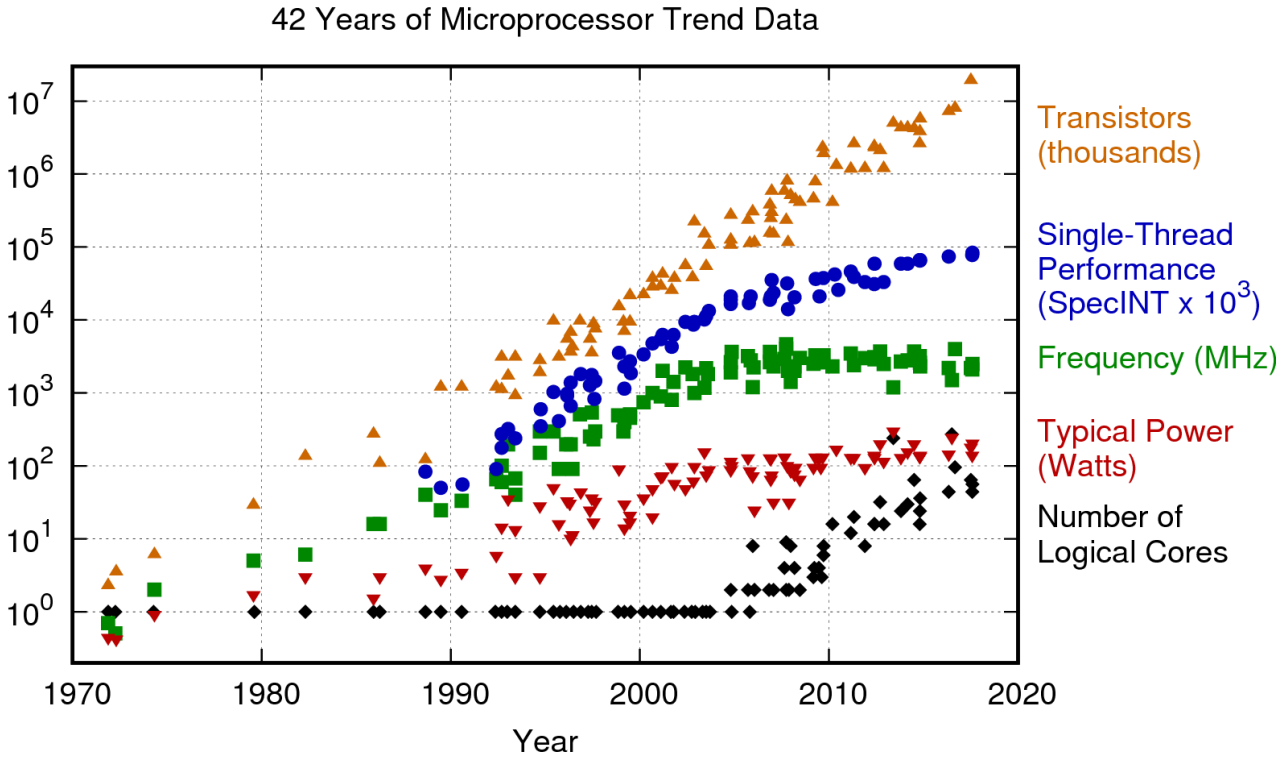
DEPLOY VTA ON INTEL FPGA

HARMAN
A SAMSUNG COMPANY

LIANGFU CHEN

11/16/2019

Moore's Law is Slowing Down

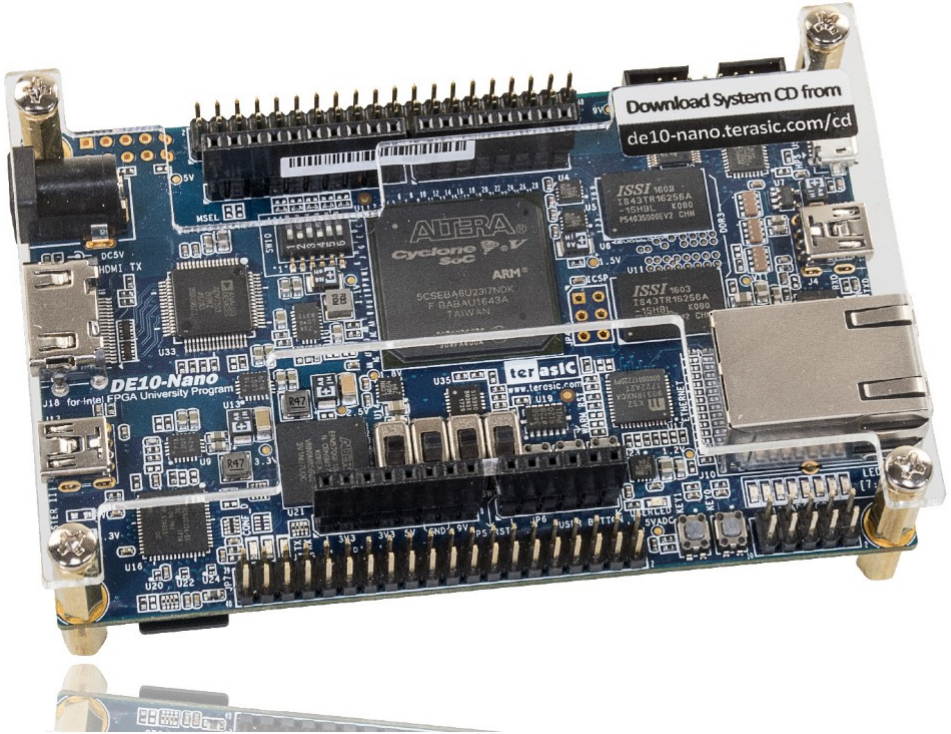


Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten
New plot and data collected for 2010-2017 by K. Rupp

Multi-Vendor Support

DEPLOY VTA ON INTEL FPGA

Terasic DE10-Nano



(Currency: USD)

Price: $130

Academic: $110

Buy it now

Software - CMA

Contiguous Memory Allocation – Linux Kernel

cma_alloc(*length, cacheable=0, data_type='void'*) [source]

Parameters:
- **length** (*int*) – Length of the allocated buffer. Default unit is bytes.
- **cacheable** (*int*) – Indicating whether or not the memory buffer is cacheable.
- **data_type** (*str*) – CData type of the allocated buffer. Should be a valid C-Type.

Returns: An CFFI object which can be accessed similar to arrays.

Return type: cffi.FFI.CData

1. Total size of buffer is automatically calculated as size = length * sizeof(data_type)
2. This buffer is allocated inside the kernel space using xlnk driver. The maximum allocatable memory is defined at kernel build time using the CMA memory parameters. For Pynq-Z1 kernel it is specified as 128MB.

The unit of *length* depends upon the *data_type* argument.

https://pynq.readthedocs.io/en/v2.0/pynq_package/pynq.xlnk.html

```
root@de10-nano:~# cat /proc/meminfo | grep -i cma
CmaTotal:          16384 kB
CmaFree:           15972 kB
```

Download Linux kernel from
https://github.com/altera-opensource/linux-socfpga/archive/rel_socfpga-4.9.76-ltsi-rt_18.08.02_pr.tar.gz

Configure Linux kernel

```
#
# Default contiguous memory area size:
#
CONFIG_CMA_SIZE_MBYTES=128
CONFIG_CMA_SIZE_SEL_MBYTES=y
# CONFIG_CMA_SIZE_SEL_PERCENTAGE is not set
# CONFIG_CMA_SIZE_SEL_MIN is not set
# CONFIG_CMA_SIZE_SEL_MAX is not set
CONFIG_CMA_ALIGNMENT=8
```

