Application Note 009: Search Subsystem Initialization using LNI8010 Network Database Coprocessor

1 INTRODUCTION

This application note explains the initialization procedure for a search subsystem built around Cypress Semiconductor Corporation's LNI8010 Network Coprocessor and LNI70X0 Network Search Engines (NSEs). Cypress Semiconductor's search subsystem is configurable to meet different kinds of interface and system requirements. A typical search subsystem is considered here; the initialization procedures will have to be modified appropriately in order to reflect any deviation from the example shown in this application note.

For details on a specific part, please refer to Cypress Semiconductor product specifications. Sample code is included for reference (see Section 5).

2 SEARCH SUBSYSTEM

Cypress Semiconductor's LNI70X0 family of NSEs, which employ leading-edge Associative Processing Technology™ (APT), empower network processors' lookup and/or search capabilities with very fast and deterministic response times. These NSEs are cascadable to support the requirements of varying depths of database tables, and can also interface with optional SSRAMs.

NSEs have a unique interface, and may require bridging circuitry to interface with a network processor. Cypress's LNI8010 Network Database Coprocessor bridges a network processor with Cypress's LNI70X0 NSEs. The LNI8010 device has a glueless interface to Cypress Semiconductor's NSEs as well as an industry standard SSRAM bus interface for easy accessibility from a network processor (note that most network processors have an SSRAM interface). LNI8010 also manages the search pipeline in order to optimize the search process. All accesses to the LNI70X0 NSEs or SSRAMs from the network processor are through the LNI8010 coprocessor.

SSRAMs may be connected directly to a network processor ("index mode") or to LNI70X0 NSEs, gaining access through an LNI8010 coprocessor ("associative mode"). Associative mode is used in building higher-performance subsystems, and is considered for this application note (as is the cascaded LNI70X0 NSEs in a search subsystem).

The hardware configuration of the LNI8010 coprocessor is decided by the hard-wired configuration bits. These bits select the width of the interface, the byte order format for access, and the type of SSRAM interface used.

The LNI8010 device is shown in Figure 1, and the hard-wired configuration bits are set as follows.

- **IWIDTH**: Logical 0. This selects 32 bits as the coprocessor data bus width.
- **BIG/LTL_L**: Logical 0. This selects the Little Endian byte order format.
- **IFC_CFG[2:0]**: All bits at Logical 0. These bits select the coprocessor interface type. ZBT pipelined mode (000) is selected.

