Atmark Techno

XPS SIL00 (v1.00a)

Introduction

SUZAKU でプログラミングを行った際、動作確認に苦労したことはないでしょうか？ SUZAKU I/O シリーズの LED/SW ボードは単色 LED、押しボタンスイッチ、ロータリコードスイッチ、7 セグメント LED、シリアルポートが搭載されています。SUZAKU を使った開発において、簡単な動作確認、FPGA や Linux の学習用としてご利用いただけるように設計されております。
XPS-SIL00 を使うことにより LED/SW ボードにスロット機能を簡単に実現できます。

Features

XPS-SIL00 は以下のような特徴を持っています。

- SUZAKU I/O シリーズ LED/SW ボード専用
- 7 セグメント LED の数字をデコードして出力
- 7 セグメント LED をダイナミック点灯
- 押しボタンスイッチ、ロータリコードスイッチの入力を受け取る
- トリガー信号により、単色 LED を順次点灯
- 割り込み

Block Diagram

XPS-SIL00 のブロック図を示します。

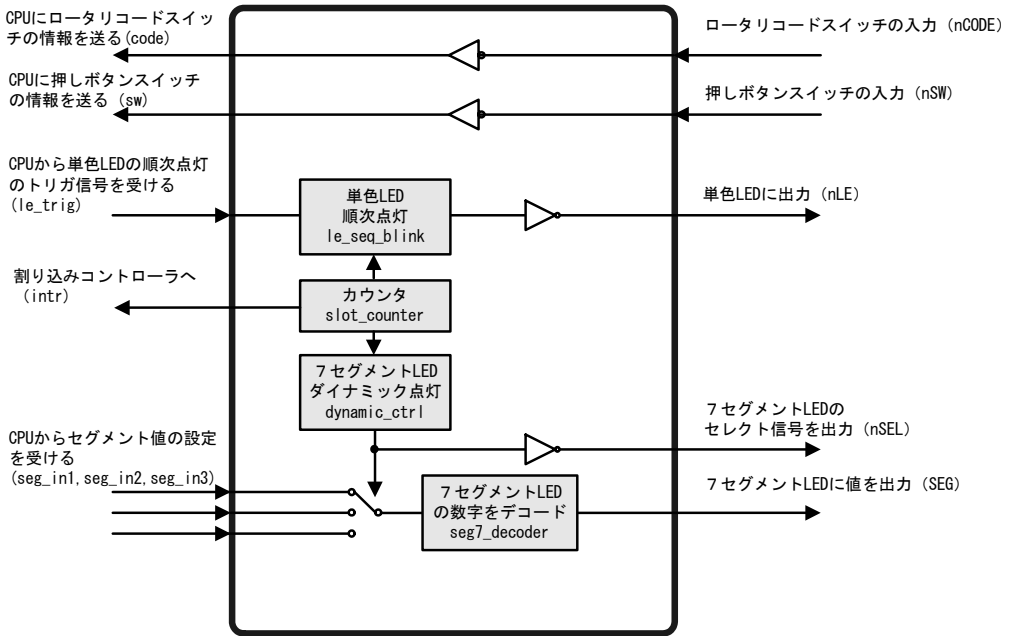

Figure 1 : XPS-SIL00 Block Diagram

Parameters

XPS-SIL00 のパラメータを示します。

Table 1 : Design Parameters

Feature/Description	Parameter Name	Allowable Values	Default Values	VHDL Type
XPS-SIL00 Base Address	C_BASEADDR	Valid Address	None	std_logic_vector
XPS-SIL00 High Address	C_HIGHADDR	Valid Address	None	std_logic_vector
PLB Address Width	C_PLB_AWIDTH	32	32	integer
PLB Data Width	C_PLB_DWIDTH	32, 64, 128	128	integer
Number of PLB Masters	C_SPLB_NUM_MASTERS	1:16	8	Integer
PLB Master ID Bus Width	C_SPLB_MID_WIDTH	1:4	3	Integer
Width of the Slave Data Bus	C_SPLB_NATIVE_DWIDTH	32, 64, 128	32	Integer
Selects point-to-point or shared PLB topology	C_SPLB_P2P	0,1	0	Integer
Selects the transactions as being single beat or burst	C_SPLB_SUPPORT_BURSTS	0,1	0	Integer
Data Width of Smallest Master	C_SPLB_SMALLEST_MASTER	32, 64, 128	32	Integer
Period of the PLB Clock	C_SPLB_CLK_PERIOD_PS	Valid Period	11428	Integer
Target FPGA Family	C_FAMILY	virtex4	virtex4	string