DEPLOY VTA ON INTEL FPGA

Software - CMA

Contiguous Memory Allocation – Linux Kernel Module

Setup Environment Variables

```
export PATH=/usr/local/intelFPGA_lite/18.1/embedded/ds-5/sw/gcc/bin:$PATH
export LD_LIBRARY_PATH=/usr/local/intelFPGA_lite/18.1/embedded/ds-5/sw/gcc/arm-linux-gnueabihf/lib:$LD_LIBRARY_PATH
export ARCH=arm
export CROSS_COMPILE=/usr/local/intelFPGA_lite/18.1/embedded/ds-5/sw/gcc/bin/arm-linux-gnueabihf-
export CC=${CROSS_COMPILE}gcc
export CXX=${CROSS_COMPILE}g++
export SYSROOT="/usr/local/intelFPGA_lite/18.1/embedded/ds-5/sw/gcc/arm-linux-gnueabihf"
```

Navigate to 3rdparty/cma and build kernel module

```
cd {VTAROOT}/3rdparty/cma
# configure KSOURCE_DIR in settings.mk
vi settings.mk
make
```

Copy kernel module to DE10-Nano and Install Module

```
root@de10-nano:~# insmod cma.ko
[   22.175781] cma: loading out-of-tree module taints kernel.
[   22.181858] CMA_INFO: Initializeing Contigous Memory Allocator module
```

