**Documentation of SHA512 IP**

The SHA512 IP communicates via AXI-lite interface with 32 bits data bus. It allows only unidirectional communication as of now.



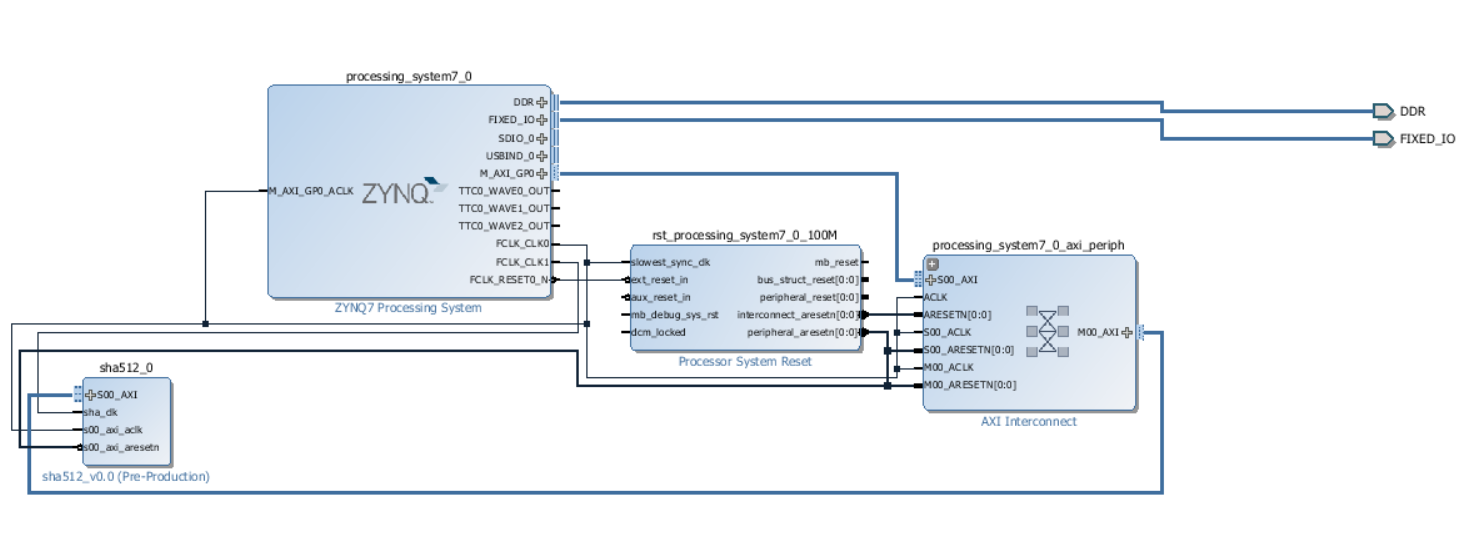
**The addresses of registers are as follows:**

