EUROPEAN WORKSHOP ON ON-BOARD DATA PROCESING (OBDP2021), 14-17 JUNE 2021

DATA PROCESSING IN SPACE – THE HISAOR CHIP

Barry Kavanagh[(1)], Michael Ryan[(2)]

[(1)]O.C.E. Technology Ltd., NovaUCD, Dublin, Ireland, Email:barry.kavanagh@ocetechnology.com
[(2)]O.C.E. Technology Ltd., NovaUCD, Dublin, Ireland, Email:michael.ryan@ocetechnology.com

ABSTRACT

The design of a device for data processing in space must provide good radiation tolerance, low power consumption, the ability to interface with a range of different data sources, and high-speed signal and neural network processing. It should also be compatible with the main software development platforms, including those used in artificial intelligence. The design of the new *hisaor* system-on-chip from O.C.E. Technology balances these requirements in a way that offers an effective solution across a wide range of potential applications, including those that use neural network processing. This paper describes and explains the principal choices made in arriving at the *hisaor* design and illustrates its use.

1. Overview

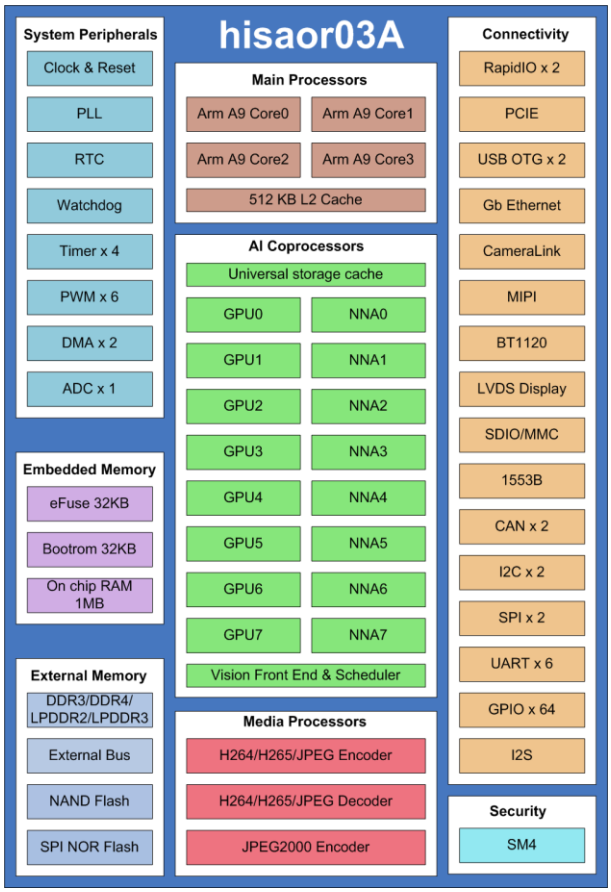

Figure 1: *hisaor* main subsystems

All design involves trade-offs, and the *hisaor* design is no exception. Requirements have to be prioritised and direct support for those of lower priority may have to be omitted.

A high priority for *hisaor* was image processing performance. This involves not only processing the image but also obtaining it in an appropriate form. While attention tends to be focussed on the processing stage obtaining the image is not trivial, so an early design decision was to include media processors and associated interface connections in the *hisaor* design.

The image processing itself requires both graphical and neural network processing and an Artificial Intelligence Unit to provide this was a high priority. This was implemented as 8 graphical processing units and 8 neural network units sharing a cache and scheduler. As well as image processing this provides high throughput floating point and integer calculations.

Other processing is needed, including for overall control of *hisaor*. Here the design choice was a Main Processors Unit with 4 ARM Cortex-A9 [1] cores with the ARM SIMD NEON extensions and the Coresight monitoring and debugging facility.

With those choices made a number of other important design trade-offs still needed to be considered, in particular in relation to data transfer and internal vs. external memory.

