CESG SW Hyperscale PRC Tencent Engineering Team

How to enable IAX on SPR

Version 1.0

Ji, Yu
8-11-2021

How to enable IAX on SPR

1. Kernel build and IAX device check

CPU info:

```
        Architecture:        x86_64
        CPU op-mode(s):      32-bit, 64-bit
        Byte Order:          Little Endian
        CPU(s):              112
        On-line CPU(s) list: 0-111
        Thread(s) per core:  1
        Core(s) per socket:  56
        Socket(s):           2
        NUMA node(s):        2
        Vendor ID:           GenuineIntel
        BIOS Vendor ID:      Intel(R) Corporation
        CPU family:          6
        Model:               143
        Model name:          Genuine Intel(R) CPU $0000%@
        BIOS Model name:     Genuine Intel(R) CPU $0000%@
        Stepping:            2
        CPU MHz:             1700.000
        CPU max MHz:         3600.0000
        CPU min MHz:         800.0000
        BogoMIPS:            3400.00
        Virtualization:      VT-x
        L1d cache:           48K
        L1i cache:           32K
        L2 cache:            2048K
        L3 cache:            107520K
        NUMA node0 CPU(s):   0-55
        NUMA node1 CPU(s):   56-111
Flags:          fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat
pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm
constant_tsc art arch_perfmon pebs bts rep_good nopl xtopology nonstop_tsc cpuid
aperfmperf tsc_known_freq pni pclmulqdq dtes64 monitor ds_cpl vmx smx est tm2
ssse3 sdbg fma cx16 xtpr pdcm pcid sse4_1 sse4_2 x2apic movbe popcnt
tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm abm 3dnowprefetch
cpuid_fault epb cat_l3 cat_l2 cdp_l3 invpcid_single intel_ppin cdp_l2 ssbd mba ibrs
ibpb stibp ibrs_enhanced tpr_shadow vnmi flexpriority ept vpid ept_ad fsgsbase
tsc_adjust bmi1 avx2 smep bmi2 erms invpcid cqm rdt_a avx512f avx512dq rdseed
adx smap avx512ifma clflushopt clwb intel_pt avx512cd sha_ni avx512bw avx512vl
xsaveopt xsavec xgetbv1 xsaves cqm_llc cqm_occup_llc cqm_mbm_total
```

```
cqm_mbm_local split_lock_detect avx_vnni avx512_bf16 wbnoinvd dtherm ida arat
pln pts hwp hwp_act_window hwp_epp hwp_pkg_req avx512vbmi umip pku ospke
waitpkg avx512_vbmi2 gfni vaes vpclmulqdq avx512_vnni avx512_bitalg tme
avx512_vpopcntdq la57 rdpid bus_lock_detect cldemote movdiri movdir64b enqcmd
fsrm avx512_vp2intersect md_clear serialize tsxldtrk pconfig arch_lbr avx512_fp16
flush_l1d arch_capabilities
```

BOARD info:

```
# dmidecode 3.2
Getting SMBIOS data from sysfs.
SMBIOS 3.3.0 present.
# SMBIOS implementations newer than version 3.2.0 are not
# fully supported by this version of dmidecode.

Handle 0x0002, DMI type 2, 15 bytes
Base Board Information
	Manufacturer: Intel
	Product Name: VulcanCity
	Version: Default string
	Serial Number: Default string
	Asset Tag: Default string
	Features:
		Board is a hosting board
		Board is replaceable
	Location In Chassis: Default string
	Chassis Handle: 0x0003
	Type: Motherboard
	Contained Object Handles: 0

Handle 0x0018, DMI type 10, 24 bytes
On Board Device 1 Information
	Type: Unknown
	Status: Enabled
	Description: Device 1
On Board Device 2 Information
	Type: Unknown
	Status: Enabled
	Description: Device 2
On Board Device 3 Information
	Type: Unknown
	Status: Enabled
	Description: Device 3
On Board Device 4 Information
```

```
        Type: Unknown
        Status: Enabled
        Description: Device 4
On Board Device 5 Information
        Type: Unknown
        Status: Enabled
        Description: Device 5
On Board Device 6 Information
        Type: Unknown
        Status: Enabled
        Description: Device 6
On Board Device 7 Information
        Type: Unknown
        Status: Enabled
        Description: Device 7
On Board Device 8 Information
        Type: Unknown
        Status: Enabled
        Description: Device 8
On Board Device 9 Information
        Type: Unknown
        Status: Enabled
        Description: Device 9
On Board Device 10 Information
        Type: Unknown
        Status: Enabled
        Description: <BAD INDEX>

Handle 0x0092, DMI type 41, 11 bytes
Onboard Device
        Reference Designation: Onboard ETHERNET Controller
        Type: Ethernet
        Status: Enabled
        Type Instance: 1
        Bus Address: 0000:01:00.0

Handle 0x0093, DMI type 41, 11 bytes
Onboard Device
        Reference Designation: Onboard VGA
        Type: Video
        Status: Enabled
        Type Instance: 1
        Bus Address: 0000:03:00.0
Handle 0x0094, DMI type 41, 11 bytes
Onboard Device
        Reference Designation: Onboard SATA Controller
        Type: SATA Controller
        Status: Enabled
        Type Instance: 1
        Bus Address: 0000:00:17.0

Handle 0x0095, DMI type 41, 11 bytes
Onboard Device
        Reference Designation: Onboard SATA Controller
        Type: SATA Controller
        Status: Enabled
        Type Instance: 2
 Bus Address: 0000:00:18.0
```