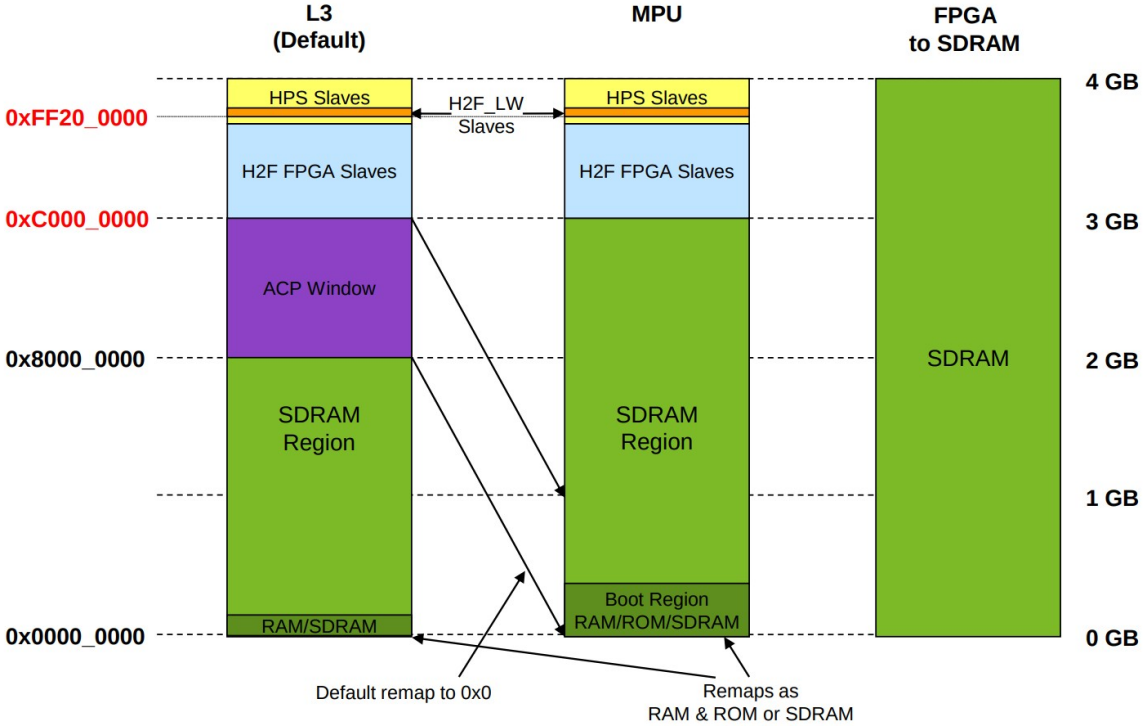
CMA API Reference

int	**cma_init** (void) Initialize CMA api (basically perform open() syscall). More...
int	**cma_release** (void) Release CMA api (basically perform close() syscall). More...
void *	**cma_alloc_cached** (size_t size) Allocate cached, physically contigous memory. More...
void *	**cma_alloc_noncached** (size_t size) Allocate noncached, physically contigous memory. More...
int	**cma_free** (void *mem) Release physically contigous memory. More...
unsigned	**cma_get_phy_addr** (void *mem) Get physical memory of cma memory block (should be used for DMA). More...

DEPLOY VTA ON INTEL FPGA

Software - Driver

Cyclone V & Arria V SoC HPS Physical Memory Map




```
vta_phy_addr_t VTAMemGetPhyAddr(void* buf) {
  return cma_get_phy_addr(buf) + 0x80000000;
}

class VTADevice {
 public:
  VTADevice() {
    // VTA stage handles
    vta_host_handle_ = VTAMapRegister(VTA_HOST_ADDR);
  }
  ~VTADevice() {
    // Close VTA stage handle
    VTAUnmapRegister(vta_host_handle_);
  }
  int Run(vta_phy_addr_t insn_phy_addr,
          uint32_t insn_count,
          uint32_t wait_cycles) {
    VTAWriteMappedReg(vta_host_handle_, 0x04, 0);
    VTAWriteMappedReg(vta_host_handle_, 0x08, insn_count);
    VTAWriteMappedReg(vta_host_handle_, 0x0c, insn_phy_addr);
    VTAWriteMappedReg(vta_host_handle_, 0x0, VTA_START);

    // Loop until the VTA is done
    unsigned t, flag = 0;
    for (t = 0; t < wait_cycles; ++t) {
      flag = VTAReadMappedReg(vta_host_handle_, 0x00);
      flag &= 0x2;
      if (flag == 0x2) break;
      std::this_thread::yield();
    }
    // Report error if timeout
    return t < wait_cycles ? 0 : 1;
  }
 private:
  // VTA handles (register maps)
  void* vta_host_handle_{nullptr};
};
```