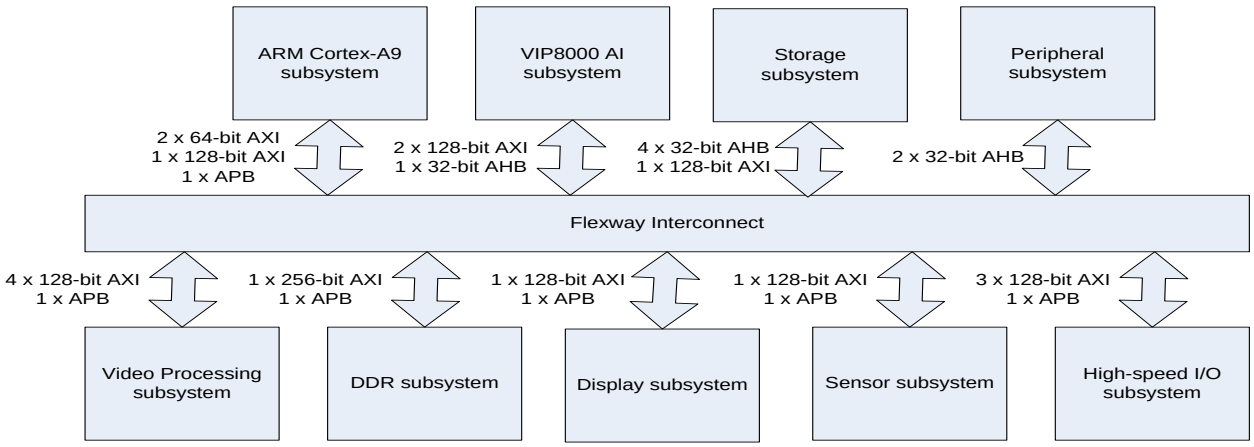

Figure 2: *hisaor* internal bus architecture

It was clear from an early stage that rapid transfer of data between the different subsystems would be critical to overall performance. As can be seen from Fig. 2 the internal bus architecture arrived at for *hisaor* was one of the most complex aspects of its design.

This bus design consists of:

- An Arteris network-on-chip (NoC) [2] used as the central interconnect fabric.
- Multiple DW_axi used as a subsystem AXI interconnect fabric.
- Multiple DW_ahb used as a subsystem AHB interconnect fabric.
- Multiple DW_apb used for low bandwidth peripherals and control of system devices.

It was important to have high bandwidth for the external DDR memory as on-chip memory is limited to the cache memories and 1 MiB of SRAM. This internal memory could have been increased by not including some subsystems on the chip, but after analysis the design decision came down in favour of a sophisticated high-speed bus with local caches on the main processing units and 1 MiB on-chip SRAM and with a high-performance interface to external DDR memory,

For this main DDR memory the *hisaor* choice was an Enhanced Universal DDR Memory Controller (uMCTL2) combined with DDR PHY to implement a memory interface to DRAM. The features include:

- Compatible with JEDEC standard DDR4/DDR3L/LPDDR3 SDRAM
- Data rates up to 2666Mbps(1333MHz) for DDR4/DDR3L/LPDDR3
- Support for single rank (chip select)
- Support for up to 3GiB DDR memory
- Support for up to 64-bit data bus width
- Support for an additional 8 bus lines for ECC
- Single host port with 256bits AXI4 bus interface for system access. The AXI bus clock is asynchronous with the DDR clock
- Advanced command reordering and scheduling to maximize bus utilization
- Low power modes, including power-down and self-refresh for DDR4/DDR3L/LPDDR3 DRAM; clock stop and deep power-down for LPDDR3 DRAM.

Minimising power consumption was seen as critical in the design, with low power modes on all IP cores.

Again critical is radiation tolerance. The basic silicon-on-insulator implementation of the chip gives excellent SEL protection, other aspects are referred to below.

Described below also are some of the other subsystems evident in Fig.1, with greater details given on the main subsystems mentioned above.

But what about software?

