Using the RCCE Power Management Calls
revision 1.1

Revision History for Document

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>Initial release.</td>
</tr>
<tr>
<td>1.1</td>
<td>Added definition of triple typedef</td>
</tr>
</tbody>
</table>

1 Introduction

The RCCE power management interface does not allow you to set SCC cores to a specific voltage. What RCCE does provide are calls to set the frequency divider.

What is the frequency divider? The frequency is defined relative to a global reference clock that is set when the SCC platform first starts up. This reference clock can vary, but in almost all cases it is 1.6Ghz. The frequency divider is the integer with which you divide the reference clock.

When you intialized the SCC (with sccBmc –i), you chose a tile frequency of of either 800 MHz or 533 MHz. The choices you have are taken from the *.rlb files in /opt/sccKit/current/settings. When the tile frequency is 800 MHz, the frequency divider is 2; when the tile frequency is 533 MHx, the frequency divider is 3.

RCCE_iset_power() sets the frequency divider and also modifies the voltage for the tile. It sets the voltage to the lowest voltage consistent with the frequency setting.

Note also that the RCCE power management calls change the frequency and voltage in a RCCE power domain. The RCCE power domain is a 2x2 array of tiles (8 cores), numbered from 0 through 5. The RCCE numbering is different from the SCC numbering.

RCCE numbers the domains continuously starting at 0 and moving right for the first row and then again for the second row. RCCE power domain 0 contains cores 0, 1, 2, 3, 12, 13, 14, and 15.

<table>
<thead>
<tr>
<th>RCCE: 3</th>
<th>RCCE: 4</th>
<th>RCCE: 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCC: 0</td>
<td>SCC: 1</td>
<td>SCC: 3</td>
</tr>
<tr>
<td>RCCE: 0</td>
<td>RCCE: 10</td>
<td>RCCE: 2</td>
</tr>
<tr>
<td>SCC: 4</td>
<td>SCC: 5</td>
<td>SCC: 7</td>
</tr>
</tbody>
</table>

The SCC starts numbers the power domains starting in the upper right. The numbering is not continuous. The SCC skips the numbers 2 and 6.

When you check voltage values with the sccBMC status command, the output is displayed with the SCC numbering scheme. For example if you change the voltage and frequency in RCCE power domain 0, the sccBmc status command will show a change in OPVR VCC4 under Tertiary supplies.
2 Examples

A consequence of the way RCCE sets voltage and frequency is that if you set the frequency divider to the very same value you initialized with, the voltage decreases, and you cannot return the voltage to its original value with RCCE.

Here is an example. Assume that you initialized your SCC system with Tile533_Mesh800_DDR800. The tiles are then running at 533 MHz which is a frequency divider of 3. So if you now execute

RCCE_iset_power (3, &request, &newFreqDiv, &newVoltageDiv);

The tile frequency does not change, but the voltage does. Why does the voltage change? It’s a consequence of the way RCCE_iset_power() works. RCCE_iset_power() calls an internal RCCE routeine called RC_voltage_level with Fdiv as an argument and returns a Vlevel as in

\[ Vlevel = RC\_voltage\_level(Fdiv); \]

Vlevel is an index into an array of voltage values and frequencies. Vlevel is the index of the first entry whose frequency is greater than or equal to the reference clock divided by the specified frequency divider. In this case, that index is 1. The specified voltage is then 0.8.

```c
triple RC\_V\_MHz\_cap[] = {
    /* 0 */ {0.7, 0x70, 460},
    /* 1 */ {0.8, 0x80, 598},
    /* 2 */ {0.9, 0x90, 644},
    /* 3 */ {1.0, 0xA0, 748},
    /* 4 */ {1.1, 0xB0, 875},
    /* 5 */ {1.2, 0xC0, 1024},
    /* 6 */ {1.3, 0xD0, 1198}
};
```

where

```c
typedef struct {
    float volt;
    int   VID;
    int   MHz_cap;
} triple;
```

Here is an example of the program’s invocation with some debug prints. Note that only rck00 has reasonable values for the printed values. That’s because rck00 is the power domain master.

tekubasx@marc101:/shared/tekubasx/Power$ rccerun -nue 4 -f rc.hosts powertest
pssh -h PSSH\_HOST\_FILE.6084 -t -1 -p 4 /shared/tekubasx/Power/mpb.6084 < /dev/null
pssh -h PSSH\_HOST\_FILE.6084 -t -1 -P -p 4 /shared/tekubasx/Power/powertest 4 0.533 00 01 02 03 < /dev/null
rck00: UE 0, Core ID 0; size of V dom 0 is 4, size of F dom 0 is 2
rck01: UE 1, Core ID 1; size of V dom 0 is 4, size of F dom 0 is 2
rck02: UE 2, Core ID 2; size of V dom 0 is 4, size of F dom 1 is 2
rck03: UE 3, Core ID 3; size of V dom 0 is 4, size of F dom 1 is 2
rck01: newVoltageDiv, newFreqDiv: 0 0
Now if you look at status with `sccBmc --c status`, you see the following.

```
Tertiary supplies:
  OPVR VCC0: 1.0948 V
  OPVR VCC1: 1.0939 V
  OPVR VCC2: 1.0913 V
  OPVR VCC3: 1.0936 V
  OPVR VCC4: 0.8421 V
  OPVR VCC5: 1.0897 V
  OPVR VCC7: 1.0870 V
```

The frequency stayed the same but the voltage decreased. Can you increase the voltage in RCCE power domain 0 with a RCCE call? Yes, you can but then you must also increase the frequency. If you execute

```
RCCE_iset_power (2, &request, &newFreqDiv, &newVoltageDiv);
```

The frequency increases to 800 MHz and the voltage to about 1.1 V.

```
Tertiary supplies:
  OPVR VCC0: 1.0964 V
  OPVR VCC1: 1.0955 V
  OPVR VCC2: 1.0958 V
  OPVR VCC3: 1.0950 V
  OPVR VCC4: 1.1114 V
  OPVR VCC5: 1.0960 V
  OPVR VCC7: 1.0945 V
```

Re-loading SCC Linux will not return the voltage to its original value. Nor will re-initializing the SCC. You have to cycle the SCC power.

But there is another way. You have to record the initial voltage value and then use memory-mapped I/O to change the voltage directly for that power domain. Has anyone done this?