****

Kria KV260 familiarizing the board:

https://www.amd.com/en/products/system-on-modules/kria/k26/kv260-vision-starter-kit/getting-started-ubuntu/getting-started.html
Note: Above link is for Ubuntu.
https://www.amd.com/en/products/system-on-modules/kria/k26/kv260-vision-starter-kit/getting-started/getting-started.html
Note: Above link is for PetaLinux.
Note: In the above links, we can navigate further using the "Next Step" button.
https://github.com/Xilinx/smartcam
(the github link is anyway accessible by opening the software repository link)

Note that the board, module and chip are different. The Kria KV260 starter kit, is a board which have KV260 SOM (module) mounted, with K26 SOC inside. Basically, the board enables the I/O to the SOM. The board has many I/O connectors, circuitry for the I/O connectors.

****

Kria KV260 execution steps (when .xmodel file and testbench module is available):
Note: By “testbench module” it just refers to a module which can run the .xmodel on KV260.

The reference link is:
https://xilinx.github.io/Vitis-AI/3.0/html/index.html

Can ignore the initial part. Just skip to the page “Host Installation Instructions”.

Note: The link also explains about the steps involved in preparing a neural network for execution on the Kria KV260 board (converting a neural network model into .xmodel file which can be run on the Kria KV260 board. Steps can be summarized as below:
- Training the neural network.
- Freezing the inference graph.
- Test the accuracy of the frozen graph
- Perform calibration for quantization, quantization.
- Test the accuracy of the quantized graph.
- Compile the quantized model into a .xmodel file. The .xmodel file is targeted for a board.
Above steps are not required when we have the .xmodel file already available.

1) One time step: Host initial preparation.
For a CPU only host, there is no need for any preparation. In the future, check the possibility of making use of the AMD graphics card and installing the required driver and doing the necessary steps; the webpage includes the details of it. The understanding is that Docker with graphics card is useful if we do the quantization on the Docker (not sure if this is correct).
Note: Attempted installing the graphics driver for AMD graphics card, but was unsuccessful due to unmet dependencies (and not correctable!).

2) One time step: Docker install and verification.
The webpage gives a link which can be followed to install the docker. Open the link, select Ubuntu to get info on how to install Docker. There are four different ways in which Docker Engine can be installed. Select the second option “Set up and install Docker Engine from Docker’s apt repository”.
Note: The first option is to install Docker Desktop (Docker Engine comes with it). It is mentioned as the easiest way, but avoided this way as no idea why Docker Desktop is for. What is the difference between Docker Desktop and Docker Engine??

3) One time step: Clone the Vitis-AI repository from Github.
Open a terminal in the login user's home directory, and clone the Vitis-AI repository to this directory.
Note: Once the Vitis-AI repository has been cloned, the full path to the Vitis-AI repository will be /home/user/Vitis-AI. Don't confuse the "user" with "usr". Here "user" is the login user's name which is "user" itself.

4) One time step: Chose one of the pre-built Docker containers and pull it into the Docker Engine. Need to chose the appropriate container for CPU only processing with PyTorch support. This is a one time step, thereafter whenever we want to run this Docker container, we can execute docker_run.sh script present in the Vitis-AI folder. The command should be given as below:
./docker_run.sh xilinx/vitis-ai-<pytorch|opt-pytorch|tensorflow2|opt-tensorflow2|tensorflow>-<cpu|rocm>:latest
Errors:

Permission denied error when executing the below command:
docker pull xilinx/vitis-ai-pytorch-cpu:latest.
Solution:
sudo usermod -a -G docker $USER

5) One time step: **(skip this step for now, refer the observation below)** Install cross-compiler.
Follow the installation steps given in the webpage. The cross-compiler setup script is present in the Vitis-AI repository itself. Once cross-compiler has been installed, a folder named "petalinux_sdk_2022.2" will appear in the user's home folder.
Issues:

No "mpsoc" folder found in the Git repository.
The Vitis-AI folder was created using the command below:
git clone https://github.com/Xilinx/Vitis-AI
But the subfolder "board_setup" was missing "mpsoc" subfolder.
Solution: Go inside the git repository folder (Vitis-AI), then execute the below command:
git switch 3.0
This command will switch the git repository to the version 3.0. Note that when we do the git clone, only the latest version will be cloned (which was 3.5).

Important Observation: The process as per the documentation is to install the cross-compiler and source it before starting the Docker container. But, this way of doing it cause problem. If getting compilation error repeatedly, it can imply a corrupted Vitis-AI repository. As a solution, clone the Vitis-AI repository freshly. Don't install it on the host machine. Run the Docker container, activate the appropriate conda environment, then install the cross compiler, then execute the source command shown at the end of the installation logs. Do "unset LD_LIBRARY_PATH" ONLY if there is error stating "Your environment is misconfigured".

Note: Instead of cloning the Vitis-AI repository again, alternatively we can backup the Vitis-AI repository (create a copy of the /home/user/Vitis-AI folder), and restore it to start freshly.

