MIPS MT Training

Policy Manager

www.mips.com

This section covers the policy manager implementations of the MT K cores.

MT Policy Manager

- **A Policy Manager (PM) is tasked with giving longer-term hints to the Dispatch Scheduler.**
  - The Policy Manager's objective is to achieve whatever performance allocation is desired in the system.
  - The Policy Manager is external to the core.
- **MIPS will delivers 2 choices of a policy manager.**
  - Round-Robin
  - Weighted Round-Robin
- **In addition, the you may design your own Policy Manager.**

MIPS 2

The policy manager is a device that works with the Dispatch Scheduler of the Core.

+ The objective of the Policy manager is to help you control the performance of each thread in you system so you can achieve you desired performance allocation of the CPU. This is commonly called Quality of Service or QOS.

+ The policy is a device external to the core so it may be changed to suite your needs.

+ The core includes a choice of 2 types of policy managers.

+ A Basis Round Robin policy manager that gives equal weight to all threads in the system

+ or a Weighted Round Robin Policy manager where threads are placed in different priority groups.

The rest of this section will cover the use of these 2 policy managers.

+ Your design can change either of these policy managers or you can create your own.

MT Basic Round-Robin

- **Round-Robin Policy Manager**
  - All TCs are assigned to the same priority level.
  - All TCs are statically given the same weight and bandwidth, and will be fairly allocated amongst all runnable TCs.
  - This PM does not implement the thread-scheduling CP0 registers (*TCSchedule and VPESchedule)*. Writes to these registers will be ignored. Reads from these registers will return -1.

The Round Robin Policy manager implements a equal scheduling algorithm.

+ All threads have the same priority

+ all thread have the same weight

+ the 2 thread scheduling registers TCSchedule and VPESchedule have no effect for it an dreads from these registers will return a -1, or all bits set.

MT Weighted Round-Robin

- **Weighted Round-Robin Policy Manager**
  - Threads are broken down into 4 groups. The Threads in the highest numbered group have the highest priority
  - TCs in the highest priority numbered group will run in round-robin order until none are runnable. Then TCs in the next highest priority group will run, and so forth.
  - A TC is assigned to a group using the GPR field in the CP0 register *TCSchedule*.

MIPS 4

The Weighted Round Robin Policy manager

+ A thread can belong to one of 4 groups, 0 through 3 with 3 being the highest priority group.

+ Each Thread in the highest priority group will run in a round robin fashion until there are no threads runnable within that group. Then threads in the next group will run in the same fashion.

+ A thread is assigned to a group by the GPR field in its TCSchedule CP0 register.

MT Weighted Round-Robin

- **Issues with Static Priorities**
  - A Group's priorities are static and are assigned according to group number (Group 0 will be assign the lowest priority.)
  - Group starvation can take place.
    - TCs in lower priority groups will only run if there are no runnable TCs in a higher priority group.

The simple Weight round robin as described in the previous slide might cause a thread to be starved of any cycles.

It could also cause a deadlock condition if a higher priority thread is dependent on a action of a lower priority thread and the lower priority thread never gets to run because other higher priority threads are always runnable.

MT Weighted Round-Robin GPO

- **Group Priority Override**
  - Group Priority Override is set or cleared using the GPO bit in the VPESchedule CP0 register.
  - When GPO is cleared, priorities are rotated between each group.
  - TCs in the next higher group will get at least twice the bandwidth of the TCs in the lower group.
  - A group may have more than one runnable TC in it. Each runnable TC within a group will run one cycle when that group is the highest in priority.

MIPS 6

To help solve the problem of thread starvation the Weighted Round Robin Policy manager can be configured to rotate priorities but still give the highest numbered groups more, but not all of the CPU time. This guaranties all threads will run but some will run more Then others.

+ Rotating Priorities are controlled by the GPO bit in the VPESchedule CP0 register. When this bit is set static group priorities are used. When it is cleared Priorities are rotated between groups.

+ The threads in the Highest Numbered group get a chance to run twice as much as the next lower priority group.

+ Since each group can have more then one thread assigned to it for a rotation where a group has the highest priority all runnable threads within that group will get to run.

MT Weighted Round-Robin Table

- **Table of priorities per rotation (GPO Cleared):**
  - Highest number within a rotation is the highest priority.
  - If there are no threads runnable at the highest priority then the next runnable thread at the next highest priority will run.

Rotation	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Group 0 (Lowest group)	0	1	0	1	0	1	0	3	0	1	0	1	0	1	0
Group 1	1	0	1	3	1	0	2	1	2	0	1	3	1	0	2
Group 2	2	3	2	0	2	3	1	2	1	3	2	0	2	3	1
Group 3 (Highest group)	3	2	3	2	3	2	3	0	3	2	3	2	3	2	3

Here is table that illustrates the priority rotation.

Each column is a rotation and shows the group priority for the rotation. The highest priority is circled.

Lets look at column one. The number in the column represents the priority the group has for that rotation so here you can see the priorities match the group number therefore group 3 has the highest priority and Group 2 the next and so forth.

In column 2 the priorities have rotated so now group 2 has the highest priority .

To go through a full rotation cycle it takes 15 rotations. Note this is not processor cycles as you will see on the next slide.