Software availability was a major consideration in the selection of the IP cores licenced from different vendors for use in *hisaor*. For example drivers for use with OpenVX under Linux on the main ARM processors were required where appropriate.
In many ways this was as important a decision as any of the hardware design decisions, and the resulting ease of use has enabled us to try many applications in the limited time since manufacture of the first evaluation board.

2. Acquiring the image:

To provide maximum flexibility and throughput in dealing with different image standards three video processing IP cores were selected, all from VeriSilicon, the VC8000E encoder [3], VC8000D decoder, and JPEG2K-E encoder.

The VC8000E is fully compliant with the ISO/IEC 15444-1 JPEG 2000 standard, supports up to 8Kx8K image or tile resolution and can sustain a very high, over 1080p30, throughput.

The VC8000D supports video decoding up to 4K@30fps. It supports HEVC, H.264, SVC, MVC and JPEG decoding, and reference frame compression (RFC).

The JPEG2K-E core is a video and still image encoder that implements Part 1 of the JPEG 2000 lossy and lossless image compression standard. It is fully compliant with the ISO/IEC 15444-1 JPEG 2000 standard and can sustain a 1080p30 throughput.

(Full details of these IP cores can be obtained from VeriSilicon)

To connect the external images source the sensor subsystem includes Camera Link, MIPI-CSI2, and BT1120 units.

Camera Link is a serial communication protocol standard designed for the purpose of standardizing scientific and industrial video products including cameras, cables and frame grabbers. The MIPI CSI-2 Host Controller is used for the reception of data from a CSI-2 compliant camera sensor. The BT1120 controller module provides an interface complaint with HDTV digital video interface standard BT1120.

The combination of these industrial standard connections with the powerful media processors makes it relatively straightforward to capture an image and convert it to the format required for further image processing.

3. Image Processing:

Image processing is computationally intensive, typically involving neural network processing and huge numbers of calculations. A core objective of the *hisaor* design was to achieve a high throughput for such calculations, and this led lead to the adoption of the approach taken in the central AI Coprocessors subsystem.

This subsystem's main features are:

- Eight Neural Network Engine (NN) and eight Parallel Processing Unit (PPU)
- OpenVX 1.1/1.2 compliance, including extensions
- Convolutional Neural Network acceleration
- Ultra-threaded parallel processing units
- Low main CPU loading
- MMU functionality supported
- Performance Counters for DMA Profiling
- Data transfers between Neural Network Engine and Parallel Processing Units with SRAM as local storage
- Neural Network Engine and Parallel Processing Unit synchronization with hardware semaphores

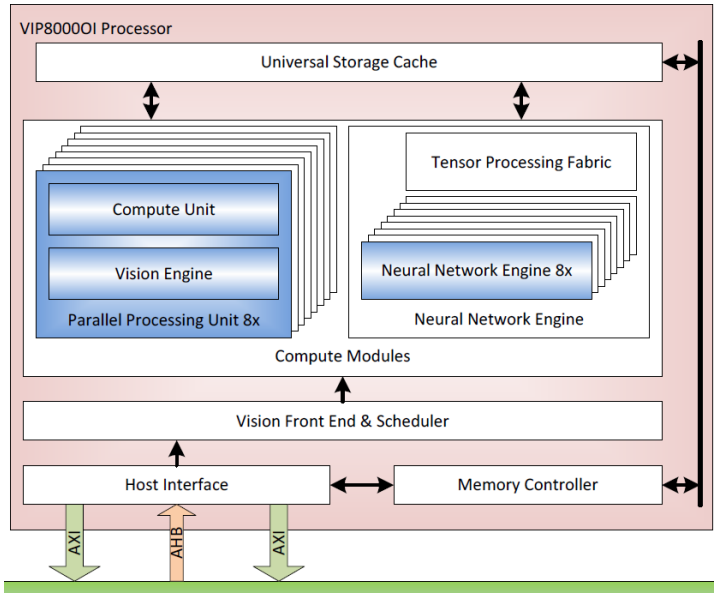

Figure 3: *hisaor* AI Coprocessors

At 1 GHz clock speed the highly parallel AI Coprocessors are theoretically capable of providing 64 Gflops and 12 TOPs. ([4] VIP8000).

