NPU IP

Figure 1 shows the overview of the NPU deployed on Zynq platform. Basically, NPU works as an accelerator and it consists of two AXI ports including an AXI mater port and an AXI slave port. The AXI master port is used to access the instructions, weights and IO features stored in main memory while the AXI slave port is used to configure NPU and report NPU status to the PS. To minimize the communication between NPU and processors in PS, we have NPU configured with the addresses of the data to be used for neural network execution including instructions, input features, output features and weights first, Then NPU will automatically load the instructions and start the neural network execution according to the instruction until it completes the inference of an input feature map. For each inference, NPU will issue an interruption to notify the processors.

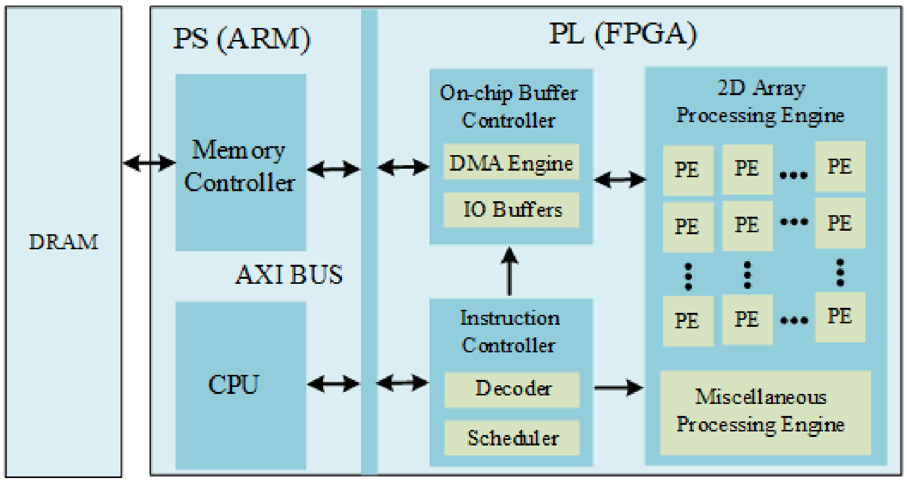

Figure 1 NPU deployed on Zynq

Inside NPU, it has three major modules including computing module, on-chip buffer module and instruction module. The computing module has a 2D computing array mainly for the most computing-intensive operations such as convolution and full connection. Meanwhile, it has a multi-functional processing engine for miscellaneous computing tasks such as sigmoid and Relu which is less computing-intensive. This module can reuse the computing results directly from the 2D computing array such that the computing module does not need to retrieve the data from the buffer nor DRAM again. Another critical component of NPU is the on-chip buffer module. It has data fully buffered to ensure efficient computing on the computing array. Since the computing pattern used by the computing array is not always sequential, we need to have the data reorganized before the computing from time to time. To save the data reorganization, we have a specialized DMA module to perform the data movement together with the data reorganization. While all the computing pattern, data movement and on-chip buffer are conducted following the definition of the instructions, NPU thus needs an instruction module to decode the fine-grained controlling signals out of the compact instructions generated by the compiler.

We have the NPU packaged as an IP which can be loaded into Vivado integrator and deployed on Zynq FPGAs directly. The packaged NPU interface is described in Figure 2 and detailed in Table 1.

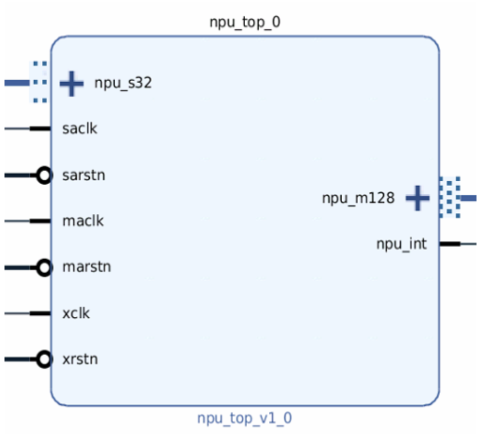

Figure 2 NPU IP interface

Table 1 NPU interface

ID	Interface	Function
1	npu_s32	AXI Slave port, it is used to configure NPU and read the NPU status.
2	npu_m128	AXI Master port, it is used to access DRAM.
3	saclk	AXI Slave clock and it should be lower than 40MHz
4	sarstn	AXI Slave reset
5	maclk	AXI Master clock, it should be lower than 40MHz
6	marstn	AXI Master reset
7	xclk	NPU computing kernel clock, it should be lower than80MHz.
8	xstn	NPU kernel reset
9	npu_int	NPU kernel interruption, active high

The definition of the major NPU registers are listed as follows.