After 15 rotations you can see the group 3 had the highest priority twice as much as group 2, 8 times verses 4 and group 2 was twice as much as group 1, 4 times verses 2 and 1 was twice that of group 0, 2 times verses 1.

MT Weighted Round-Robin Table

- **Example of a 4 TC system with 3 TCs in Group 1 and 1 TC in Group 0.**

Cycle Count	Rotation Count	Group 3 Priority (Highest)	Group 2 Priority	Group 1 Priority	Group 0 Priority (Lowest)
1	1	3	2	1	0
2	1	3	2	1	0
3	1	3	2	1	0
4	2	3	2	0	1
5	3	3	2	1	0
6	3	3	2	1	0
7	3	3	2	1	0
8	4	2	0	3	1
9	4	2	0	3	1

The chart shows the cycle effect of having more than one TC in a group. There are 4 threads in this system. Three of the threads have been assigned to group 1 and one of the threads is assigned to group 0. The first column shows the cycle count the second column shows us which rotation out of the 15 the policy manager is in and the remaining columns show the priority of each group for each cycle.

+ Lets look at row one. As we saw in the previous slide the group number and the priority number match for this first rotation . Since there are no threads in group 3 or 2 the highest priority is group 1.

+ Because there are 3 threads on group 1 each of those threads gets 1 cycle each so the threads in group 1 use up the first 3 cycles.

+ Once all threads that are runnable in group 1 have run the priority rotates at cycle 4. This time group 0 has the highest priority so the 1 thread that is in group 0 runs for one cycle.

+ Since there is only 1 thread in group 0 the priority will rotate again at cycle 5 where group is again the highest priority group with runnable threads. Each thread in group 1 run one cycle in turn.

Then the priority rotates again at cycle 8 but for this rotation group 1 is still the highest priority group with runnable threads so it will run for 3 more cycles.

MT Weighted Round-Robin Table

- **Example Continued:**

Cycle Count	Rotation Count	Group 3 Priority	Group 2 Priority	Group 1 Priority	Group 0 Priority
10	4	2	0	3	1
11	5	3	2	1	0
12	5	3	2	1	0
13	5	3	2	1	0
14	6	2	3	0	1
15	7	3	1	2	0
16	7	3	1	2	0
17	7	3	1	2	0
18	8	0	2	1	3
19	9	3	1	2	0

MIPS

9

This slide and the next continue to show the rotations and the running group for each cycle.

MT Weighted Round-Robin Table

- **Example Continued:**

Cycle Count	Rotation Count	Group 3 Priority	Group 2 Priority	Group 1 Priority	Group 0 Priority
20	9	3	1	2	0
21	9	3	1	2	0
22	10	2	3	0	1
23	11	3	2	1	0
24	11	3	2	1	0
25	11	3	2	1	0
26	12	2	0	3	1
27	12	2	0	3	1
28	12	2	0	3	1
29	13	3	2	1	0

MIPS

MT Weighted Round-Robin Table

- **Example Continued:**

Cycle Count	Rotation Count	Group 3 Priority	Group 2 Priority	Group 1 Priority	Group 0 Priority
30	13	3	2	1	0
31	13	3	2	1	0
32	14	2	3	0	1
33	15	3	1	2	0
34	15	3	1	2	0
35	15	3	1	2	0
Total Cycles Group Ran		0	0	30	5

MIPS 11

This slide show the end of the rotation and totals the number of cycles each Group used.

What we see is each thread in group 1 ran twice a much as a thread in group 0.

+ Since there were 3 threads in group 1 and 1 thread in group 0, then to complete the 15 integrations of a full rotation cycle it took 35 CPU cycles.

+ out of the 35 cycles the 3 threads in group 1 had the higher priority to run for total of 30 cycles each thread in the group getting a equal share. Each thread then ran for 10 cycles.

+ out of the 35 cycles the single thread in group 0 had the higher priority to run for 5 cycles.

+ so you can see that each thread in group 1 ran for 10 cycles and each thread in group 0 ran for 5 cycles which accounts for any thread in group 1 running twice as much as a thread in group 0.

MT Weighted Round-Robin Run Table Explained

- **Each thread in Group 1 gets twice as much time as the thread in Group 0:**
  - The full rotation required 35 cycles to complete.
  - Out of these 35 cycles, group1 is higher priority than group0 for 30 cycles.
  - Group1 contains 3 TCs which will all run, whenever Group1 has the highest priority in a run round-robin fashion,
  - Group 0s 1 TC has higher priority for 5 cycles.
  - Threads in group 1 run for 10 cycles each and threads in group 0 run for 5 cycles each.

MIPS 12

Let me summaries what the previous slide demonstrated.

+ What we see is each thread in group 1 ran twice a much as a thread in group 0.

+ Since there were 3 threads in group 1 and 1 thread in group 0, then to complete the 15 integrations of a full rotation cycle it took 35 CPU cycles.

+ out of the 35 cycles the 3 threads in group 1 had the higher priority to run for total of 30 cycles each thread in the group getting a equal share. Each thread then ran for 10 cycles.

+ out of the 35 cycles the single thread in group 0 had the higher priority to run for 5 cycles.

+ so you can see that each thread in group 1 ran for 10 cycles and each thread in group 0 ran for 5 cycles which accounts for any thread in group 1 running twice as much as a thread in group 0.