**Documentation of ChaCha IP**



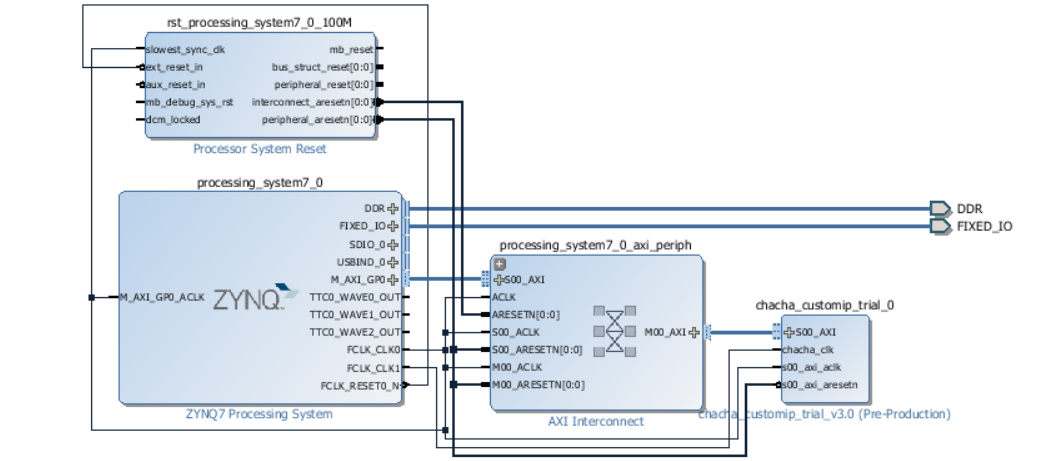
The Chacha IP communicates via AXI-lite interface with 32 bits data bus. It allows only unidirectional communication as of now.

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

* ADDR\_CTRL = 8'h08;
* CTRL\_INIT\_BIT = 0;
* CTRL\_NEXT\_BIT = 1;
* ADDR\_STATUS = 8'h09;
* STATUS\_READY\_BIT = 0;
* ADDR\_KEYLEN = 8'h0a;
* KEYLEN\_BIT = 0;
* ADDR\_ROUNDS = 8'h0b;
* ROUNDS\_HIGH\_BIT = 4;
* ROUNDS\_LOW\_BIT = 0;
* ADDR\_KEY0 = 8'h10;
* ADDR\_KEY1 = 8'h11;
* ADDR\_KEY2 = 8'h12;
* ADDR\_KEY3 = 8'h13;
* ADDR\_KEY4 = 8'h14;
* ADDR\_KEY5 = 8'h15;
* ADDR\_KEY6 = 8'h16;
* ADDR\_KEY7 = 8'h17;
* ADDR\_IV0 = 8'h20;
* ADDR\_IV1 = 8'h21;
* ADDR\_DATA\_IN0 = 8'h40;
* ADDR\_DATA\_IN1 = 8'h41;
* ADDR\_DATA\_IN2 = 8'h42;
* ADDR\_DATA\_IN3 = 8'h43;
* ADDR\_DATA\_IN4 = 8'h44;
* ADDR\_DATA\_IN5 = 8'h45;
* ADDR\_DATA\_IN6 = 8'h46;
* ADDR\_DATA\_IN7 = 8'h47;
* ADDR\_DATA\_IN8 = 8'h48;
* ADDR\_DATA\_IN9 = 8'h49;
* ADDR\_DATA\_IN10 = 8'h4a;
* ADDR\_DATA\_IN11 = 8'h4b;
* ADDR\_DATA\_IN12 = 8'h4c;
* ADDR\_DATA\_IN13 = 8'h4d;
* ADDR\_DATA\_IN14 = 8'h4e;
* ADDR\_DATA\_IN15 = 8'h4f;
* ADDR\_DATA\_OUT0 = 8'h80;
* ADDR\_DATA\_OUT1 = 8'h81;
* ADDR\_DATA\_OUT2 = 8'h82;
* ADDR\_DATA\_OUT3 = 8'h83;
* ADDR\_DATA\_OUT4 = 8'h84;
* ADDR\_DATA\_OUT5 = 8'h85;
* ADDR\_DATA\_OUT6 = 8'h86;
* ADDR\_DATA\_OUT7 = 8'h87;
* ADDR\_DATA\_OUT8 = 8'h88;
* ADDR\_DATA\_OUT9 = 8'h89;
* ADDR\_DATA\_OUT10 = 8'h8a;
* ADDR\_DATA\_OUT11 = 8'h8b;
* ADDR\_DATA\_OUT12 = 8'h8c;
* ADDR\_DATA\_OUT13 = 8'h8d;
* ADDR\_DATA\_OUT14 = 8'h8e;
* ADDR\_DATA\_OUT15 = 8'h8f;