Hardware

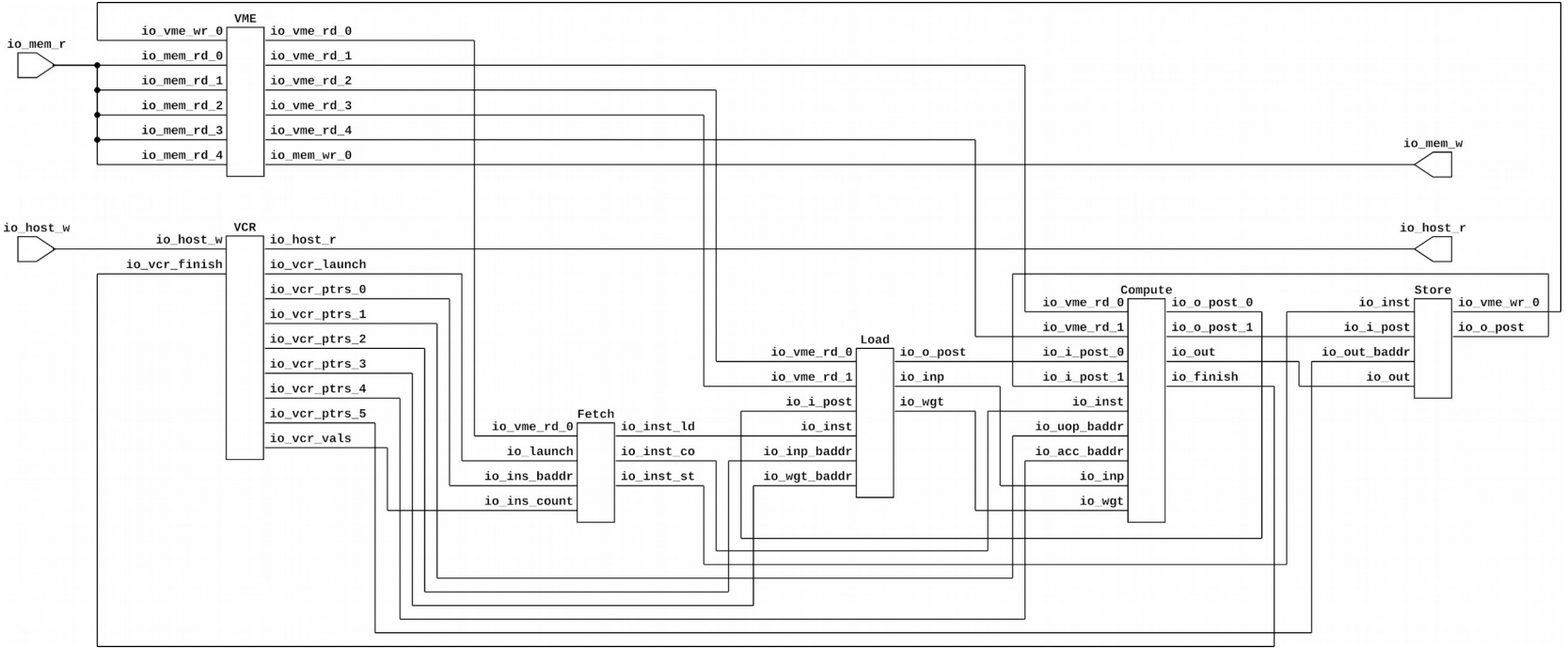
Configure Chisel VTA for DE10-Nano

```
/** PynqConfig. Shell configuration for Pynq */
class PynqConfig
    extends Config((site, here, up) => {
      case ShellKey =>
        ShellParams(
          hostParams = AXIParams(coherent = false,
                                 addrBits = 16,
                                 dataBits = 32,
                                 lenBits = 8,
                                 userBits = 1),
          memParams = AXIParams(coherent = true,
                                addrBits = 32,
                                dataBits = 64,
                                lenBits = 8,
                                userBits = 1),
          vcrParams = VCRParams(),
          vmeParams = VMEParams()
        )
    })
```

```
/** De10Config. Shell configuration for De10 */
class De10Config
    extends Config((site, here, up) => {
      case ShellKey =>
        ShellParams(
          hostParams =
            AXIParams(addrBits = 16,
                      dataBits = 32,
                      idBits = 13,
                      lenBits = 4),
          memParams = AXIParams(coherent = true,
                                addrBits = 32,
                                dataBits = 64,
                                lenBits = 4, // limit to 16 beats
                                userBits = 5),
          vcrParams = VCRParams(),
          vmeParams = VMEParams()
        )
    })
```