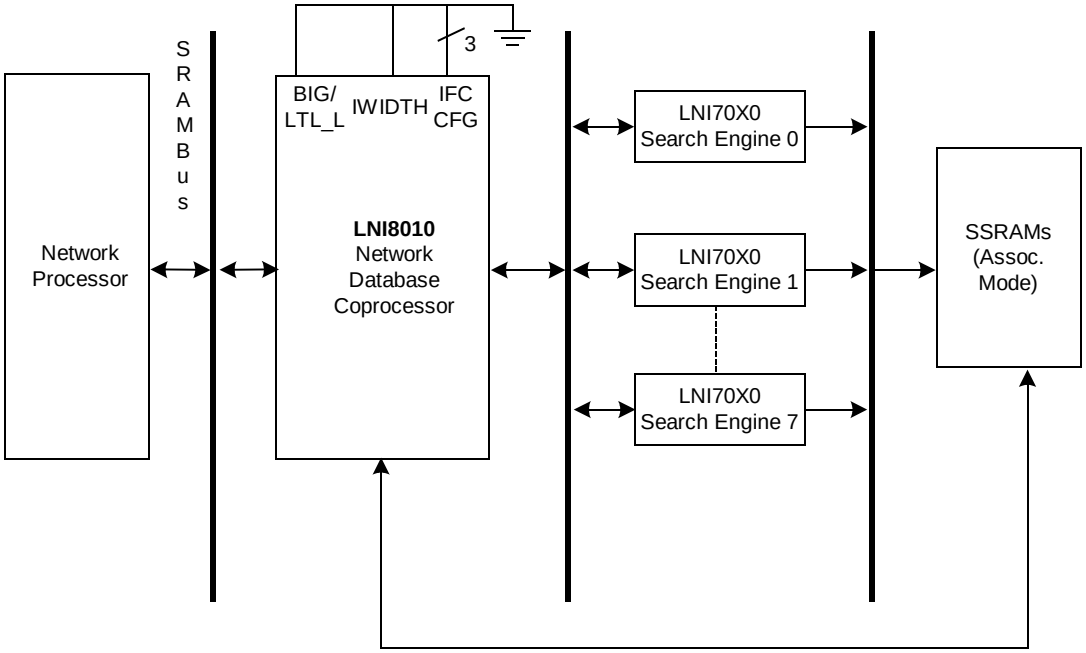

FIGURE 1. SYSTEM CONFIGURATION IN ASSOCIATIVE MODE

3 INITIALIZATION PROCEDURES

On power-up, it is necessary to initialize both the LNI8010 coprocessor and LNI70X0 NSEs. The LNI8010 device is initialized first, followed by the search engine registers and then the data and mask arrays. Please refer to the Cypress LNI8010 and LNI70X0 product specifications for more details.

The LNI8010 coprocessor is then memory mapped into the network processor's SSRAM region. It can be accessed through 4K of memory space. The type of interface presented to the network processor is selected by the hard-wired IFC_CFG[2:0] pins. The hard-wired pin IWIDTH selects the data-bus width, while another hard-wired pin, BIG/LTL_L, selects the big- or little-Endian byte-ordering format.

The coprocessor has a configuration area of 1K, and an operating area of 2K. The configuration area is used for programming the coprocessor via a 64-bit configuration register.

Once the LNI8010 device is configured, the network processor will use the operating area to configure the NSEs, initialize and manage the protocol layer tables, and perform searches.

The following steps explain the system initialization procedures:

1. **Do the software reset on LNI8010 coprocessor**. Set the software reset (SRST) bit in the configuration register (see Table 1). The SRST will be active for eight clock cycles and will automatically clear after the reset has taken effect.
2. **Disable error interrupts signals**. Mask interrupts by writing the mask register in the LNI8010 coprocessor with "0xFFFFFFFF" (all ones).
3. **Set the LNI8010 coprocessor's configuration parameters**. Program the configuration register in the coprocessor with appropriate values determined by the following hardware configuration:

- **TLSZ**. Determines the NSE configuration for the specific table size (depending on the number of NSEs).
- **HLAT**. Determines the latency of the associated SSRAMs (depending on the SSRAMs used)
- **CPCFG**. Sets the width of the processor and context IDs that will be driven on the CPID bus after the completion of the operation.
- **SSRAM present**. Indicates whether or not the associative SSRAM is connected to the NSEs.
- **INTR polarity**. Selects the polarity of the interrupt signal.
- **Search result bit in data field**. Decides if the HIT/MISS information will be attached to the associative data field in bit[63].
- **External transceiver present**. Set if an external transceiver is used to drive several NSEs.

4. Perform SRST on LNI70X0 NSEs using a broadcast WRITE command from the LNI8010 coprocessor.
5. Write the configuration parameters (see Table 2) into the LNI70X0 NSEs' command registers using a broadcast WRITE command from the LNI8010 device. **Note**. All other devices except the last one in the chain must be programmed with LDEV = 0 (bit[7]) and LRAM =0 (bit[8])
6. Rewrite the configuration parameters of the last LNI70X0 device's command register using a WRITE command from the LNI8010 device. Here LDEV bit[7] = 1 and LRAM bit[8] = 1. With the LDEV bit set, this device will drive the search successful flag (SSF) and the search successful valid (SSV) signals in cycles in which none of the upstream devices has driven these signals.

   The last RAM bit of the table (LRAM) is indicative of the last NSE on the SRAM bus. When set, the last device drives the SADR, OE_L, WE_L, and ALE_L signals.
7. Set the Global Mask Register (GMR) pair 0 (GMRs 0 and 1) with all ones (0xFFFFFFFF). A GMR must be programmed for NSE data writes, because one out of sixteen mask registers is always used during a data array or mask array WRITE (or SEARCH) operation. Initially, one mask register pair must be programmed with all bits to 1; selecting it during a WRITE operation allows NSE data to be written.

   The LNI7010 and LNI7020 devices must be initialized with all 0s in the NSE's data and all 1s for the NSE mask before the user loads any useful information.
8. Use the programmed GMR pair 0 to initialize the mask array.
9. Use the programmed GMR pair 0 to initialize the data array.
10. If associative mode SSRAMs are present in the search subsystem, write the SSRAM entries with appropriate values. For WRITEs into SSRAMs, only 64 bits of data are relevant. The use of layer attribute valid bits in the LNI8010 descriptor command is not required.

The NSE is now ready for SEARCH operations. Sample code is provided at the end of this application note for reference.

