ABHISHEK NAYAK
PUID : 33791158
nayak19@purdue.edu

Hardware Accelerator Design

ECE 695R: System-on-Chip Design (Homework #3)

29th September 2021

OVERVIEW

In this Homework, I have designed a hardware accelerator, added it to the SoC that I earlier built in HW #2, and modified the software to use the accelerator. I was provided with reference software that implements the compute-intensive parts of the original IEEE 802.11b “Wi-Fi” MAC protocol.

DESIGN DESCRIPTION

In the design, the CRC module interfaces with the Avalon bus through the standard Avalon Slave interface.

The processor sends the payload/frame data for CRC computation one byte at a time. A single byte of payload/frame is sent in the lower byte of the 32-bit writedata bus. The CRC computation in the SW implementation inside the ‘for’ loop [index = ((int)(crc_temp >> 24) ^ data[i]) & 0xFF; crc_temp = (crc_temp << 8) ^ crc_table[index];] is translated to RTL design of the CRC Accelerator (in crc_accelerator.v).

The CRC is computed on the following clock cycle using the CRC table stored in the internal memory of the CRC module. The current value of the CRC can be read at any time by reading the base address, but then CRC is reset to 0x4E08BFB4 the following clock cycle. Hence, the final value is read only once from the base address. In other words, the crc_temp value (0x4E08BFB4) initialised before every frame evaluation is set as reset value in the CRC Accelerator HW. The final XOR operation (crc_temp = crc_temp ^ 0xFFFFFFFF) is implemented in software by flipping all the bits of crc_temp.

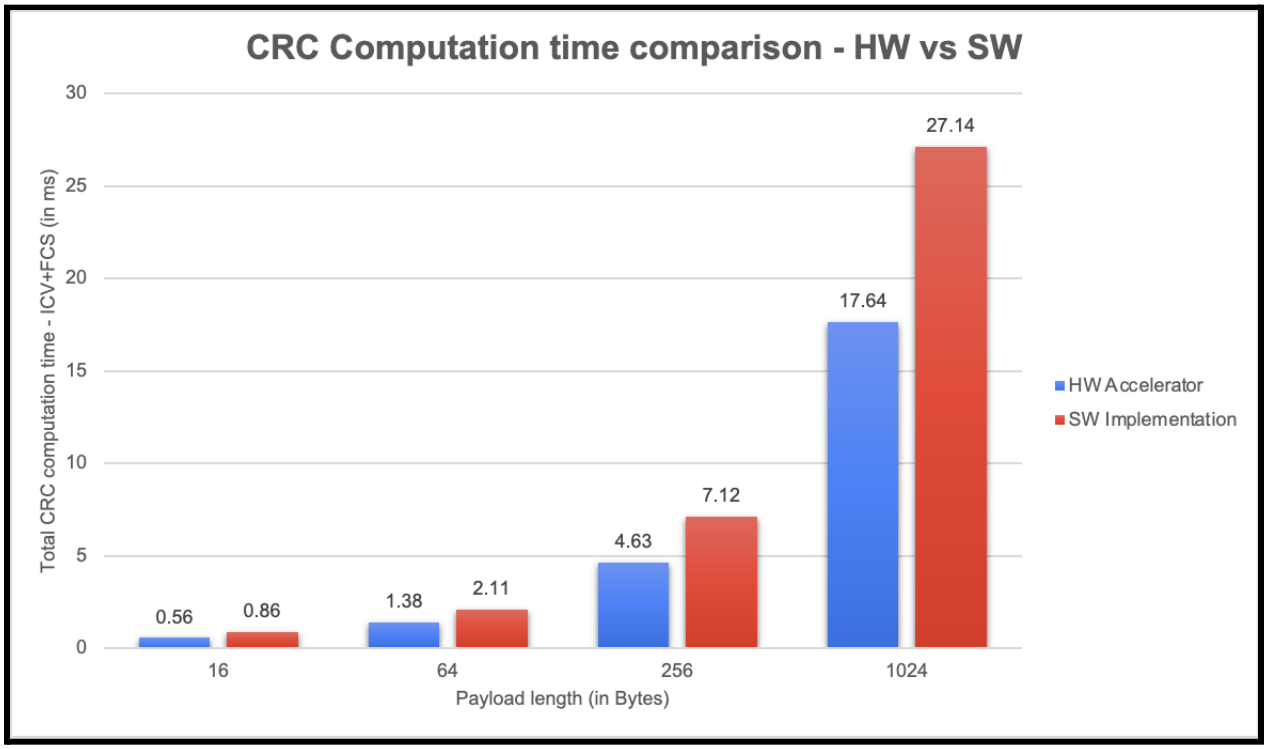
The performance calculator peripheral is also added to the SoC for accurately timing the reference software and HW accelerated implementation.