Note: If installing the cross-compiler inside the Docker container, then we need to do it every time we invoke it. Need to see if there is a way to save the changes to the container image.

Commands to be executed after Step-9 (to be executed on the Docker container).
cd /workspace/board_setup/mpsoc
sudo chmod u+r+x host_cross_compiler_setup.sh
./host_cross_compiler_setup.sh

6) One time step: Setup the SD card image:

Follow the below steps:

Download the Vitis AI pre-built SD card image from the appropriate link given in the webpage.
Note: Will need to create a UserID and password by giving the details.
Note: The file name will be: “xilinx-kv260-dpu-v2022.2-v3.0.0.img”.
We can access the same link through a different webpage, given below:
https://docs.amd.com/r/en-US/ug1354-xilinx-ai-sdk/Setting-Up-the-Target
In the above link select "3.0 English". Download the image from the link given against "The pre-built image for the KV260 starter kit". Extract the downloaded img.gz file. Inside the extracted folder, we can find the file named "petalinux-sdimage.wic".
Is it PetaLinux that will be present in the image?? How to change it to Ubuntu?? It was a different webpage that was followed to setup the board. There was option to chose Ubuntu or PetaLinux. Need to confirm.

Use BalenaEtcher to burn the image file onto the SD card.
Need a card reader. Download Balena Etcher...extract the .zip file, then open "balena-etcher", choose "flash from file", select the "petalinux-sdimage.wic" file, then select target (once we connect card reader, it will be listed in the targets), then click flash.

Insert the imaged SD card into the target slot of KV260.

7) Source the cross-compiler environment setup script using the below command:
(skip this step for now, refer the observation under Step-5)
source ~/petalinux_sdk_2022.2/environment-setup-cortexa72-cortexa53-xilinx-linux
Issues:

If the command failed, run the below to enable Cross Compiler
unset LD_LIBRARY_PATH
source ~/petalinux_sdk_2022.2/environment-setup-cortexa72-cortexa53-xilinx-linux

8) Run the Docker container:
cd /home/user/Vitis-AI
./docker_run.sh xilinx/vitis-ai-pytorch-cpu:latest
Note: Notice that the /workspace directory in Docker corresponds to your /Vitis-AI directory on the host.

9) Inside the Docker container: Activate the vitis-ai-pytorch conda environment, using the below command.
conda activate vitis-ai-pytorch

Execute the skipped steps Step-5 followed by Step-7 here.

10) Inside the Docker container: Compile (cross-compile) the application.
cd examples/vai_runtime/resnet50_pt
bash –x build.sh
Note: Here in this example, resnet50_pt is a folder containing the application program modules (or driver module and other supporting modules??). The folder also contains build.sh script and Makefile. If we are writing our own application, we should create a package similar to this. The master module in the application program modules is the one to be invoked. Its the application that executes the deep learning model (.xmodel file).
Note: The build.sh will invoke the cross-compiler.

Errors:

Error when compiling the resnet50_pt example:
bash –x build.sh
If we copy the above command from the documentation webpage, we may get error, so just type the command.

11) Connect the board (power cable, LAN, serial port connection), power up, initial setup.

Connect serial port and start “terminal” application, and power up the board.
Note: We can use “gtkterm” as the terminal app over serial port. The data exchange is over the serial port connection using the UART hardware protocol.
Note: Instead of gtkterm, we can use SSH to get a terminal, its over LAN.
Note: If we start the terminal application before powering up the board, we can see the boot logs (just to know that the board is working as expected).

Note: We can install gtkterm using the command “sudo apt install gtkterm”. Open the gtkterm (sudo gtkterm) and do the configuration to select the right USB port (its /dev/ttyUSB1). GTKTerm is a graphical terminal emulator for Linux and other POSIX-compliant operating systems that allows users to communicate with devices that have a serial interface. Some examples of devices that can be communicated with using GTKTerm include microcontrollers, embedded computers, modems, GPS receivers, and CNC machines.

Note: The Kria Starter kit available at Netrasemi has an additional start button (and circuit) (for measuring current usage).

If DHCP is not present, then do network config.
Issues:

Issue connecting Kria board to LAN:
Connect PC and Board in a one to one connection using LAN cable. Executed the below command on the PC. Use any ip address from the Class A private ip address range 10.0. 0.0 to 10.255. 255.255..
sudo ifconfig enp4s0 10.0.0.10
Then execute the below command on board (via gtkterm serial connection login).
ifconfig eth0 10.0.0.20
We can use the ifconfig command to get the details of the network card/port. On the new systems the network port name can be enp4s0 instead of eth0.
To get internet on PC, use a “wifi dongle” to connect to wifi.
Network name: TP-Link_2C79

Setup SSH connection (this is not required as we have gtkterm)
[Host] $ ssh -X root@[TARGET_IP_ADDRESS]
[Host] $ ssh -X user@[HOST_IP_ADDRESS]
We can start the ssh shell either on host (using first command above) or on the target board (using the second command above). Note that “root” and “user” are the login user’s user names on the board and host machine respectively. Replace the ip addresses with the actual ip addresses of the target and host.