I/O Signals

XPS-SIL00 の I/O Signals を示します。

Table 2 : I/O Signals

Signal Name	I/O	Initial State	Description
SPLB_Clk	I		PLB Clock
SPLB_Rst	I		PLB RESET
PLB_ABus[0:31]	I		PLB Address Bus
PLB_UABus[0:31]	I		PLB Upper Address Bits
PLB_PAValid	I		PLB Primary Address Valid
PLB_SAValid	I		PLB Secondary Address Valid
PLB_rdPrim	I		PLB Read Primary
PLB_wrPrim	I		PLB Write Primary
PLB_masterID[0:C_SPLB_MID_WIDTH-1]	I		PLB Master Identification
PLB_abort	I		PLB Transaction Abort
PLB_busLock	I		PLB Bus Lock
PLB_RNW	I		PLB Read Not Write
PLB_BE[0:C_SPLB_DWIDTH/8-1]	I		PLB Byte Enable
PLB_MSize[0:1]	I		PLB Master Size
PLB_size[0:3]	I		PLB Size
PLB_type[0:2]	I		PLB Transaction Type
PLB_lockErr	I		PLB Bus Lock Error
PLB_wrDBus[0:C_SPLB_DWIDTH-1]	I		PLB Write Data Bus
PLB_wrBurst	I		PLB Write Burst
PLB_rdBurst	I		PLB Read Burst
PLB_wrPendReq	I		PLB Write Pending Request
PLB_rdPendReq	I		PLB Read Pending Request

PLB_wrPendPri[0:1]	I		PLB Write Pending Priority
PLB_rdPendPri[0:1]	I		PLB Read Pending Priority
PLB_reqPri[0:1]	I		PLB Request Primary
PLB_TAttribute[0:15]	I		PLB Transaction Attribute
Sl_addrAck	O	0	PLB Slave Address Acknowledge
Sl_SSize[0:1]	O	0	PLB Slave Size
Sl_wait	O	0	PLB Slave Wait
Sl_rearbitrate	O	0	PLB Rearbitrate
Sl_wrDAck	O	0	PLB Write Data Acknowledge
Sl_wrComp	O	0	PLB Write Complete
Sl_wrBTerm	O	0	PLB Burst Terminate
Sl_rdDBus[0:C_SPLB_DWIDTH-1]	O	0	PLB Read Data Bus
Sl_rdWdAddr[0:3]	O	0	PLB Read Word Address
Sl_rdDAck	O	0	PLB Read Data Acknowledge
Sl_rdComp	O	0	PLB Read Complete
Sl_rdBTerm	O	0	PLB Read Burst Terminate
Sl_MBusy[0:C_SPLB_NUM_MASTERS-1]	O	0	PLB Slave Master Busy
Sl_MWrErr[0:C_SPLB_NUM_MASTERS-1]	O	0	PLB Master Write Error
Sl_MRdErr[0:C_SPLB_NUM_MASTERS-1]	O	0	PLB Read Error
Sl_MIRQ[0:C_SPLB_NUM_MASTERS-1]	O	0	PLB Master Interrupt Request
IP2INTC_Irpt	O	0	XPS-SIL00 Interrupt Active high signal
SEG(0:7)	O	0	7Seg LED Signal Output
nSEL(0:2)	O	1	7Seg LED Select Signal Output
nLE(0:3)	O	1	LED Signal Output
nSW(0:2)	I		Push SW Signal Input
nCODE(0:3)	I		Rotary SW Signal Input