NPU Control Registers (offset=0x0)

Bit	Attr.	Reset Value	Description
31:30	RO	0x0	reserved
29	RW	0x0	reserved
28	RW	0x0	reserved
27	RW	0x0	reserved
26	RW	0x0	reserved
25	RW	0x1	reserved
24	RW	0x1	wrap_type0 {wrap_type1, wrap_type0} =2’h1: the address wrap boundary is 4095/8191/12287/16383. {wrap_type1, wrap_type0} =2’h0: the address wrap boundary is 8191/16383. {wrap_type1, wrap_type0} =2’h2: the address wrap boundary is 16383. Note: This setting may affect the buffer allocation of instruction compiler.
23:16	RW	0x7	reserved
15:12	RW	0x0	reserved
11:9	RO	0x0	reserved
8	RW	0x0	reserved
7:4	RO	0x0	reserved
3	R/W	0x0	reserved
2	R/WSC	0x0	init_inst write 1 to start load instruction. when read, 1'b1: initial instruction is ongoing; 1'b0: initial instruction is done or idle.
1	R/WSC	0x0	restart write 1 to restart NPU from PC (jump pc in jump instruction). when read, 1'b1: NPU is ongoing; 1'b0: NPU is done or idle.
0	R/WSC	0x0	start write 1 to start NPU from PC (idstaddr). when read, 1'b1: NPU is ongoing; 1'b0: NPU is done or idle.

NPU IMR (offset=0x04)

Bit	Attr	Reset Value	Description
31:9	RO	0x0	reserved
8	RW	0x1	reserved
7	RW	0x1	reserved
6	RW	0x1	reserved
5	RW	0x1	reserved
4	RW	0x1	reserved
3	RW	0x1	reserved
2	RW	0x1	reserved
1	RW	0x1	npu_finish_mask 1'b1: npu_finish interrupt is masked 1'b0: npu_finish interrupt is unmasked
0	RW	0x1	init_inst_finish_mask 1'b1: init_inst_finish interrupt is masked 1'b0: init_inst_finish interrupt is unmasked

NPU ISR(offset=0x08)

Bit	Attr	Reset Value	Description
31:28	RO	0x0	reserved
27:24	RO	0x0	reserved
23:20	RO	0x0	reserved
19:16	RO	0x0	npu_finish_status NPU operation finished status, valid only when npu_finish interrupt status asserted. Possible finished status as the following: 4’h0: encounter stop instruction 4’h1: encounter jump instruction 4’h2: encounter un-defined or exceptional instruction 4’h3: DMA error, AXI read access encounter error response 4’h4: DMA error, AXI write access encounter error response 4’h5: DMA queue instruction FIFO overflow happened Others: reserved
15:4	RO	0x0	reserved
15:13	RO	0x0	reserved
12:10	RO	0x0	reserved
9	RO	0x0	reserved
8	W1C	0x0	reserved
7	W1C	0x0	reserved
6	W1C	0x0	reserved
5	W1C	0x0	reserved
4	W1C	0x0	reserved
3	W1C	0x0	reserved
2	W1C	0x0	reserved
1	W1C	0x0	npu_finish NPU operation finished interrupt status. Write 1 to clear this bit.
0	W1C	0x0	init_inst_finish Initial instruction finished interrupt status. Write 1 to clear this bit.

NPU ISRC_ADDR (offset=0x40)

Bit	Attr	Reset Value	Description
31:4	RW	0x0000000	isrcaddr Instruction source external DDR address bit[31:4], bit[3:0] are always 0.
3:0	RO	0x0	reserved

NPU_IDST_ADDR (offset=0x44)

Bit	Attr	Reset Value	Description
31:16	RO	0x0	Reserved
15:0	RW	0x0000	idstaddr Instruction destination buffer address. When set init_inst, it is used as DMA destination buffer start address; When set start, it is used as start instructions PC value.

NPU_INST_DEPTH (offset=0x48)

Bit	Attr	Reset Value	Description
31:16	RO	0x0	reserved
15:0	RW	0x0000	instdepth Instruction depth or length (unit: 16-Byte). Use instdepth+1 as instruction depth or length.