4. Central control

A main processing unit is needed to take responsibility for allocating work to the AI Coprocessors and to carry out other tasks. This unit will typically run an operating system such as Linux.

The design choice here was an ARM Quadcore Cortex-A9 MPCore system that delivers high performance and provides good power management, and debug capabilities.

Each Cortex-A9 MPCore has the NEON vector processing and VFPv3 extensions, 32KiB of Level1 instruction cache, 32KiB of Level1 data cache, program trace capability (PTM), and separate power management domains. A 512 KiB Level2 cache is shared by the four cores with cache consistency management. The unit also includes the standard ARM Coresight to support monitoring and debug, an interrupt controller (GIC), and a clock and a reset manager.

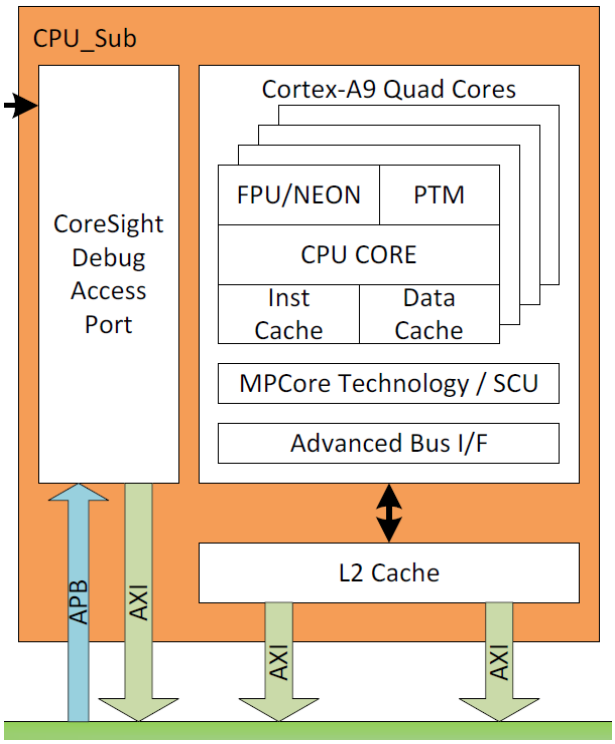

Figure 4: *hisaor* Main Processors

In addition to the high speed external DDR system memory the main processors also use a 64KiB on-chip ARM CA9 Boot ROM and 1MiB of on-chip SRAM. This SRAM is connected to the central network fabric by a 128-bit AXI bus width and can be accessed by all the AXI masters on the chip. A typical use is as a high bandwidth shared buffer.

The interrupt controller complies with ARM Generic Interrupt Controller Architecture Specification 1.0. It is a single unit responsible for centralizing all interrupt sources before dispatching them to an individual Cortex-A9. Its control registers are memory-mapped.

5. Clocks

hisaor has five types of clock sources:
1) 24MHz used as the primary clock source for the PLLs and to generated the clock for CPU, VIP, VPU, DDR, BUS, high-speed interfaces, etc. For all PLLs, the 24MHz clock can be used as the PLL reference clock directly.
2) 32KHz used by the real time clock (RTC).
3) 62.5MHz used as Rapid IO reference clock.
4) 100MHz used as PCIe reference clock.
5) JTAG TCK clock input from debugger.

In order to meet the clock requirement for the different IP blocks, *hisaor* has a number of PLLs:

- PLL_CPU0, used to generate the clock for the main processors unit with its four Cortex-A9 cores.
- PLL_VIP, used to generate the clock for GPU.
- PLL_VPU, used to generate the clock for VPU.
- PLL_SOC: used to generate the clock for SOC, including BUS, HSIO, peripherals, etc.
- PLL_DDR: used to generate the clock for DRAM PHY and controller.
- PLL_DISP: used to generate the clock for video output display interface.
- PLL_SENS: used to generate the clock for video input sensor interface.
- PLL_HSIO: used to generate the clock for high speed interface.
- PLL_AUD, used to generate the clock for audio interfaces.