The LNI7010 and LNI7020 devices' database entry size is configurable as 34 bits, 68 bits, 136 bits, or 272 bits. In the 34-bit-entry mode, the database size is 32K entries. In the 68-bit-entry mode, the database size is 16K entries, in the 136-bit-entry mode, the database size is 8K, and in the 272- bit-entry mode, the database size is 4K entries. The LNI7010 and LNI7020 devices are programmable to support multiple databases of varying entry sizes. The example shown in this application note is for 68-bit lookups with the NSEs configured as 16K x 68 bits. The device can easily be configured for 136-bit lookups or 272-bit lookups.

4 REFERENCES

TABLE 1. LNI8010 CONFIGURATION REGISTER

CONFIGURATION REGISTER [63:0]									
Addr	63–12	11	10	9	8	7–6	5–3	2–1	0
0–1	Reserved	External Transceiver Present	SEARCH Result Bit in Data Field	INTR_Polarity	SSRAM Present	CPCFG	HLAT	TLSZ	SRST

TABLE 2. LNI7010 COMMAND REGISTER

FIELD	RANGE	INITIAL VALUE	DESCRIPTION
SRST	[0]	0	**Software Reset.** "1" resets the device with the same effect as the hardware reset. Internally, it generates a reset pulse lasting for eight CLK cycles. This bit automatically resets to "0" when "1" is written to it.
DEVE	[1]	0	**Device Enable.** If the DEVE reads 0, it does not perform any SEARCH, WRITE, LEARN, or READ operations. It keeps the SRAM bus in a 3-state condition and forces the cascade interface outputs to 0.
TLSZ	[3:2]	01	**Table Size.** The host ASIC must program this field to configure the chips into a table of a certain size. This field affects the pipeline latency of the SEARCH and LEARN operations as well as the accesses to the SRAM (SADR [21:0], CE_L, OE_L, WE_L, ALE_L, SSV, SSF, and ACK). Once programmed, the search latency stays constant. Latency # CLK 00: 1 device 4 01: 2-8 device 5 10: 8-31 device 6 11: Reserved
HLAT	[6:4]	000	**Latency of Hit Signals.** This field adds latency to the SSF, SSV, and ACK signals by the following number of CLK cycles during SEARCH and SRAM PIO READ operations. 000: 0 100: 4 001: 1 101: 5 010: 2 110: 6 011: 3 111: 7
LDEV	[7]	0	**Last NSE in Table.** This device is the last NSE in the depth-cascaded table. In the event of a SEARCH failure, the device with this bit set drives the hit signals as follows: SSF = 0, SSV = 1. During nonSEARCH cycles, the device with this bit set drives the signals as follows. SSF = 0, SSV = 0.
LRAM	[8]	0	**Last Device on SRAM Bus.** This device is the last NSE on this SRAM bus. In cycles where an NSE does not drive the SRAM bus, the device with this bit set drivesSADR, CE_L, and WE_L in their inactive state. This bit sets a default driver for the SRAM control signals (SADR, CE_L, WE_L, and OE_L). **Note**: OE_L is always asserted or deasserted. Connect the OE_L from the device to the SRAM bus, with the LRAM bit set.
CFG	[16:9]	00000000	**Table Configuration.** The device is internally divided into four quadrants of 4K x 68 bits, each of which can be configured as 4K x 68 bits, 2K x 136 bits, or 1K x 272 bits, as follows. 00: 4K x 68 bits 01: 2K x 136 bits 10: 1K x 272 bits Bits[10:9] apply to configuring the first quadrant in the address space. Bits[12:11] apply to configuring the second quadrant in the address space. Bits[14:13] apply to configuring the third quadrant in the address space. Bits[16:15] apply to configuring the fourth quadrant in the address space.
	[67:17]	0	Reserved.