Register

XPS-SIL00 の LED、スイッチ制御レジスタを示します。

Table 3 : Register Overview

Register Name	Description	PLB Address	Access
seg_in1	7Seg LED1 Data Register	C_BASEADDR+0x00	Read/Write
seg_in2	7Seg LED2 Data Register	C_BASEADDR+0x01	Read/Write
seg_in3	7Seg LED3 Data Register	C_BASEADDR+0x02	Read/Write
le_trig	LED Trigger Signal Register	C_BASEADDR+0x03	Read/Write
sw	Push SW Data Register	C_BASEADDR+0x04	Read
code	Rotary SW Data Register	C_BASEADDR+0x05	Read

※8bit アクセスのみ対応。16bit 及び 32bit アクセスはできません。

• seg_in1~3(7Seg LED1~3 Data Register)

0～F を 7 セグメント LED に読み書きします。

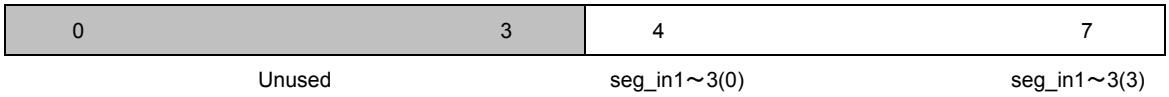

Figure 2 : 7SegLED Data Register

Table 4 :7SegLed Data Register Description

Bits	Name	Core Access	Description	Reset Value
0 : 3	Unused	Read/Write	Unused	zeroes
4	seg_in1～3(0)	Read/Write	7 セグメント LED のデータを読み書きします。 0～F の 16 進値を設定すると、セグメント LED に 0～F の数字を表示します。	0
5	seg_in1～3(1)	Read/Write		0
6	seg_in1～3(2)	Read/Write		0
7	seg_in1～3(3)	Read/Write		0

• le_trig(LED Trigger Register)

‘0’で消灯、‘1’で単色 LED1～4 が順次点灯します。

Figure 3 : LED Trigger Register

Table 5 : LED Trigger Register Description

Bits	Name	Core Access	Description	Reset Value
0 : 6	Unused	Read/Write	Unused	zeroes
7	le_trig	Read/Write	0 : 単色 LED　消灯 1 : 単色 LED　順次点灯	0

• sw(Push SW Data Register)

押しボタンスイッチ(SW1～SW3)の値を受け取ります。

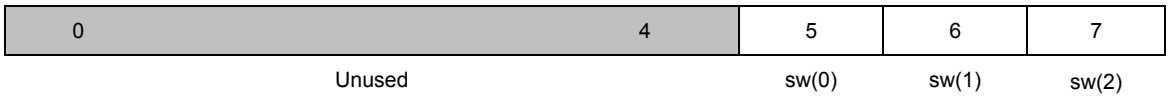

Figure 4 : Push SW Data Register

Table 6 : Push SW Data Register Description

Bits	Name	Core Access	Description	Reset Value
0 : 4	Unused	Read/Write	Unused	zeroes
5	sw(0)	Read	SW3 の ON/OFF を読み込む 0 : OFF 1 : ON	0
6	sw(1)	Read	SW2 の ON/OFF を読み込む 0 : OFF 1 : ON	0
7	sw(2)	Read	SW1 の ON/OFF を読み込む 0 : OFF 1 : ON	0

• code(Rotary SW Data Register)

ロータリコードスイッチの値を受け取ります。

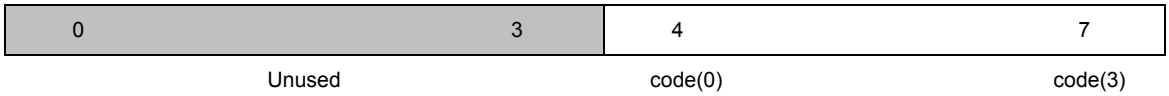

Figure 5 : Rotary SW Data Register

Table 7 : Rotary SW Data Register Description

Bits	Name	Core Access	Description	Reset Value
0 : 3	Unused	Read/Write	Unused	zeroes
4	code(0)	Read	ロータリコードスイッチの値を 0～F の 16 進値で読み込みます。	0
5	code(1)	Read		0
6	code(2)	Read		0
7	code(3)	Read		0