Setup DISPLAY
The ideal setup is to connect a monitor using HDMI cable. But is this is not possible, then export the display using the below command.
[Target] $ export DISPLAY=[HOST_IP_ADDRESS]:0.0

12) Copy the executable files, model file(s) if any (.xmodel) and data into the board.

Executable files: Copy the compiled (cross-compiled) executable files to the board.
Note: Where to find the executable files?? When we run the build.sh to compile the source files, executable files will be written to the same folder.

Model files (.xmodel files): If there is .xmodel file readily available in Vitis-AI Model Zoo, then check if the model is available on the microSD card image (check in directory "/usr/share/vitis_ai_library/models" on the board), if present we can use it straight away. If not present, then check if the .xmodel file is available in the Vitis-AI repository on the host machine. If present, we can copy it to the board. If not present, then check if the .xmodel file is present in the Vitis-AI model zoo online. If present, we can download the file to host machine and copy to the board, or directly download into board (if LAN connection available). If not present, then we may have to train a model (or use a pre-trained model), quantize, compile, etc, to create the .xmodel file.
Example:
[Host] $ cd /Vitis-AI
[Host] $ wget https://www.xilinx.com/bin/public/openDownload?filename=resnet50-zcu102_zcu104_kv260-r3.0.0.tar.gz -O resnet50-zcu102_zcu104_kv260-r3.0.0.tar.gz
[Host] $ scp resnet50-zcu102_zcu104_kv260-r3.0.0.tar.gz root@[TARGET_IP_ADDRESS]:~/
[Target] $ tar -xzvf resnet50-zcu102_zcu104_kv260-r3.0.0.tar.gz
[Target] $ cp resnet50 /usr/share/vitis_ai_library/models -r

Data files: Download and copy the data required for running the application, onto the board.

Errors:

scp error: No such file or directory. The error message is as below (2 lines):
sh: line 1: /usr/libexec/sftp-server: No such file or directory
scp: Connection closed

This issue is only on the host machine. We can execute the scp on target machine (the Kria board), via the gtkterm terminal. Remember that scp can work both in push and pull mode, so we can copy from host to target as well as target to host, by executing scp on the target. The order of files is same (source followed by destination) for both modes. The path to the file on remote system (in this case its the host machine), should be prefixed with "login-user-id@remote-ipaddress:".

13) Run the model
Run the master application module.

Other issues faced:
1) Ubuntu non stable version issue: instructions to upgrade from 23.10 to Noble:
https://infotechys.com/upgrade-ubuntu-23-10-to-24-04/#elementor-toc__heading-anchor-3

2) While trying to run Balena Etcher, received error "no partition table, not bootable".
Was not using the correct image file. It appears that the image given in the below link is not useful.
https://xilinx-wiki.atlassian.net/wiki/spaces/A/pages/1641152513/Kria+SOMs+Starter+Kits#Starter-Kit-Pre-Built-Software

Additional Info:
Kria starter kit: Kria K26 is the chip (a production ready hardware platform).
Kria KV260 Vision AI Starter Kit (chip + board) is an evaluation and development platform for the K26 SOM.

Do firmware update (is this required??). firmware are pre-built into the starter kit. Where do the firmware reside?? In the microSD card??

Linux (Ubuntu) need to be flashed into a microSD card. Username: ubuntu, Password: ubuntu (Will be prompted to change on first login). Instead of Ubuntu, we can use Embedded Linux (image can be built using the Yocto or PetaLinux tools). The Embedded Linux built by PetaLinux tools, we generally mention as PetaLinux.

To run the smart camera demo app provided by Xilinx, follow the steps in the link given below.
https://www.amd.com/en/products/system-on-modules/kria/k26/kv260-vision-starter-kit/getting-started/trying-the-smart-camera-app.html
This is not tried yet!

Vitis-AI Repository (Model Zoo, example application, etc):

https://xilinx.github.io/Vitis-AI/3.5/html/docs/reference/ModelZoo_Github_web.htm

Note: This link is for Vitis-AI 3.5 version. This page contains just a list. The webpage with detailed models info is the one below:

https://github.com/Xilinx/Vitis-AI/tree/master/model_zoo

Note: Above link is for Vitis-AI 3.5 repository. But many of the models are available only in Vitis-AI 3.0 repository. To switch to 3.0 repository, click on the “github tree” button on the left side of the window and change “master” to “3.0”. The webpage link will be changed as below:

https://github.com/Xilinx/Vitis-AI/tree/3.0/model_zoo

Some example applications (driver modules to run the models in the model zoo) can be found in the below link (part of the Vitis-AI repository):

https://github.com/Xilinx/Vitis-AI/tree/3.0/examples

Use the command “git diff” to see the changes made to the git repository.
Use the command “git stash” to revert the changes.

Use the below command to install something on PetaLinux.
sudo dnf install canberra-gtk-module

To install make and other C++ compilation tools:
sudo apt install build-essential

****