5 SAMPLE CODE FOR INITIALIZING LNI8010 AND LNI70X0 DEVICES

```
/*
The following code example uses a pointer pcp that points to the LNI8010 device.
*/

/*
Overview of the initialization procedure:
    1.  Do software reset on LNI8010 coprocessor
    2.  Disable error interrupt signals.
    3.  Set the LNI8010 coprocessor's configuration parameters
    4.  Do software reset on LNI70X0 Search Engines using a broadcast WRITE command from the LNI8010
    5.  Write the configuration parameters into LNI70X0 Search Engines' Command register using WRITE
        command(s) from LNI8010.
    6.  Write the configuration parameters of the Last LNI70X0 device's Command register using WRITE
        command from LNI8010
*/
/** 1 **/
       /* Reset LNI8010 */
       pcp->ConfigLow |= CP_RESET_BIT;      /* set reset bit in low word of config register */
       cpDelay( EIGHT_CYCLES);              /* delay for 8 cycles */

/** 2 **/
       /* Disable all interrupts */
       pcp->InterruptMaskLow = 0xffffffff;  /* set error interrupt mask lower 32 bits */
       pcp->InterruptMaskHigh = 0xffffffff; /* set error interrupt mask high 32 bits */

/** 3 **/
       /* Assemble coprocessor config parameters in a local variable configParams */
       configParams = 0;

/* The necessary parameters for Search Engines can be defined with the appropriate values */

#define NUMBER_OF_SES 4  /* Number of Search Engines is 4 in this example */
#define TLSZ    0x01
#define HLAT    0x02   /* depends on the SRAM device */
#define CPCFG   0x00
#define CP_TLSZ_SHIFT  1
#define CP_HLAT_SHIFT  3
#define CP_CPCFG_SHIFT 6
#define CP_SRAM_PRESENT              0x00000100
#define CP_INTR_POLARITY      0x00000200
#define CP_SEARCH_RESULT_IN_DATA      0x00000400
#define CP_EXTERNAL_TRANSCEIVER_PRESENT      0x00000800

      /* Set table size as would be set in Search Engine (SE)*/
        configParams |= (TLSZ << CP_TLSZ_SHIFT);
      /* Set hit latency as would be set in SE */
        configParams |= (HLAT << CP_HLAT_SHIFT);
        configParams |= (CPCFG << CP_CPCFG_SHIFT);/* not relevant for Sitera */
        configParams |= CP_SRAM_PRESENT;      /* Indicate SRAM is present */
        configParams |= CP_INTR_POLARITY;/* choose interrupt polarity as required for h/w interface */
        configParams |= CP_SEARCH_RESULT_IN_DATA;

        pcp->ConfigLow = configParams;       /* set config low word */
                                    /* the high word of config is not used */

/* Search Engine command parameters */
#define SE_RESET_BIT   0x00000001
#define SE_DEVE_BIT    0x00000002
#define SE_LDEV_BIT    0x00000080
#define SE_LRAM_BIT    0x00000100
#define SE_CFG         0x00000000
#define SE_TLSZ_SHIFT  2
#define SE_HLAT_SHIFT  4
#define SE_CFG_SHIFT   9

#define CP_ACC_LOC_SE_REG 3

#define CP_ACC_LOC_SHIFT     27
#define CP_LB1_SHIFT   12
```



```
#define CP_START_BIT   0x01000000



#define BROADCAST_ADDR 0x1f
#define SE_ID_SHIFT    19
#define SE_COMMAND_REG_ADDR 56

#define CMR_DONE_BIT   0x80000000
#define CMR_ERROR_BIT  0x20000000







/** 4 **/
/* First we write a reset to all the search engines */

        commandParams = 0;
       commandParams |= SE_RESET_BIT;       /* to do reset */

/*
Fill coprocessor command descriptor 0 with WRITE command parameters.
This WRITE will program the Command Register of all Search Engines
*/

/*
Prepare the local variables, commandDescWordLow and commandDescWordHigh ;
*/
       commandDescWordLow = 0;
       commandDescWordHigh = 0;

/*
 * For a Search Engine register write; specify command code, access location, Layer Attributes 1 bits,
and direct mode
 */
       commandDescWordLow = CP_WRITE_COMMAND; /* Set WRITE command code */

       /* Set Search Engine region for write as - RegisterRegion */
       commandDescWordLow |= (CP_ACC_LOC_SE_REG << CP_ACC_LOC_SHIFT);

       /* Place the 4 least sig. bits of the data in the “Layer Attributes 1” field */
       commandDescWordLow |= (commandParams & 0xf) << CP_LB1_SHIFT);

       /*
        * For a SE register write, SearchSuccessfulRegisterIndex,
        * GlobalMaskRegisterIndex, ComparandRegisterIndex are not relevant;
        * ProcessorId, ContextId are set to zero
        */
       commandDescWordHigh = 0;

       /*
        * We use broadcast to write into all Search Engines' command words,
        * Assemble the broadcast address of SE Command Register
        */