The ip has been tested with core-clk of 50Mhz and axi-clk of 250Mhz.

**Example design:**



**Test Code:**

#include <stdio.h>

#include "platform.h"

#include "xil\_printf.h"

#include "xparameters.h"

#include "sleep.h"

#include "xuartps\_hw.h"

#include "addr.h"

int main()

{

init\_platform();

// u32 r0,r1,addr1,addr2;

u32 ready,out\_temp, i;

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

//write parameters

Xil\_Out32(XPAR\_CHACHA\_CUSTOMIP\_TRIAL\_0\_S00\_AXI\_BASEADDR+(ADDR\_KEY0<<2), 0X00000000);

Xil\_Out32(XPAR\_CHACHA\_CUSTOMIP\_TRIAL\_0\_S00\_AXI\_BASEADDR+(ADDR\_KEY1<<2), 0X00000000);

Xil\_Out32(XPAR\_CHACHA\_CUSTOMIP\_TRIAL\_0\_S00\_AXI\_BASEADDR+(ADDR\_KEY2<<2), 0X00000000);

Xil\_Out32(XPAR\_CHACHA\_CUSTOMIP\_TRIAL\_0\_S00\_AXI\_BASEADDR+(ADDR\_KEY3<<2), 0X00000000);

Xil\_Out32(XPAR\_CHACHA\_CUSTOMIP\_TRIAL\_0\_S00\_AXI\_BASEADDR+(ADDR\_KEY4<<2), 0X00000000);

Xil\_Out32(XPAR\_CHACHA\_CUSTOMIP\_TRIAL\_0\_S00\_AXI\_BASEADDR+(ADDR\_KEY5<<2), 0X00000000);

Xil\_Out32(XPAR\_CHACHA\_CUSTOMIP\_TRIAL\_0\_S00\_AXI\_BASEADDR+(ADDR\_KEY6<<2), 0X00000000);

Xil\_Out32(XPAR\_CHACHA\_CUSTOMIP\_TRIAL\_0\_S00\_AXI\_BASEADDR+(ADDR\_KEY7<<2), 0X00000000);

Xil\_Out32(XPAR\_CHACHA\_CUSTOMIP\_TRIAL\_0\_S00\_AXI\_BASEADDR+(ADDR\_IV0<<2), 0X00);

Xil\_Out32(XPAR\_CHACHA\_CUSTOMIP\_TRIAL\_0\_S00\_AXI\_BASEADDR+(ADDR\_IV1<<2), 0X00);

Xil\_Out32(XPAR\_CHACHA\_CUSTOMIP\_TRIAL\_0\_S00\_AXI\_BASEADDR+(ADDR\_KEYLEN<<2), 0X00);

Xil\_Out32(XPAR\_CHACHA\_CUSTOMIP\_TRIAL\_0\_S00\_AXI\_BASEADDR+(ADDR\_ROUNDS<<2), 0X08);

Xil\_Out32(XPAR\_CHACHA\_CUSTOMIP\_TRIAL\_0\_S00\_AXI\_BASEADDR+(ADDR\_CTRL<<2), 0X01);

//reset and init// wait 4 core clks

usleep(10);

Xil\_Out32(XPAR\_CHACHA\_CUSTOMIP\_TRIAL\_0\_S00\_AXI\_BASEADDR+(ADDR\_CTRL<<2), 0X00);

do{

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

}while(!ready);