Interrupt

XPS-SIL00 の割り込みレジスタを示します。

Table 8 : Interrupt Parameters

Register Name	Description	PLB Address	Access
DIER	Device Interrupt Enable Register	C_BASEADDR+0x08	Read/Write
DGIER	Global Interrupt Enable	C_BASEADDR+0x1C	Read/Write
IPISR	IP Interrupt Status Register	C_BASEADDR+0x20	Read/Tpggle on Write
IPIER	IP Interrupt Enable Register	C_BASEADDR+0x28	Read/Write

- **DIER(Device Interrupt Enable Register) Offset 0x08**

Figure 6 : Device Interrupt Enable Register

Table 9 : Device Interrupt Enable Register Description

Bits	Name	Core Access	Description	Reset Value
0 : 27	Unused	Read	Unused	zeroes
28	LVIP	Read/Write	Level Interrupt Enable 0 : disabled 1 : enabled	0
29	IPIRE	Read/Write	Interrupt Request Enable 0 : disabled 1 : enabled	0
30	IE(1)	Read/Write	Interrupt Enable 1 0 : disabled 1 : enabled	0
31	IE(0)	Read/Write	Interrupt Enable 0 0 : disabled 1 : enabled	0

• DGIER(Device Global Interrupt Enable Register) Offset 0x1C

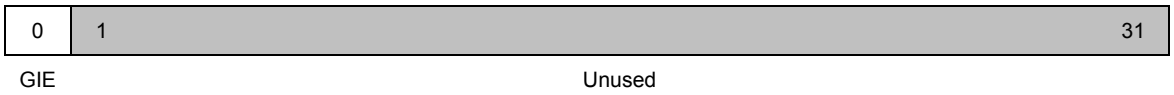

Figure 7 : Device Global Interrupt Enable Register

Table 10 : Device Global Interrupt Enable Register Description

Bits	Name	Core Access	Description	Reset Value
0	GIE	Read/Write	Global Interrupt Enable. 0 : Interrupt disabled; no interrupt possible from this device. 1 : Device Interrupt enabled.	0
1 : 31	Unused	Read	Unused	zeroes

• IPISR(IP Interrupt Status Register) Offset 0x20

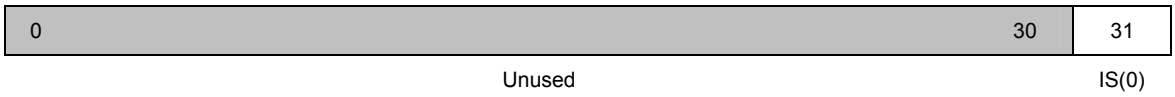

Figure 8 : IP Interrupt Status Register

Table 11 : IP Interrupt Status Register Description

Bits	Name	Core Access	Description	Reset Value
0 : 30	Unused	Read	Unused	zeroes
31	IS(0)	Read/TOW (Toggle on Write) 1 : Write	Interrupt Status 0 : not active. 1 : active	0

• IPIER(IP Interrupt Enable Register) Offset 0x28

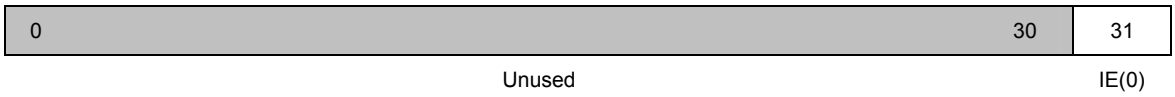

Figure 9 : IP Interrupt Enable Register

Table 12 : IP Interrupt Enable Register Description

Bits	Name	Core Access	Description	Reset Value
0 : 30	Unused	Read	Unused	zeroes
31	IE(0)	Read/Write	Interrupt Enable 0 : disabled 1 : enabled	0

