr>
<td>6b (100% Switching)</td>
<td>390 mA</td>
<td>440 mA</td>
<td></td>
</tr>
</tbody>
</table>

The following is a copy of the program used.

```
#include "def21060.h"
#define N 22

 overposting

.endseg;

赖zept     seg_rth;
.nop;
.jump start;
```
.ENDSEG;

.SEGMENT/PM    seg_pmco;

start:
   l0=@buffdm;
   b0= buffdm;
   m0= 0x1;

   l8=@buffpm;
   b8= buffpm;
   m8= 0x1;

   r0=dm(i0,m0), r4 =pm(i8,m8);
   r8=dm(i0,m0), r12=pm(i8,m8);

   call addsub;

addsub:
   r7=r0+r4, r15=r0-r4, r0=dm(i0,m0), r4 =pm(i8,m8);
   jump addsub (db);
   r7=r0+r4, r15=r0-r4, r0=dm(i0,m0), r4 =pm(i8,m8);
   r7=r0+r4, r15=r0-r4, r0=dm(i0,m0), r4 =pm(i8,m8);

mulacc:
   r7=r0*r4(SSFR), r15=r8+r12, r0=dm(i0,m0), r4 =pm(i8,m8);
   jump mulacc (db);
   r7=r0*r4(SSFR), r15=r8+r12, r0=dm(i0,m0), r4 =pm(i8,m8);
   r7=r0*r4(SSFR), r15=r8+r12, r0=dm(i0,m0), r4 =pm(i8,m8);

mas:  r7=r0*r4(SSFR), r15=r8+r12, r14=r8-r12,
      r0=dm(i0,m0), r4 =pm(i8,m8);
      jump mas (db);
      r7=r0*r4(SSFR), r15=r8+r12, r14=r8-r12,
      r0=dm(i0,m0), r4 =pm(i8,m8);
      r7=r0*r4(SSFR), r15=r8+r12, r14=r8-r12,
      r0=dm(i0,m0), r4 =pm(i8,m8);

.ENDSEG;