* ADDR\_NAME0 = 8’h00
* ADDR\_NAME1 = 8’h01
* ADDR\_VERSION = 8’h02
* ADDR\_CTRL = 8’h08
* CTRL\_INIT\_BIT = 0
* CTRL\_NEXT\_BIT = 1
* CTRL\_MODE\_LOW\_BIT = 2
* CTRL\_MODE\_HIGH\_BIT = 3
* CTRL\_WORK\_FACTOR\_BIT = 7
* ADDR\_STATUS = 8’h09
* STATUS\_READY\_BIT = 0
* STATUS\_VALID\_BIT = 1
* ADDR\_WORK\_FACTOR\_NUM = 8’h0a
* ADDR\_BLOCK0 = 8’h10
* ADDR\_BLOCK1 = 8’h11
* ADDR\_BLOCK2 = 8’h12
* ADDR\_BLOCK3 = 8’h13
* ADDR\_BLOCK4 = 8’h14
* ADDR\_BLOCK5 = 8’h15
* ADDR\_BLOCK6 = 8’h16
* ADDR\_BLOCK7 = 8’h17
* ADDR\_BLOCK8 = 8’h18
* ADDR\_BLOCK9 = 8’h19
* ADDR\_BLOCK10 = 8’h1a
* ADDR\_BLOCK11 = 8’h1b
* ADDR\_BLOCK12 = 8’h1c
* ADDR\_BLOCK13 = 8’h1d
* ADDR\_BLOCK14 = 8’h1e
* ADDR\_BLOCK15 = 8’h1f
* ADDR\_BLOCK16 = 8’h20
* ADDR\_BLOCK17 = 8’h21
* ADDR\_BLOCK18 = 8’h22
* ADDR\_BLOCK19 = 8’h23
* ADDR\_BLOCK20 = 8’h24
* ADDR\_BLOCK21 = 8’h25
* ADDR\_BLOCK22 = 8’h26
* ADDR\_BLOCK23 = 8’h27
* ADDR\_BLOCK24 = 8’h28
* ADDR\_BLOCK25 = 8’h29
* ADDR\_BLOCK26 = 8’h2a
* ADDR\_BLOCK27 = 8’h2b
* ADDR\_BLOCK28 = 8’h2c
* ADDR\_BLOCK29 = 8’h2d
* ADDR\_BLOCK30 = 8’h2e
* ADDR\_BLOCK31 = 8’h2f
* ADDR\_DIGEST0 = 8’h40
* ADDR\_DIGEST1 = 8’h41
* ADDR\_DIGEST2 = 8’h42
* ADDR\_DIGEST3 = 8’h43
* ADDR\_DIGEST4 = 8’h44
* ADDR\_DIGEST5 = 8’h45
* ADDR\_DIGEST6 = 8’h46
* ADDR\_DIGEST7 = 8’h47
* ADDR\_DIGEST8 = 8’h48
* ADDR\_DIGEST9 = 8’h49
* ADDR\_DIGEST10 = 8’h4a
* ADDR\_DIGEST11 = 8’h4b
* ADDR\_DIGEST12 = 8’h4c
* ADDR\_DIGEST13 = 8’h4d
* ADDR\_DIGEST14 = 8’h4e
* ADDR\_DIGEST15 = 8’h4f
* MODE\_SHA\_512\_224 = 2’h0
* MODE\_SHA\_512\_256 = 2’h1
* MODE\_SHA\_384 = 2’h2
* MODE\_SHA\_512 = 2’h3
* CTRL\_INIT\_VALUE = 2’h1
* CTRL\_NEXT\_VALUE = 2’h2
* CTRL\_WORK\_FACTOR\_VALUE = 1’h1

The ip has been tested with core-clk of 60Mhz and axi-clk of 60Mhz.

**Note:** As of now, both core and axi clocks should be same. Future updates may allow using different clocks

**Example design:**



**Test Code:**

**#include** <stdio.h>

**#include** "platform.h"

**#include** "xil\_printf.h"

**#include** "addr.h"

**#include** "xil\_io.h"

**int** **main**()

{

init\_platform();

**int** i, ready, out\_temp;

print("Hello World\n\r");

Xil\_Out32(XPAR\_SHA512\_0\_S00\_AXI\_BASEADDR+(ADDR\_BLOCK0<<2), 0X61626380);

Xil\_Out32(XPAR\_SHA512\_0\_S00\_AXI\_BASEADDR+(ADDR\_BLOCK1<<2), 0X00000000);

Xil\_Out32(XPAR\_SHA512\_0\_S00\_AXI\_BASEADDR+(ADDR\_BLOCK2<<2), 0X00000000);

Xil\_Out32(XPAR\_SHA512\_0\_S00\_AXI\_BASEADDR+(ADDR\_BLOCK3<<2), 0X00000000);

Xil\_Out32(XPAR\_SHA512\_0\_S00\_AXI\_BASEADDR+(ADDR\_BLOCK4<<2), 0X00000000);

Xil\_Out32(XPAR\_SHA512\_0\_S00\_AXI\_BASEADDR+(ADDR\_BLOCK5<<2), 0X00000000);

Xil\_Out32(XPAR\_SHA512\_0\_S00\_AXI\_BASEADDR+(ADDR\_BLOCK6<<2), 0X00000000);

Xil\_Out32(XPAR\_SHA512\_0\_S00\_AXI\_BASEADDR+(ADDR\_BLOCK7<<2), 0X00000000);

Xil\_Out32(XPAR\_SHA512\_0\_S00\_AXI\_BASEADDR+(ADDR\_BLOCK8<<2), 0X00000000);

Xil\_Out32(XPAR\_SHA512\_0\_S00\_AXI\_BASEADDR+(ADDR\_BLOCK9<<2), 0X00000000);

Xil\_Out32(XPAR\_SHA512\_0\_S00\_AXI\_BASEADDR+(ADDR\_BLOCK10<<2), 0X00000000);

Xil\_Out32(XPAR\_SHA512\_0\_S00\_AXI\_BASEADDR+(ADDR\_BLOCK11<<2), 0X00000000);

Xil\_Out32(XPAR\_SHA512\_0\_S00\_AXI\_BASEADDR+(ADDR\_BLOCK12<<2), 0X00000000);

Xil\_Out32(XPAR\_SHA512\_0\_S00\_AXI\_BASEADDR+(ADDR\_BLOCK13<<2), 0X00000000);