Two PLLs are located to minimise jitter:

- PLL_CPU0 is put close to the quad-core main processors unit to minimize jitter.
- PLL_DDR is put close to the DDR PHY to make it easier to meet the jitter requirement of the JEDEC standard.

Two different types of PLLs were used:

- PLL_CPU0, PLL_VIP, PLL_VPU, PLL_SOC, PLL_SENS and PLL_HSIO are integer PLLs, configured at different frequencies.
- PLL_DDR, PLL_DISP, PLL_AUD are fractional PLLs and can be configured to the very accurate frequencies required by audio and video interfaces.

Each PLL has its own dedicated configuration register, which can be accessed through the APB bus, allowing software to enable/disable the PLL or change its clock frequency. All the fractional PLL can support on-the-fly frequency adjustment in small steps without glitches on the clock output.

6. Power:

One of the main criteria for selecting IP cores for use in the *hisaor* was their efficiency in the use of electric power when active and whether they could in effect be turned off when not in use.

The different units on *hisaor* result in a need for 5 different power supply voltages, typically supplied by an external power management integrated circuit (PMIC). These voltages fall under the following main headings:

- VDD_CPU for the main processors Cortex-A9s
- VDD_VIP is for the media processors.
- VDD_DDR for the DDR controller & PHY.
- VDD_RTC for the always-on RTC.
- VDD_SOC for the other *hisaor* modules.

Keeping these as clean as possible led to a number of decisions with regard to the ways in which they were connected:

1) The GPIO pads have external power supplies for 3.3V and 1.8V IO voltages. The IO pad core voltage are supplied directly by VDD_SOC (except for RTC IO).
2) The DDR controller & PHY have two external power supplies, VDD_DDR for the controller & PHY, and VDDIO_DDR for IO.
3) For all the integrated analogue modules, their 1.8V analogue power and 0.8V digital power are supplied through external power pads. These supplies are separated from other power pads on the package to help keep them clean, but they can be shared with other power rails on the board to reduce the number of power supplies from the PMIC.
4) For all the integrated PHYs, the PCIe PHY, MIPI PHY, Camera Link PHY, Rapid IO PHY, LVDS TX PHY, SDEMMC PHY and USB20 PHY, their 3.3V, 2.5V, 1.8V and 0.8V supplies are provided through external power pads. The supplies to those PHYs are separated from other power pads on the package to help keep them clean, but they can be shared with other power rails on the board to reduce the number of PMIC connections.
5) For the RTC, the 0.8V core logic supply and 3.3V analog/IO supply are supplied externally.
6) There are no integrated LDOs inside the chip (except for the Analog PHY internal LDO).

hisaor supports the following power modes:

- RUN Mode: In this mode, a Cortex-A9 CPU in the main processing unit is active and running.
- IDLE Mode: This mode can be entered automatically by a Cortex-A9 CPU when it has no thread ready to run. All high-speed devices are inactive, DRAM & bus clocks are reduced, most of the internal logic is clock gated but still remains powered. Compared to RUN mode, all the external power from the PMIC remains the same, and most of the IPs remain their state, so the interrupt response latency in this mode is very small.
- SUSPEND Mode: This mode provides the maximum power saving with all clocks off and all unnecessary power supplies off. The CA9 CPU platform is fully power gated, all internal digital logic & analogue circuits that can be powered down are off, all PHYs are power gated.
  The exit time from this mode is much longer than from IDLE Mode but the power consumption is also much lower.
- RTC Mode: In this mode, only the power for the RTC domain remains on so as to keep the RTC logic alive.
- OFF Mode: This mode has all power rails OFF.

The IDLE Mode and the SUSPEND Mode are the two main low power modes used with Linux.
Typical power consumption in run mode is 5W.