1.1 Configure BIOS

- "EDKII Menu" -> "Socket Configuration" -> "IIO Configuration" -> "Intel VT for directed IO (VT -d)" → Intel VT for directed IO → Enable
- "EDKII Menu" -> "Socket Configuration" -> "IIO Configuration" -> "PCI ENQCMD/ENQCMDS" → Enable
- "EDKII Menu" -> "Socket Configuration" -> "Processor Configuration" -> "Enable LP Global" → Single LP

1.2 Prepare to build kernel

- yum group install "Development Tools"
- yum install ncurses-devel bison flex elfutils-libelf-devel openssl-devel
- kernel version Linux 5.13.0-dsa-micros+ x86_64

- git clone https://github.com/torvalds/linux.git
- git checkout v5.13
- git revert 9bfecd05833918526cc7357d55e393393440c5fa
- cd linux
- cp /boot/config-4.18.0-305.3.1.el8.x86_64 .config
- make menuconfig
- Check build options:
  ```
  CONFIG_IRQ_REMAP=y
  CONFIG_INTEL_IOMMU=y
  CONFIG_INTEL_IOMMU_SVM=y
  CONFIG_IMS_MSI=y
  CONFIG_IMS_MSI_ARRAY=y
  CONFIG_IRQ_REMAP=y
  CONFIG_PCI_ATS=y
  ```

```
CONFIG_PCI_PRI=y
CONFIG_PCI_PASID=y
CONFIG_INTEL_IDXD=m
CONFIG_INTEL_IDXD_SVM=y
CONFIG_VFIO_MDEV_IDXD=m
CONFIG_INTEL_IDXD_KTEST=m
CONFIG_DMA_ENGINE=m
CONFIG_DMATEST=m
CONFIG_IRQ_BYPASS_MANAGER=m
CONFIG_AUXILIARY_BUS=y
CONFIG_UACCE_IDXD=m
CONFIG_INTEL_IDXD_CDEV_LEGACY=y
```

- Additional options:
```
CONFIG_CRYPTO_DEV_IAX_CRYPTO=y
CONFIG_INTEL_IDXD_PAGE_CLEAR=m
```

1.3 Build and install kernel

- make -jX , where X is twice the number of physical cores (lscpu -> CPU(s))
- make modules
- make modules_install
- make install
- make headers_install INSTALL_HDR_PATH=/usr
- grub2-mkconfig -o /boot/grub2/grub.cfg
- grubby --set-default /boot/vmlinuz-5.13.0-dsa-micros+
- grubby --update-kernel=ALL --args="intel_iommu=on,sm_on"
- reboot
- uname -msr

1.4 IAX device check

- lspci -nn | grep 0cfe

  The result should be like this

```
6a:02.0 System peripheral [0880]: Intel Corporation Device
[8086:0cfe]
6f:02.0 System peripheral [0880]: Intel Corporation Device
[8086:0cfe]
74:02.0 System peripheral [0880]: Intel Corporation Device
[8086:0cfe]
79:02.0 System peripheral [0880]: Intel Corporation Device
[8086:0cfe]
e7:02.0 System peripheral [0880]: Intel Corporation Device
[8086:0cfe]
ec:02.0 System peripheral [0880]: Intel Corporation Device
[8086:0cfe]
```

```
f1:02.0 System peripheral [0880]: Intel Corporation Device
[8086:0cfe]
f6:02.0 System peripheral [0880]: Intel Corporation Device
[8086:0cfe]
```

