[v4,2/2] i386: Add support to get/set/migrate Intel Processor Trace feature

diff (/patch/881308/raw/) | mbox (/patch/881308/mbox/) | series (/series/31863/mbox/)

Message ID	1520182116-16485-2-git-send-email-luwei.kang@intel.com
State	New
Headers	show
Series	Untitled series #31863
Related	show

Commit Message

Kang, Luwei (/project/qemu-devel/list/?submitter=69591) March 4, 2018, 4:48 p.m.

```
From: Chao Peng <chao.p.peng@linux.intel.com>

Add Intel Processor Trace related definition. It also add
corresponding part to kvm_get/set_msr and vmstate.

Signed-off-by: Chao Peng <chao.p.peng@linux.intel.com>
Signed-off-by: Luwei Kang <luwei.kang@intel.com>
---
 target/i386/cpu.h     | 22 ++++++++++++++++++++
 target/i386/kvm.c     | 51 +++++++++++++++++++++++++++++++++++++++++++++++++++
 target/i386/machine.c | 38 ++++++++++++++++++++++++++++++++++++++
 3 files changed, 111 insertions(+)
```

Patch

diff (/patch/881308/raw/) | mbox (/patch/881308/mbox/) | series (/series/31863/mbox/)

```
diff --git a/target/i386/cpu.h b/target/i386/cpu.h
index f5b9ecb..5457d3b 100644
--- a/target/i386/cpu.h
+++ b/target/i386/cpu.h
@@ -415,6 +415,21 @@  typedef enum X86Seg {
 #define MSR_MC0_ADDR                    0x402
 #define MSR_MC0_MISC                    0x403

+#define MSR_IA32_RTIT_OUTPUT_BASE       0x560
+#define MSR_IA32_RTIT_OUTPUT_MASK       0x561
+#define MSR_IA32_RTIT_CTL               0x570
+#define MSR_IA32_RTIT_STATUS            0x571
+#define MSR_IA32_RTIT_CR3_MATCH         0x572
+#define MSR_IA32_RTIT_ADDR0_A           0x580
+#define MSR_IA32_RTIT_ADDR0_B           0x581
+#define MSR_IA32_RTIT_ADDR1_A           0x582
+#define MSR_IA32_RTIT_ADDR1_B           0x583
+#define MSR_IA32_RTIT_ADDR2_A           0x584
+#define MSR_IA32_RTIT_ADDR2_B           0x585
+#define MSR_IA32_RTIT_ADDR3_A           0x586
+#define MSR_IA32_RTIT_ADDR3_B           0x587
+#define MAX_RTIT_ADDRS                  8
+
 #define MSR_EFER                        0xc0000080

 #define MSR_EFER_SCE   (1 << 0)
@@ -1154,6 +1169,13 @@  typedef struct CPUX86State {
     uint64_t msr_hv_stimer_config[HV_STIMER_COUNT];
     uint64_t msr_hv_stimer_count[HV_STIMER_COUNT];

+    uint64_t msr_rtit_ctrl;
+    uint64_t msr_rtit_status;
+    uint64_t msr_rtit_output_base;
+    uint64_t msr_rtit_output_mask;
+    uint64_t msr_rtit_cr3_match;
+    uint64_t msr_rtit_addrs[MAX_RTIT_ADDRS];
+
     /* exception/interrupt handling */
     int error_code;
     int exception_is_int;
diff --git a/target/i386/kvm.c b/target/i386/kvm.c
index f9f4cd1..097c953 100644
--- a/target/i386/kvm.c
+++ b/target/i386/kvm.c
@@ -1811,6 +1811,25 @@  static int kvm_put_msrs(X86CPU *cpu, int level)
                 kvm_msr_entry_add(cpu, MSR_MTRRphysMask(i), mask);
             }
         }
+        if (env->features[FEAT_7_0_EBX] & CPUID_7_0_EBX_INTEL_PT) {
+            int addr_num = kvm_arch_get_supported_cpuid(kvm_state,
```

```
+                                                0x14, 1, R_EAX) & 0x7;
+
+            kvm_msr_entry_add(cpu, MSR_IA32_RTIT_CTL,
+                              env->msr_rtit_ctrl);
+            kvm_msr_entry_add(cpu, MSR_IA32_RTIT_STATUS,
+                              env->msr_rtit_status);
+            kvm_msr_entry_add(cpu, MSR_IA32_RTIT_OUTPUT_BASE,
+                              env->msr_rtit_output_base);
+            kvm_msr_entry_add(cpu, MSR_IA32_RTIT_OUTPUT_MASK,
+                              env->msr_rtit_output_mask);
+            kvm_msr_entry_add(cpu, MSR_IA32_RTIT_CR3_MATCH,
+                              env->msr_rtit_cr3_match);