7. Radiation Tolerance

hisaor is based on Global Foundry's 22 nm semiconductor FDX process, a silicon on insulator technology which makes *hisaor* immune to single event latchups (SELs). The memory controller for the main external memory allows BCH error detection and correction for single bit errors, and error detection for all double bit errors.

We expect *hisaor* to demonstrate a high resistance to single event effects (SEEs) and an ability to tolerate a high total ionisation dose (TID) and at present are awaiting results for the corresponding tests.

8. Other features

A range of other IP Cores to provide I/O and Network connectivity were in included in the hisaor design, as can be seen from a glance at Fig.1.

These were sourced from a number of IP Core suppliers. Details of them can be found in the hisaor documentation, and as they are of fairly standard functionality we decided not to describe them any further here.

9. Physical Features

The package is HFCBGA-896, 25 x 25mm, with 0.8 mm pitch. This is a thermally enhanced FCBGA that can dissipate up to 8 watts when mounted using thermal interface materials.
Operating temperature range is -40°C to +125°C

Figure 5: *hisaor*

10. Development Hardware

The initial *hisaor* Evaluation Board is similar to most evaluation boards for system-on-chip devices, the major difference being the connectivity it provides for *hisaor*'s video inputs.

The board operates at 800MHz, provides 3 GiB of DDR4 RAM, and various connections including for CameraLink, MIPI, BT1120, LVDS Display, Giga Ethernet, USB2.0, SD card, 1M/10M 1553B bus Controller, CAN bus controller, UART, and GPIO. It typically boots into Linux from an SD card.

Figure 6: *hisaor* Evaluation Board

11. Software Development Environment

In testing and benchmarking so far Linux Ubuntu has been used as the operating system for the main ARM Cortex-A9 processors.

Development for the AI Coprocessors was greatly simplified by the availability of the Acuity IDE from VeriSilicon and by the work of the KHRONOS group in making OpenVX [5] and OpenCL available.

Using KHRONOS tools an A.I. model based on various frameworks could be converted to their neural network exchange format (NNEF), and then with the Acuity IDE quantized and optimized and an OpenVX graph created and optimized. The Acuity IDE could then produce a C programme with the appropriate OpenVX or OpenCL calls for compilation with *gcc* and downloading to *hisaor*.

It is fair to say that the overall procedure turned out to be more straightforward than we had expected, thanks in no small part to the work of VeriSilicon and the KHRONOS group.

