Fully Pipelined AES Core

Subhasis Das

Basic Architecture

This core meets the NIST FIPS-197 specifications. The basic block diagram is given in Figure 1.

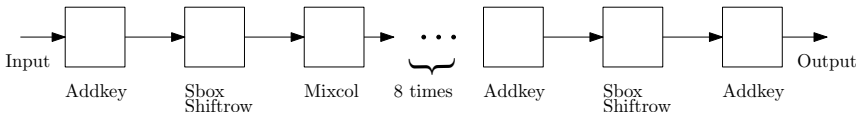

Figure 1: Basic Architecture

I have generated each of the roundkeys in two steps. Let us call

$$\text{RotWord(Sbox}(C_3)\text{) xor RCon } = f(C_3)$$

Then, we can see that

$$\begin{aligned} C'_0 &= f(C_3) \text{ xor } C_0 \\ C'_1 &= f(C_3) \text{ xor } C_0 \text{ xor } C_1 \\ C'_2 &= f(C_3) \text{ xor } C_0 \text{ xor } C_1 \text{ xor } C_2 \\ C'_3 &= f(C_3) \text{ xor } C_0 \text{ xor } C_1 \text{ xor } C_2 \text{ xor } C_3 \end{aligned}$$

where C_i is the column i of the current roundkey and C'_i is the column i of the next roundkey. This first step of generating $f(C_3)$ is done alongwith the addkey step of the previous cycle and the second step is done in the combined S-Box and ShiftRows step.

The inputs to the overall processor are as follows:

- clk_i: System Clock, Data I/O at rising edge
- rst_i: Asynchronous Reset, active high, initializes all inputs to all stages and the final output to zero.
- plaintext_i: 16×8 bits plaintext input
- keyblock_i: 16×8 bits keyblock input

The output is

- ciphertext_o: 16×8 bits ciphertext output

The timing diagram is shown in Figure 2.

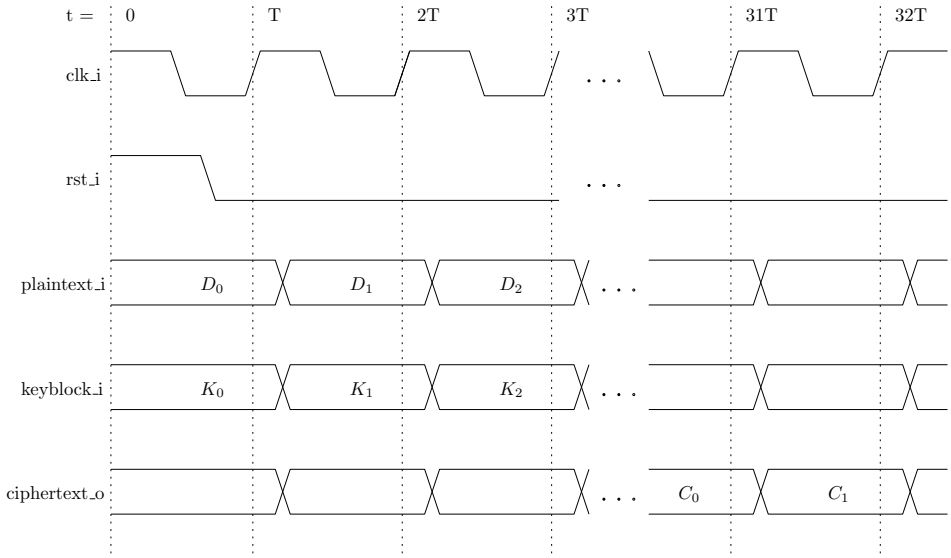

Figure 2: Timing Diagram

The `trunk/rtl/vhdl` directory contains the whole source code.

The sample testbench is in `trunk/bench/vhdl`.

For compiling and running the testbench, the script `sim_isim.sh` in `trunk/sim/rtl_sim/run` directory can be used for Xilinx ISim simulator and `sim_ghdl.sh` for GHDL. The testbench takes in plaintext and key data from `vectors.dat` in `trunk/sim/rtl_sim/src` directory. The expected ciphertext data should be present in `cipher.dat` in `trunk/sim/rtl_sim/src` directory. The results are written to `output.log` in `trunk/sim/rtl_sim/log` directory. The final line is 'OK' if all tests pass, else it is 'FAIL'. This can be used to automate checkings over large test datasets.

The `trunk/syn/Xilinx/run` directory contains the `synth.sh` shell script, which will synthesize the design when run using Xilinx ISE WebPack tools.

The speed optimized synthesis results with timing driven map on a Xilinx 5VLX50T device is shown in Table 1.

f_{max}	$\approx$ 330 MHz
Max throughput	$\approx$ 42 Gbps
Slice Registers's	7873 (27%)
Slice LUT's	14724 (51%)
Bonded IOB's	386 (80%)

Table 1: Design Statistics

All the synthesis, map and place and route logs are available in `trunk/syn/Xilinx/log` directory.