You should have 2 sockets 8 IAX device

- ls /sys/bus/dsa/devices/iax*

The result should be like this:

```
/sys/bus/dsa/devices/iax1:
cdev_major   cmd_status      engine1.0   engine1.2   engine1.4
engine1.6  errors  group1.0  group1.2  max_batch_size  max_groups
max_transfer_size   max_work_queues_size   op_cap           power
subsystem   uevent  wq1.0  wq1.2  wq1.4  wq1.6
clients    configurable  engine1.1  engine1.3  engine1.5  engine1.7
gen_cap    group1.1    group1.3    max_engines         max_tokens
max_work_queues    numa_node              pasid_enabled  state
token_limit  version  wq1.1  wq1.3  wq1.5  wq1.7

/sys/bus/dsa/devices/iax11:
cdev_major   cmd_status     engine11.0   engine11.2   engine11.4
engine11.6    errors    group11.0    group11.2    max_batch_size
max_groups   max_transfer_size   max_work_queues_size   op_cap
power  subsystem   uevent  wq11.0  wq11.2  wq11.4  wq11.6
clients        configurable   engine11.1   engine11.3   engine11.5
engine11.7    gen_cap    group11.1    group11.3    max_engines
max_tokens  max_work_queues   numa_node        pasid_enabled
state  token_limit  version  wq11.1  wq11.3  wq11.5  wq11.7

/sys/bus/dsa/devices/iax13:
cdev_major   cmd_status     engine13.0   engine13.2   engine13.4
engine13.6    errors    group13.0    group13.2    max_batch_size
max_groups   max_transfer_size   max_work_queues_size   op_cap
power  subsystem   uevent  wq13.0  wq13.2  wq13.4  wq13.6
clients        configurable   engine13.1   engine13.3   engine13.5
engine13.7    gen_cap    group13.1    group13.3    max_engines
max_tokens  max_work_queues   numa_node        pasid_enabled
state  token_limit  version  wq13.1  wq13.3  wq13.5  wq13.7

/sys/bus/dsa/devices/iax15:
```

```
cdev_major   cmd_status     engine15.0   engine15.2   engine15.4
engine15.6    errors     group15.0    group15.2    max_batch_size
max_groups  max_transfer_size  max_work_queues_size  op_cap
power  subsystem   uevent  wq15.0  wq15.2  wq15.4  wq15.6
clients         configurable    engine15.1    engine15.3    engine15.5
engine15.7     gen_cap     group15.1     group15.3     max_engines
max_tokens  max_work_queues   numa_node         pasid_enabled
state  token_limit  version  wq15.1  wq15.3  wq15.5  wq15.7

/sys/bus/dsa/devices/iax3:
cdev_major    cmd_status       engine3.0    engine3.2    engine3.4
engine3.6  errors  group3.0  group3.2  max_batch_size  max_groups
max_transfer_size   max_work_queues_size   op_cap          power
subsystem   uevent   wq3.0  wq3.2  wq3.4  wq3.6
clients    configurable  engine3.1  engine3.3  engine3.5  engine3.7
gen_cap    group3.1    group3.3    max_engines         max_tokens
max_work_queues     numa_node               pasid_enabled  state
token_limit  version  wq3.1  wq3.3  wq3.5  wq3.7

/sys/bus/dsa/devices/iax5:
cdev_major    cmd_status       engine5.0    engine5.2    engine5.4
engine5.6  errors  group5.0  group5.2  max_batch_size  max_groups
max_transfer_size   max_work_queues_size   op_cap          power
subsystem   uevent   wq5.0  wq5.2  wq5.4  wq5.6
clients    configurable  engine5.1  engine5.3  engine5.5  engine5.7
gen_cap    group5.1    group5.3    max_engines         max_tokens
max_work_queues     numa_node               pasid_enabled  state
token_limit  version  wq5.1  wq5.3  wq5.5  wq5.7

/sys/bus/dsa/devices/iax7:
cdev_major    cmd_status       engine7.0    engine7.2    engine7.4
engine7.6  errors  group7.0  group7.2  max_batch_size  max_groups
max_transfer_size   max_work_queues_size   op_cap          power
subsystem   uevent   wq7.0  wq7.2  wq7.4  wq7.6
clients    configurable  engine7.1  engine7.3  engine7.5  engine7.7
gen_cap    group7.1    group7.3    max_engines         max_tokens
max_work_queues     numa_node               pasid_enabled  state
token_limit  version  wq7.1  wq7.3  wq7.5  wq7.7

/sys/bus/dsa/devices/iax9:
cdev_major    cmd_status       engine9.0    engine9.2    engine9.4
engine9.6  errors  group9.0  group9.2  max_batch_size  max_groups
max_transfer_size   max_work_queues_size   op_cap          power
subsystem   uevent   wq9.0  wq9.2  wq9.4  wq9.6
```

```
clients     configurable   engine9.1   engine9.3   engine9.5   engine9.7
gen_cap     group9.1     group9.3     max_engines          max_tokens
max_work_queues      numa_node              pasid_enabled   state
token_limit  version  wq9.1  wq9.3  wq9.5  wq9.7
```