Xil\_Out32(XPAR\_SHA512\_0\_S00\_AXI\_BASEADDR+(ADDR\_BLOCK14<<2), 0X00000000);

Xil\_Out32(XPAR\_SHA512\_0\_S00\_AXI\_BASEADDR+(ADDR\_BLOCK15<<2), 0X00000000);

Xil\_Out32(XPAR\_SHA512\_0\_S00\_AXI\_BASEADDR+(ADDR\_BLOCK16<<2), 0X00000000);

Xil\_Out32(XPAR\_SHA512\_0\_S00\_AXI\_BASEADDR+(ADDR\_BLOCK17<<2), 0X00000000);

Xil\_Out32(XPAR\_SHA512\_0\_S00\_AXI\_BASEADDR+(ADDR\_BLOCK18<<2), 0X00000000);

Xil\_Out32(XPAR\_SHA512\_0\_S00\_AXI\_BASEADDR+(ADDR\_BLOCK19<<2), 0X00000000);

Xil\_Out32(XPAR\_SHA512\_0\_S00\_AXI\_BASEADDR+(ADDR\_BLOCK20<<2), 0X00000000);

Xil\_Out32(XPAR\_SHA512\_0\_S00\_AXI\_BASEADDR+(ADDR\_BLOCK21<<2), 0X00000000);

Xil\_Out32(XPAR\_SHA512\_0\_S00\_AXI\_BASEADDR+(ADDR\_BLOCK22<<2), 0X00000000);

Xil\_Out32(XPAR\_SHA512\_0\_S00\_AXI\_BASEADDR+(ADDR\_BLOCK23<<2), 0X00000000);

Xil\_Out32(XPAR\_SHA512\_0\_S00\_AXI\_BASEADDR+(ADDR\_BLOCK24<<2), 0X00000000);

Xil\_Out32(XPAR\_SHA512\_0\_S00\_AXI\_BASEADDR+(ADDR\_BLOCK25<<2), 0X00000000);

Xil\_Out32(XPAR\_SHA512\_0\_S00\_AXI\_BASEADDR+(ADDR\_BLOCK26<<2), 0X00000000);

Xil\_Out32(XPAR\_SHA512\_0\_S00\_AXI\_BASEADDR+(ADDR\_BLOCK27<<2), 0X00000000);

Xil\_Out32(XPAR\_SHA512\_0\_S00\_AXI\_BASEADDR+(ADDR\_BLOCK28<<2), 0X00000000);

Xil\_Out32(XPAR\_SHA512\_0\_S00\_AXI\_BASEADDR+(ADDR\_BLOCK29<<2), 0X00000000);

Xil\_Out32(XPAR\_SHA512\_0\_S00\_AXI\_BASEADDR+(ADDR\_BLOCK30<<2), 0X00000000);

Xil\_Out32(XPAR\_SHA512\_0\_S00\_AXI\_BASEADDR+(ADDR\_BLOCK31<<2), 0X00000018);Xil\_Out32(XPAR\_SHA512\_0\_S00\_AXI\_BASEADDR+(ADDR\_CTRL<<2), 0X00000000+(MODE\_SHA\_512<<2)+CTRL\_INIT\_VALUE);

**do**{

ready = (Xil\_In32(XPAR\_SHA512\_0\_S00\_AXI\_BASEADDR+(ADDR\_STATUS<<2))) & 0X00000001;

}**while**(!ready);

**for**(i=0;i<16;i++){

out\_temp = Xil\_In32(XPAR\_SHA512\_0\_S00\_AXI\_BASEADDR+((ADDR\_DIGEST0 + i)<<2));

xil\_printf("%08x",out\_temp);

}

cleanup\_platform();

**return** 0;

}

**Result of Testcode:**

Hello World

DDAF35A193617ABACC417349AE20413112E6FA4E89A97EA20A9EEEE64B55D39A2192992A274FC1A836BA3C23A3FEEBBD454D4423643CE80E2A9AC94FA54CA49F

**Note:**

For MODE\_SHA\_512\_224, the output will be

4634270F707B6A54DAAE7530460842E20E37ED265CEEE9A43E8924AA (followed by zeros)

For MODE\_SHA\_512\_256, the output will be

53048E2681941EF99B2E29B76B4C7DABE4C2D0C634FC6D46E0E2F13107E7AF23 (followed by zeros)

For MODE\_SHA\_384, the output will be

CB00753F45A35E8BB5A03D699AC65007272C32AB0EDED1631A8B605A43FF5BED8086072BA1E7CC2358BAECA134C825A7 (followed by zeros)

**Note:** When encrypting next block without initializing again, use CTRL\_REG value as 2