Sample Program

• Slot Machine

XPS-SIL00 によるスロット機能実現のためのサンプルプログラム（C 言語）を示します。

Example 1 : Slot Machine Sample C Source

```
#include <ctype.h>
typedef unsigned char u8;

#define SLOT_BASE_ADDR   0xf0ffd000

/*  slot status  */
#define SLOT_READY              (0)
#define SLOT_RUN                (1)
#define SLOT_STOP               (2)

/* switch */
#define CHATTERING_CLEAR_CNT    (2)
#define SWITCH_NUM              (3)
#define SWITCH(no)              (1 << (no))
#define SWITCH_ALL              (SWITCH(0) | SWITCH(1) | SWITCH(2))
#define READ_SWITCH             (*(volatile u8 *)(SLOT_BASE_ADDR + 0x4))

/* 7seg-led */
#define SEG7_NUM                (3)
#define SEG7(no)                (1 << (no))
#define SEG7_ALL                (SEG7(0) | SEG7(1) | SEG7(2))
#define WRITE_SEG7(no,v)        (*(volatile u8 *)(SLOT_BASE_ADDR + (no)) = (v))

/* led */
#define WRITE_LEDS(v)           (*(volatile u8 *)(SLOT_BASE_ADDR + 0x3) = (v))

/* rotary switch */
#define READ_RSWITCH            (*(volatile u8 *)(SLOT_BASE_ADDR + 0x5))

static u8 clear_switch_chattering(void)
{
    static u8 count[SWITCH_NUM];
    static u8 status = 0;
    u8       reg_sw = READ_SWITCH;
    int      i;

    for (i = 0; i < SWITCH_NUM; i++) {
        if (reg_sw & SWITCH(i)) {
            if (count[i] < CHATTERING_CLEAR_CNT)
                count[i]++;
        } else {
            if (count[i] > 0)
                count[i]--;
        }

        if (count[i] == CHATTERING_CLEAR_CNT)
```

```
            status |=  SWITCH(i);
        else if (count[i] == 0)
            status &= ~SWITCH(i);
    }

    return status;
}

static void blink_leds(u8 st, u8 value[])
{
    u8 v = 0;
    if (st == SLOT_READY) {
        if (((value[0] & 0xf) == (value[1] & 0xf)) &&
            ((value[0] & 0xf) == (value[2] & 0xf)))
            v = 1;
    }
    WRITE_LEDS(v);
}

static u8 change_rotation_speed(void)
{
    u8 speed;

    switch (READ_RSWITCH) {
    case 0:
        speed = 0x3f;
        break;
    case 1:
        speed = 0x1f;
        break;
    default:
        speed = 0x0f;
        break;
    }

    return speed;
}

static int get_number_pushed_switches(u8 st)
{
    int num = 0;
    int i;

    for (i = 0; i < SWITCH_NUM; i++) {
        if ((st >> i) & 0x1)
            num++;
    }
    return num;
}

int slot(void)
{
    static u8 schedule = 0;
    static u8 sw_status = 0;
    static u8 slot_status = SLOT_READY;
```

```
    static u8 seg7_status = 7;     /* 0: unpushed, 1: pushed */
    static u8 seg7_val[SEG7_NUM] = {3,2,1};
    int  i;

    schedule++;

    sw_status = clear_switch_chattering();

    switch (slot_status) {
    case SLOT_READY:
        if (get_number_pushed_switches(sw_status) > 1)
            slot_status = SLOT_RUN;
        break;
    case SLOT_RUN:
        if (sw_status == 0)
            slot_status = SLOT_STOP;
        break;
    case SLOT_STOP:
        if (seg7_status == SEG7_ALL)
            slot_status = SLOT_READY;
        break;
    }

    if (slot_status == SLOT_STOP) {
        for (i = 0; i < SWITCH_NUM; i++) {
            if (sw_status & SWITCH(i))
                seg7_status |= SEG7(i);
        }
    } else if (slot_status == SLOT_RUN) {
        seg7_status = 0;
    }
    if ((schedule & change_rotation_speed()) == change_rotation_speed()) {
        for (i = 0; i < SEG7_NUM; i++) {
            if ((seg7_status & SEG7(i)) == 0)
                seg7_val[i]++;
        }
    }
    for (i = 0; i < SEG7_NUM; i++)
        WRITE_SEG7(i, seg7_val[i]);

    blink_leds(slot_status, seg7_val);

    return 0;
}
```