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

out\_temp = Xil\_In32(XPAR\_CHACHA\_CUSTOMIP\_TRIAL\_0\_S00\_AXI\_BASEADDR+((ADDR\_DATA\_OUT0 + i)<<2));

// XUartPs\_SendByte(STDOUT\_BASEADDRESS,out\_temp);

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

}

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

//write parameters

Xil\_Out32(XPAR\_CHACHA\_CUSTOMIP\_TRIAL\_0\_S00\_AXI\_BASEADDR+(ADDR\_KEY0<<2), 0X00112233);

Xil\_Out32(XPAR\_CHACHA\_CUSTOMIP\_TRIAL\_0\_S00\_AXI\_BASEADDR+(ADDR\_KEY1<<2), 0X44556677);

Xil\_Out32(XPAR\_CHACHA\_CUSTOMIP\_TRIAL\_0\_S00\_AXI\_BASEADDR+(ADDR\_KEY2<<2), 0X8899aabb);

Xil\_Out32(XPAR\_CHACHA\_CUSTOMIP\_TRIAL\_0\_S00\_AXI\_BASEADDR+(ADDR\_KEY3<<2), 0Xccddeeff);

Xil\_Out32(XPAR\_CHACHA\_CUSTOMIP\_TRIAL\_0\_S00\_AXI\_BASEADDR+(ADDR\_KEY4<<2), 0Xffeeddcc);

Xil\_Out32(XPAR\_CHACHA\_CUSTOMIP\_TRIAL\_0\_S00\_AXI\_BASEADDR+(ADDR\_KEY5<<2), 0Xbbaa9988);

Xil\_Out32(XPAR\_CHACHA\_CUSTOMIP\_TRIAL\_0\_S00\_AXI\_BASEADDR+(ADDR\_KEY6<<2), 0X77665544);

Xil\_Out32(XPAR\_CHACHA\_CUSTOMIP\_TRIAL\_0\_S00\_AXI\_BASEADDR+(ADDR\_KEY7<<2), 0X33221100);

Xil\_Out32(XPAR\_CHACHA\_CUSTOMIP\_TRIAL\_0\_S00\_AXI\_BASEADDR+(ADDR\_IV0<<2), 0X0f1e2d3c);

Xil\_Out32(XPAR\_CHACHA\_CUSTOMIP\_TRIAL\_0\_S00\_AXI\_BASEADDR+(ADDR\_IV1<<2), 0X4b596877);

Xil\_Out32(XPAR\_CHACHA\_CUSTOMIP\_TRIAL\_0\_S00\_AXI\_BASEADDR+(ADDR\_KEYLEN<<2), 0X01);

Xil\_Out32(XPAR\_CHACHA\_CUSTOMIP\_TRIAL\_0\_S00\_AXI\_BASEADDR+(ADDR\_ROUNDS<<2), 0X08);

Xil\_Out32(XPAR\_CHACHA\_CUSTOMIP\_TRIAL\_0\_S00\_AXI\_BASEADDR+(ADDR\_CTRL<<2), 0X01);

usleep(10);

Xil\_Out32(XPAR\_CHACHA\_CUSTOMIP\_TRIAL\_0\_S00\_AXI\_BASEADDR+(ADDR\_CTRL<<2), 0X00);

do{

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

}while(!ready);

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

out\_temp = Xil\_In32(XPAR\_CHACHA\_CUSTOMIP\_TRIAL\_0\_S00\_AXI\_BASEADDR+((ADDR\_DATA\_OUT0 + i)<<2));

// XUartPs\_SendByte(STDOUT\_BASEADDRESS,out\_temp);

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

}

cleanup\_platform();

return 0;

}

**Result of Testcode:**

* he28a5fa4a67f8c5defed3e6fb7303486aa8427d31419a729572d777953491120b64ab8e72b8deb85cd6aea7cb6089a101824beeb08814a428aab1fa2c816081b
* h60fdedbd1a280cb741d0593b6ea0309010acf18e1471f68968f4c9e311dca149b8e027b47c81e0353db013891aa5f68ea3b13dd2f3b8dd0873bf3746e7d6c567

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