Your IAX devices can also be in /sys/bus/iax/devices, it depends on the version of the kernel you use.

Useful link for more details:

https://wiki.ith.intel.com/pages/viewpage.action?pageId=1842481266

2. **Enable IAX**

2.1 Accel-config build

Repo: https://github.com/intel/idxd-config

- yum groupinstall "Development Tools"
- yum install autoconf automake libtool pkgconf rpm-build rpmdevtools
- yum install asciidoc xmlto libuuid-devel json-c-devel kmod-devel libudev-devel
- sudo make headers_install INSTALL_HDR_PATH=/usr
- ./autogen.sh
- ./configure CFLAGS='-g -O2' --prefix=/usr --sysconfdir=/etc --libdir=/usr/lib64
- make
- make check
- sudo make install

2.2 Accel-config presetting

doc for accel-config: https://github.com/intel/idxd-config/tree/master/Documentation/accfg

- accel-config config-engine iax1/engine1.0 -g 0
- accel-config config-wq iax1/wq1.0 -g 0 -s 16 -p 10 -b 1 -t 15 -m shared -y user -n app1
- accel-config enable-device iax1
- accel-config enable-wq iax1/wq1.0

Those commands only enable one IAX device(iax1) and one work queue(wq1.0) on one engine(engine1.0).

If you want all 8 iax devices enabled with 8 engines and 8 work queues, you need to do this in the shell:

```
for ((i=1;$i<=15;i=$i+2))
do
        echo $i

        accel-config config-engine iax$i/engine$i.0 -g 0
        accel-config config-engine iax$i/engine$i.1 -g 0
        accel-config config-engine iax$i/engine$i.2 -g 0
        accel-config config-engine iax$i/engine$i.3 -g 0
        accel-config config-engine iax$i/engine$i.4 -g 0
        accel-config config-engine iax$i/engine$i.5 -g 0
        accel-config config-engine iax$i/engine$i.6 -g 0
        accel-config config-engine iax$i/engine$i.7 -g 0


        accel-config config-wq iax$i/wq$i.0 -g 0 -s 16 -p 10 -b 1 -t 15 -m shared -y
user -n app1
        accel-config config-wq iax$i/wq$i.1 -g 0 -s 16 -p 10 -b 1 -t 15 -m shared -y
user -n app1
        accel-config config-wq iax$i/wq$i.2 -g 0 -s 16 -p 10 -b 1 -t 15 -m shared -y
user -n app1
        accel-config config-wq iax$i/wq$i.3 -g 0 -s 16 -p 10 -b 1 -t 15 -m shared -y
user -n app1
        accel-config config-wq iax$i/wq$i.4 -g 0 -s 16 -p 10 -b 1 -t 15 -m shared -y
user -n app1
        accel-config config-wq iax$i/wq$i.5 -g 0 -s 16 -p 10 -b 1 -t 15 -m shared -y
user -n app1
        accel-config config-wq iax$i/wq$i.6 -g 0 -s 16 -p 10 -b 1 -t 15 -m shared -y
user -n app1
        accel-config config-wq iax$i/wq$i.7 -g 0 -s 16 -p 10 -b 1 -t 15 -m shared -y
user -n app1

        accel-config enable-device iax$i

        accel-config enable-wq iax$i/wq$i.0
        accel-config enable-wq iax$i/wq$i.1
        accel-config enable-wq iax$i/wq$i.2
        accel-config enable-wq iax$i/wq$i.3
        accel-config enable-wq iax$i/wq$i.4
        accel-config enable-wq iax$i/wq$i.5
        accel-config enable-wq iax$i/wq$i.6
        accel-config enable-wq iax$i/wq$i.7
done
```