• Interrupt

XPS-SIL00 で割り込みを利用する際の MHS およびサンプルプログラム (C 言語) を示します。

PowerPC

Example 2 : PowerPC Interrupt MHS File

```
BEGIN ppc405
 # 中略
 PORT EICC405EXTINPUTIRQ = eicc405extinputirq
END

BEGIN xps_intc
 # 中略
 PORT Irq = eicc405extinputirq
 PORT Intr =  sil_intr
END

BEGIN xps_sil00
 PARAMETER INSTANCE = sil_cntlr
 PARAMETER C_BASEADDR = 0xF0FFD000
 PARAMETER C_HIGHADDR = 0xF0FFD1FF
 # 中略
 PORT IP2INTC_Irpt = sil_intr
END
```

Example 3 : Interrupt Sample C Source

```
#include <ctype.h>
#include "medium.h"
#include "slot.h"
#include "xparameters.h"
#include "xintc_l.h"
#ifdef __PPC__
#include "xpseudo_asm_gcc.h"
#include "xexception_l.h"
#endif

int sil_interrupt_handler(void * baseaddr_p)
{
    slot();
}

int interrupt_init(void)
{
    /*enable interrupt */
#ifdef __PPC__
    mtevpr(0xFFFF0000);
    for(index = XEXC_ID_FIRST; index < XEXC_ID_LAST + 1; index++){
       XExc_RegisterHandler(index, (XExceptionHandler)NullHandler, XNULL);
    }

XExc_RegisterHandler(XEXC_ID_NON_CRITICAL_INT,(XExceptionHandler)XIntc_DeviceInterruptHandler,
```

```
    (void*)XPAR_INTC_SYSTEM_DEVICE_ID);
    XExc_mEnableExceptions(XEXC_NON_CRITICAL);
#else
    microblaze_enable_interrupts();
#endif
    /* Start the interrupt controller */
    XIntc_mMasterEnable(XPAR_INTC_SYSTEM_BASEADDR);
    /* Enable timer interrupt in the interrupt controller */
    XIntc_mEnableIntr(XPAR_INTC_SYSTEM_BASEADDR, XPAR_SIL_CNTLR_IP2INTC_IRPT_MASK);
    /* start the intr sil */
     /* Enable all interrupt source from user logic.*/
    XPS_SIL00_mWriteReg(XPAR_SIL_CNTLR_BASEADDR, XPS_SIL00_INTR_IPIER_OFFSET, 0x00000001);
     /* Enable all possible interrupt sources from device.*/
    XPS_SIL00_mWriteReg(XPAR_SIL_CNTLR_BASEADDR,  XPS_SIL00_INTR_DIER_OFFSET,  INTR_TERR_MASK  |
INTR_DPTO_MASK | INTR_IPIR_MASK);
     /* Set global interrupt enable.*/
    XPS_SIL00_mWriteReg(XPAR_SIL_CNTLR_BASEADDR, XPS_SIL00_INTR_DGIER_OFFSET, INTR_GIE_MASK);
}

int interrupt_clean(void)
{
    /* disable interrupt */
#ifdef __PPC__
    XExc_mDisableExceptions(XEXC_NON_CRITICAL);
#else
    microblaze_disable_interrupts();
#endif
    /* Disable timer interrupt in the interrupt controller */
    XIntc_mDisableIntr(XPAR_INTC_SYSTEM_BASEADDR, XPAR_SIL_CNTLR_IP2INTC_IRPT_MASK);
    /* stop the timer */
    XPS_SIL00_mWriteReg(XPAR_SIL_CNTLR_BASEADDR, XPS_SIL00_INTR_IPIER_OFFSET, 0);
    XPS_SIL00_mWriteReg(XPAR_SIL_CNTLR_BASEADDR, XPS_SIL00_INTR_DIER_OFFSET,0);
    XPS_SIL00_mWriteReg(XPAR_SIL_CNTLR_BASEADDR, XPS_SIL00_INTR_DGIER_OFFSET, 0);
}

int main(void)
{
       interrupt_init();
       while(1);
       //interrupt_clean();
       return 0;
}
```

Revision History

Date	Version	Revision
2008/02/15	1.0	初版作成