PERFORMANCE MEASUREMENTS RESULTS

The execution time of the CRC computation (ICV time + FCS time) is measured for a single frame of different payload lengths - 16B, 64B, 256B, and 1024B.

The execution times shown below are measured across the reference software implementation (SW Basic), and primitive CRC accelerator (HW Basic). The speedup achieved with respect to the reference software implementation is shown for the CRC computation time.

Payload length (B)	Output frame size (B)	ICV computation (ms)		FCS computation (ms)		Total CRC computation (ms)		SpeedUp (wrt to SW)	
		HW	SW	HW	SW	HW	SW	HW	SW
16	51	0.15	0.23	0.41	0.63	0.56	0.86	1.54	1
64	99	0.56	0.85	0.82	1.26	1.38	2.11	1.53	1
256	291	2.18	3.36	2.45	3.76	4.63	7.12	1.54	1
1024	1059	8.69	13.37	8.95	13.77	17.64	27.14	1.54	1



It is noticed that the overall CRC computation speeds up by **1.54x** (with respect to SW implementation) when using the HW Accelerator.

In summary, the HW accelerator outperforms the SW implementation by significant margin.

OTHER PERFORMANCE IMPROVEMENT IDEAS

Design Idea 1

In Design IDEA1, a 32-deep synchronous FIFO sits between the Avalon Slave interface and the CRC module.

The processor now sends the payload/frame data for the CRC computation 4 bytes at a time. A word of payload/frame data is sent in the writedata bus by writing to CRC_AVALONSS_INTERFACE_0_BASE. This data is stored in the synchronous FIFO. When full, the transaction is stalled by asserting the waitrequest until the FIFO has the capacity to accept the incoming transaction.

When the FIFO is not empty, the CRC module reads from the FIFO and takes 4 cycles to compute the CRC on the data word. Hence, FIFO is read every 4 cycles by the CRC module.

In addition, the number of memory reads from the processor is reduced by reading one word at a time. This can have a significant impact on the performance of CRC and overall program as the payload and frames are read one byte at a time from memory.

The above design strategy reduces the number of accesses to/from the memory by approximately 75%, when compared to the primitive design. This is because the processor now reads and writes word by word instead of byte by byte of payload/frame data. In addition, the synchronous FIFO at the interface can help the processor to keep writing the payload/frame data even when the CRC module is busy internally.

Design Idea 2

In Design IDEA2, the strategy is for the processor to read starting few unaligned bytes of payload/frame data from memory and write byte by byte into the CRC_CONTROL register. Then, configure the DMA controller to do the data transfer for the payload/frame section in the memory which is word aligned. DMA writes to a fixed location CRC_AVALONSS_INTERFACE_0_BASE to fill the synchronous FIFO. The writes from DMA are efficient and stress the 32-deep synchronous FIFO at the CRC slave interface. The processor can then handle the ending few bytes of frame/payload data that are not word aligned. These bytes are written one by one into the CRC_CONTROL register to evaluate the CRC.