Check if the IAX devices are enabled by:

- accel-config list

3. QPL build and test

Register the access through：

https://registrationcenter.intel.com/en/forms/?productid=3389&pass=yes

Password: accept

Please be note, QPL is in alpha phase and not publicly release, persons in below list will get approved to access QPL.

```
ryanxgao@tencent.com ruanxgao
frankpang@tencent.com frank
felixtwang@tencent.com felix
hardyffu@tencent.com hardy
ivanren@tencent.com ivanren
ruippan@tencent.com ruip
```

3.1 Code modification

The test code is in ***qpl_library/examples/high-level-api/simple-operations***. Find the function you want to test and replace ***software*** as ***hardware*** in the code.

Take ***compression_example.cpp*** as an example, your code after modification should be like this:

```
 * were provided to you ("License"). Unless the License provides otherwise,
 * you may not use, modify, copy, publish, distribute, disclose or transmit
 * this software or the related documents without Intel's prior written
 * permission.
 *
 * This software and the related documents are provided as is, with no
 * express or implied warranties, other than those that are expressly
 * stated in the License.
 *
 */

//* [QPL_HIGH_LEVEL_COMPRESSION_EXAMPLE] */

#include <iostream>
```

```
#include <string>

#include <operations/compression/deflate_operation.hpp>
#include <operations/compression/inflate_operation.hpp>

auto main() -> int {
   // Source and output containers
   std::vector<uint8_t> source(1000, 5);
   std::vector<uint8_t> destination(500, 4);
   std::vector<uint8_t> reference(1000, 7);

   // Performing an operations
   auto deflate_operation = qpl::deflate_operation::builder()
         .compression_level(qpl::compression_levels::default_level)
         .compression_mode<qpl::compression_modes::dynamic_mode>()
         .gzip_mode(false)
         .build();
   auto inflate_operation = qpl::inflate_operation();

   const auto compressed_result =
qpl::execute<qpl::hardware>(deflate_operation,
                                                std::begin(source),
                                                std::end(source),
                                                std::begin(destination),
                                                std::end(destination));

   // Handle compression result
   compressed_result.handle([](uint32_t value) -> void {
                        std::cout << "Compressed size: " << value << std::endl;
                     },
                     [](uint32_t status_code) -> void {
                        throw std::runtime_error("Error: Status code - " +
std::to_string(status_code));
                     });

   const auto decompressed_result =
qpl::execute<qpl::hardware>(inflate_operation, destination, reference);

   // Handle decompression result
   decompressed_result.handle([&source, &reference](uint32_t
decompressed_size) -> void {
                           // Check if everything was alright
                           if (decompressed_size != source.size()) {
```

```
                                    throw std::runtime_error("Decompressed buffer size is
        not the same "
                                                    "as source size.");
                                }

                                // Compare source and decompressed buffers
                                for (size_t i = 0; i < source.size(); i++) {
                                    if (source[i] != reference[i]) {
                                        throw std::runtime_error("Content wasn't successfully
        compressed "
                                                        "and decompressed.");
                                    }
                                }
                            },
                            [](uint32_t status_code) -> void {
                                throw std::runtime_error("Error: Status code - " +
        std::to_string(status_code));
                            });

            std::cout << "Content was successfully compressed and decompressed." <<
        std::endl;

            return 0;
        }

//* [QPL_HIGH_LEVEL_COMPRESSION_EXAMPLE] */
```

Notice that ***qpl::software*** have been replaced by ***qpl::hardware .***

3.2 **Build and test**

- mkdir build
- cd build
- cmake -DCMAKE_INSTALL_PREFIX=/usr/local ..
- cmake --build . --target install
- cd examples/high-level-api

Now you can run the test you want. The result should be like this:

```
[root@spr02 high-level-api]# ./compression_example
Compressed size: 33
Content was successfully compressed and decompressed.
```