+            for (i = 0; i < addr_num; i++) {
+                kvm_msr_entry_add(cpu, MSR_IA32_RTIT_ADDR0_A + i,
+                              env->msr_rtit_addrs[i]);
+            }
+        }

         /* Note: MSR_IA32_FEATURE_CONTROL is written separately, see
          *       kvm_put_msr_feature_control. */
@@ -2124,6 +2143,20 @@  static int kvm_get_msrs(X86CPU *cpu)
         }
     }

+    if (env->features[FEAT_7_0_EBX] & CPUID_7_0_EBX_INTEL_PT) {
+        int addr_num =
+            kvm_arch_get_supported_cpuid(kvm_state, 0x14, 1, R_EAX) & 0x7;
+
+        kvm_msr_entry_add(cpu, MSR_IA32_RTIT_CTL, 0);
+        kvm_msr_entry_add(cpu, MSR_IA32_RTIT_STATUS, 0);
+        kvm_msr_entry_add(cpu, MSR_IA32_RTIT_OUTPUT_BASE, 0);
+        kvm_msr_entry_add(cpu, MSR_IA32_RTIT_OUTPUT_MASK, 0);
+        kvm_msr_entry_add(cpu, MSR_IA32_RTIT_CR3_MATCH, 0);
+        for (i = 0; i < addr_num; i++) {
+            kvm_msr_entry_add(cpu, MSR_IA32_RTIT_ADDR0_A + i, 0);
+        }
+    }
+
     ret = kvm_vcpu_ioctl(CPU(cpu), KVM_GET_MSRS, cpu->kvm_msr_buf);
     if (ret < 0) {
         return ret;
@@ -2364,6 +2397,24 @@  static int kvm_get_msrs(X86CPU *cpu)
         case MSR_IA32_SPEC_CTRL:
             env->spec_ctrl = msrs[i].data;
             break;
+        case MSR_IA32_RTIT_CTL:
+            env->msr_rtit_ctrl = msrs[i].data;
+            break;
+        case MSR_IA32_RTIT_STATUS:
+            env->msr_rtit_status = msrs[i].data;
+            break;
```

```
+        case MSR_IA32_RTIT_OUTPUT_BASE:
+            env->msr_rtit_output_base = msrs[i].data;
+            break;
+        case MSR_IA32_RTIT_OUTPUT_MASK:
+            env->msr_rtit_output_mask = msrs[i].data;
+            break;
+        case MSR_IA32_RTIT_CR3_MATCH:
+            env->msr_rtit_cr3_match = msrs[i].data;
+            break;
+        case MSR_IA32_RTIT_ADDR0_A ... MSR_IA32_RTIT_ADDR3_B:
+            env->msr_rtit_addrs[index - MSR_IA32_RTIT_ADDR0_A] = msrs[i].data;
+            break;
         }
     }

diff --git a/target/i386/machine.c b/target/i386/machine.c
index 361c05a..c05fe6f 100644
--- a/target/i386/machine.c
+++ b/target/i386/machine.c
@@ -837,6 +837,43 @@  static const VMStateDescription vmstate_spec_ctrl = {
     }
 };

+static bool intel_pt_enable_needed(void *opaque)
+{
+    X86CPU *cpu = opaque;
+    CPUX86State *env = &cpu->env;
+    int i;
+
+    if (env->msr_rtit_ctrl || env->msr_rtit_status ||
+        env->msr_rtit_output_base || env->msr_rtit_output_mask ||
+        env->msr_rtit_cr3_match) {
+        return true;
+    }
+
+    for (i = 0; i < MAX_RTIT_ADDRS; i++) {
+        if (env->msr_rtit_addrs[i]) {
+            return true;
+        }
+    }
+
+    return false;
+}
+
+static const VMStateDescription vmstate_msr_intel_pt = {
+    .name = "cpu/intel_pt",
+    .version_id = 1,
+    .minimum_version_id = 1,
+    .needed = intel_pt_enable_needed,
+    .fields = (VMStateField[]) {
+        VMSTATE_UINT64(env.msr_rtit_ctrl, X86CPU),
```

```
+        VMSTATE_UINT64(env.msr_rtit_status, X86CPU),
+        VMSTATE_UINT64(env.msr_rtit_output_base, X86CPU),
+        VMSTATE_UINT64(env.msr_rtit_output_mask, X86CPU),
+        VMSTATE_UINT64(env.msr_rtit_cr3_match, X86CPU),
+        VMSTATE_UINT64_ARRAY(env.msr_rtit_addrs, X86CPU, MAX_RTIT_ADDRS),
+        VMSTATE_END_OF_LIST()
+    }
+};
+
 VMStateDescription vmstate_x86_cpu = {
     .name = "cpu",
     .version_id = 12,
@@ -957,6 +994,7 @@  VMStateDescription vmstate_x86_cpu = {
 #endif
         &vmstate_spec_ctrl,
         &vmstate_mcg_ext_ctl,
+        &vmstate_msr_intel_pt,
         NULL
     }
 };
```