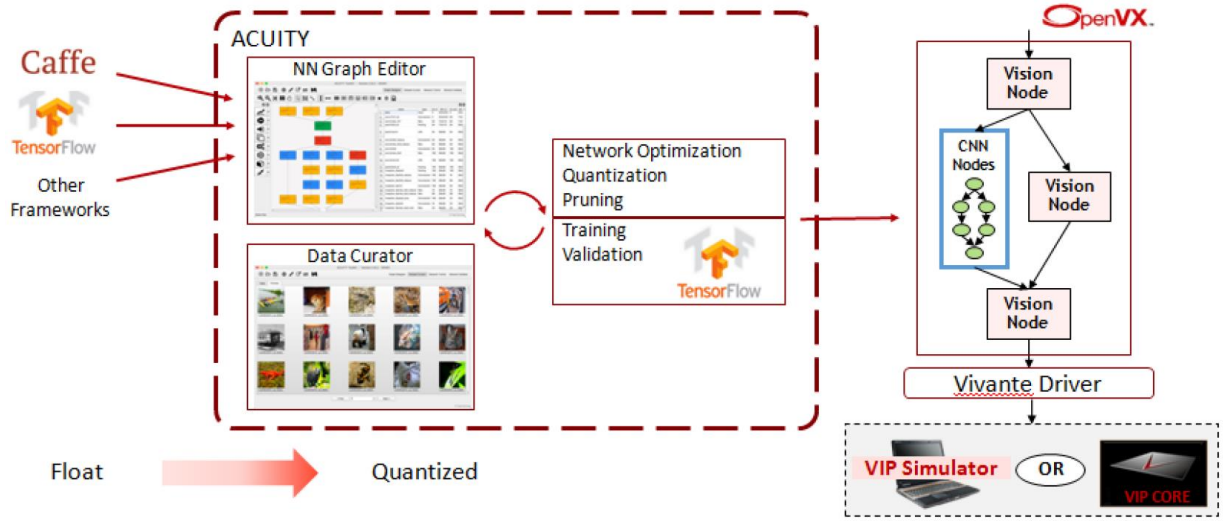

Figure7: An AI Model on *hisaor*

12. Performance Benchmarks

The AI coprocessors on *hisaor* offer a max performance at 1GHz of 64 GFLOPS for floating point and 12 TOPS for integer processing. In many cases the raw processing power is likely not to be the main limitation on throughput, which is more likely to be memory bandwidth.

Doing performance benchmarks however has turned out to be trickier than expected. A pre-release of the ESA OBPmark matrix multiplication benchmark gave the following results using an optimised version of OpenCL which uses only the GPUs.

Table 1: Matrix multiplication timings

Datatype	Matrix size	Host->Device (ms)	Kernel (ms)	Device->Host (ms)
32-bit Float	1024x1024	24	4	364
32-bit Float	2048x2048	93	17	3,063
32-bit Float	3072x3072	207	38	10,207
32-bit Float	4096x4096	385	68	24,268

The results have been verified as correct in each case, but clearly the timings above are wrong.

Apart from anything else, the scaling factor between the kernel times follows a square law quite well when a cube law (or something better if using Strassen [6]) would be expected. We need to investigate further both the benchmarking software and the way in which we are using it. Although all results were correct the above timings can only be regarded as an indication of a puzzle, one that unfortunately we have not had the opportunity to solve but will solve soon.

13. Demonstration

Two demonstrations of the Yolo model are presented. The first uses a live video feed from a USB camera connected to the USB port on *hisaor*. The application uses YoloV2 to detect objects in the live feed and classify them, showing the probability for the classification and the time taken in milliseconds to process the image from the video feed. The YoloV2 model is executed on the AI Coprocessors using both the OpenVX API and OpenCL. The diagram at Fig. 10 below shows the processing carried out by YoloV2. A video of this demonstration is available at [7].