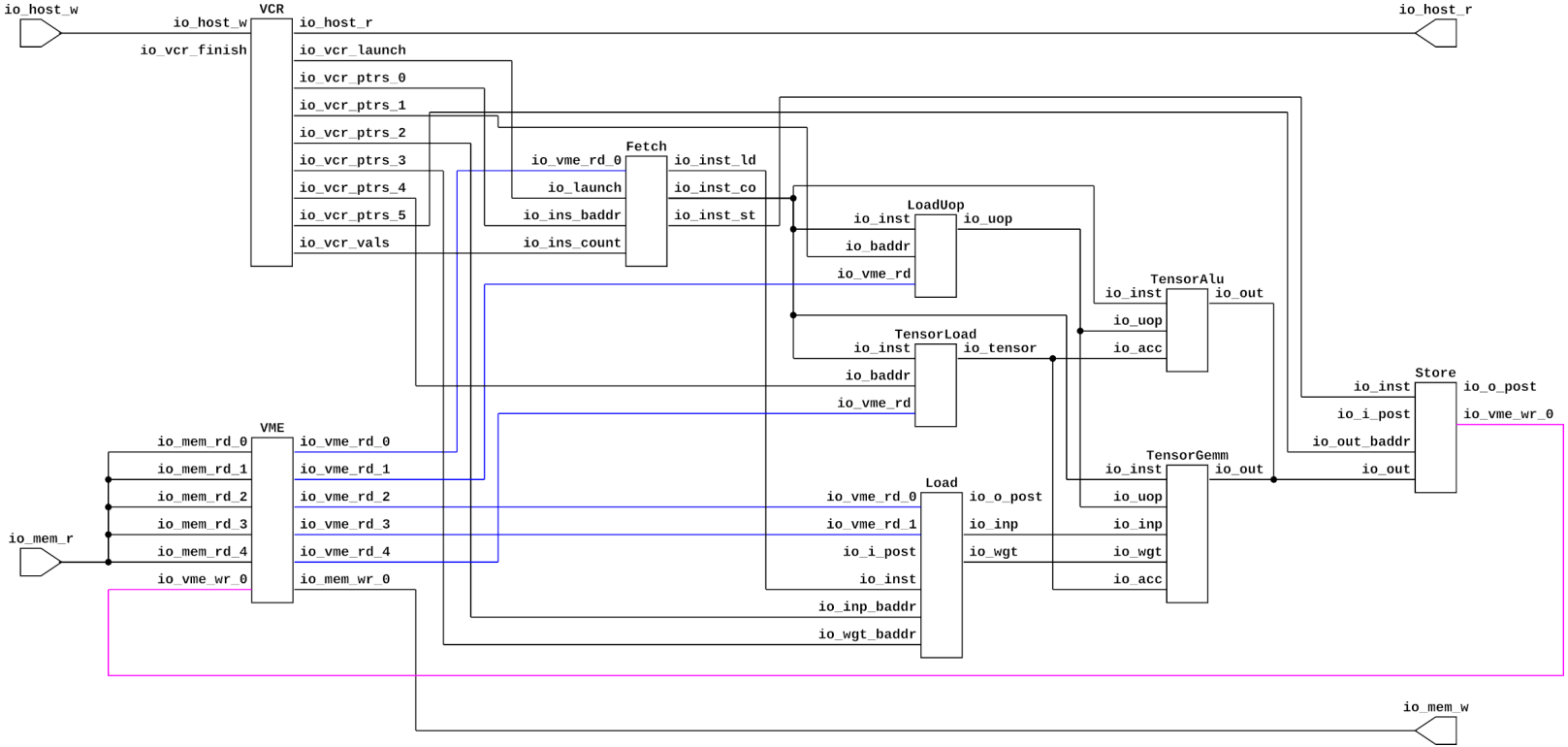
Hardware

Datapath of Chisel VTA



DEPLOY VTA ON INTEL FPGA

Hardware



Getting Started

Step 1: Get DE10-Nano and download & install Quartus Prime 18.1 Lite Edition

Step 2: Download SDCard Image from Terasic (Require Registration)

Step 3: Get files from https://github.com/liangfu/de10-nano-supplement

Step 4: Extract the files

- Step 4.1: Replace the zImage in SDCard Image
- Step 4.2: Extract rootfs_supplement.tgz to rootfs to install Python3
- Step 4.3: Copy cma.ko to home directory

Step 5: Cross compile TVM with USE_VTA_FPGA flag ON

Step 6: Copy the compiled TVM to the SDCard

Step 7: Install kernel module cma.ko and run apps/vta_rpc/start_rpc_server.sh

Step 8: Configure vta/config/de10nano_config.json to vta_config.json

Step 9: Go to vta/hardware/intel and run make command

Step 10: Get the generated .sof file programmed into hardware

Step 11: Evaluate the unit test script test_vta_insn.py with python3. Hooray!

THANK YOU

LIANGFU.CHEN@HARMAN.COM