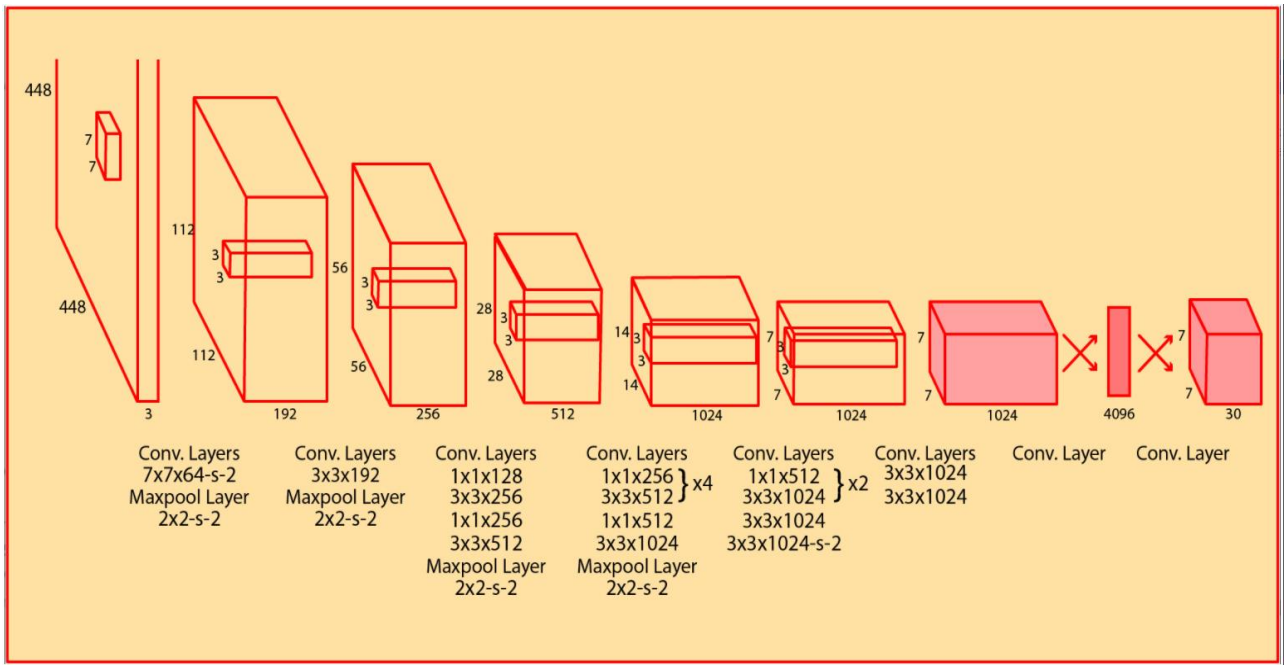

Figure 10: YoloV2 Processing Steps

The second demonstration uses YoloV3 to process a SAR image taken from a satellite and identifies three ships. The result is shown in Fig. 11. The processing time was 4 milliseconds.

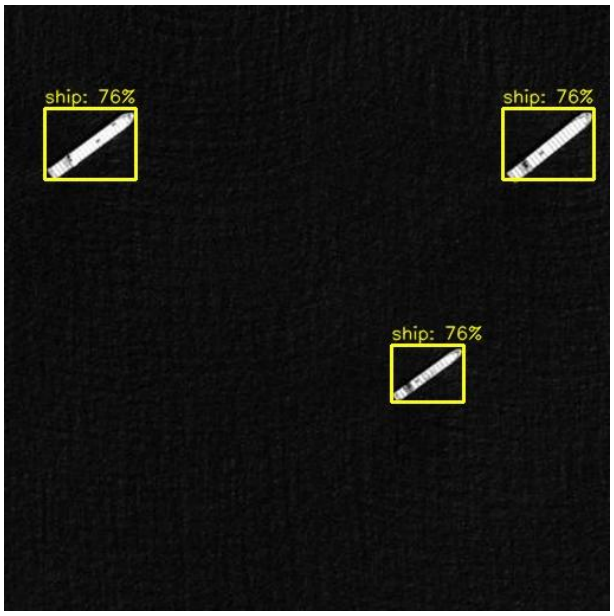

Figure 11: Result for Second Demonstration

With Linux Ubuntu running on *hisaor* setting up and running both demonstrations and the various other examples we tried turned out to be as experienced on other systems and quite straightforward.

14. Acknowledgements

It is a pleasure to acknowledge the work on this project of our partner Zhuhai Orbita Aerospace Science & Technology Co. Ltd. whose staff made a major contribution to both the design and implementation of *hisaor*. In particular we would like to thank Mr. Gong Yonghong for his unfailing help and courtesy.

We are also very grateful to VeriSilicon and to the KHRONOS group.

REFERENCES

[1] Cortex-A9 Reference Manual, ARM (2012) https://developer.arm.com/documentation/ddi0388/latest

[2] FlexNoC® Interconnect IP, Arteris IP, https://www.arteris.com/flexnoc

[3] Hantro VC8000, Verisilicon, https://www.verisilicon.com/en/IPPortfolio/HantroVC8000E

[4] VIP8000, Verisilicon, https://www.verisilicon.com/en/PressRelease/VivanteVIP8000

[5] The OpenVX™ Specification, Khronos®, Version 1.3 (2020) https://www.khronos.org/registry/OpenVX

[6] Strassen V., *Numer. Math.* **13,** 354–356 (1969). https://doi.org/10.1007/BF02165411

[7] Video of First Demonstration https://youtu.be/sPCiSRde4os