       broadcastAddr = (BROADCAST_ADDR << SE_ID_SHIFT) | SE_COMMAND_REG_ADDR;

/*
Fill the WRITE command parameters in ContextDescriptor 0
The pointer pctxDes0 points to the first context descriptor of LNI8010
*/
       pctxDes0 = &pcp->contextDescriptor[0];

       pctxDes0->data0Low = broadcastAddr;   /* bits 31 to 0 */
       pctxDes0->data0High = 0;                     /* bits 63 to 32 */

       /* Shift config bits and set the WRITE data */
```



```
/* The 4 least significant bits of LNI70X0 Command Register – Data, are placed in context descriptor
command word, the rest of the 64 bits are passed thru Data1 field */

        pctxDes0->data1Low = commandParams >> 4;
        pctxDes0->data1High = 0;

/* Set the Context Descriptor command words (high first and then Low, since Start bit is in Low word */

        pctxDes0->commandWordHigh = commandDescWordHigh;

/* Set start bit to initiate the WRITE command */
        pctxDes0->commandWordLow = commandDescWordLow | CP_START_BIT;

        /* Check for result: done bit should be set */

        while(!(pctxDes0->cdr0l & CMR_DONE_BIT))
            delay(); /*  delay for a few clock cycles */

        /* Result ready */

        if (pctxDes0->cdr0l & CMR_ERROR_BIT)
              process_error_and_status_register(); /* if error: view ESR reg */

 /** 5 **/
      /* Now after reset, do rest of the initialization of the SEs */

commandParams = 0;
        commandParams |= SE_DEVE_BIT; /* to enable search engine */
        commandParams |= (CP_TLSZ << SE_TLSZ_SHIFT); /* same value as in LNI8010 */
       commandParams |= (CP_HLAT << SE_HLAT_SHIFT); /* same value as in LNI8010 */
        commandParams |= (SE_CFG << SE_CFG_SHIFT);/* Quadrant configuration */

        /*
 * Fill coprocessor command descriptor, 0, with write command parameters -
         */

        /*
         * Prepare the commandDescWordLow and commandDescWordHigh as described above
         */
        commandDescWordLow = 0;
        commandDescWordHigh = 0;

        commandDescWordLow = CP_WRITE_COMMAND ; /* set WRITE command code */

        /* Set Search Engine region for write as - RegisterRegion */
        commandDescWordLow |= (CP_ACC_LOC_SE_REG << CP_ACC_LOC_SHIFT);

        /* Place the 4 least sig. bits in config word */
        commandDescWordLow |= (CP_LBITS_D1 << (commandParams & 0xf));

        /*
         * For a SE register write, SearchSuccessfulRegisterIndex,
         * GlobalMaskRegisterIndex, ComparandRegisterIndex are not relevant;
         * ProcessorId, ContextId are set to zero
         */
        commandDescWordHigh = 0;

        /*
        * We use broadcast to write into all Search Engines' command words,
         * assemble the broadcast address of SE Command Register
         */
        broadcastAddr = (BROADCAST_ADDR << SE_ID_SHIFT) | SE_COMMAND_REG_ADDR;

        /*
         * Fill the WRITE command in ContextDescriptor 0
         */

        pctxDes0 = &pcp->contextDescriptor[0];

        pctxDes0->data0Low = broadcastAddr;
        pctxDes0->data0High = 0;
```



```
        /* Shift config bits */

        pctxDes0->data1Low = commandParams >> 4; /* the 4 lsbits are placed in config word */
        pctxDes0->data1High = 0;

        pctxDes0->commandWordHigh = commandDescWordHigh;



/* Set start bit to initiate the WRITE command */
        pctxDes0->commandWordLow = commandDescWordLow | CP_START_BIT;

        /* Check for result: done bit should be set */

        while(!(pctxDes0->cdr0l & CMR_DONE_BIT))
                delay(); /*  delay for a few clock cycles */


        /* Result ready */


if (pctxDes0->cdr0l & CMR_ERROR_BIT)
             process_error_and_status_register(); /* if error : view ESR reg */

/** 6 **/
     /* Now REDO the above WRITE again only for the LAST
      Search Engine with the LDEV & LRAM bit set
     */




      pctxDes0->data0Low = ((NUMBER_OF_SEs - 1) << SE_ID_SHIFT) | SE_COMMAND_REG_ADDR;
     commandParams |= SE_LDEV_BIT;    /* set LDEV bit for last SE */
     commandParams |= SE_LRAM_BIT;    /* set LRAM bit for last SE */






     /* Initialization is complete when WRITE is done */

/* Using similar procedure:
Set the GlobalMaskRegister pair 0 with 0xffffffff's (all ones).
Use this GMR pair 0, to initialize the Mask Array with appropriate values.
Use this GMR pair 0, to initialize the Data Array with appropriate values.
Also write the SSRAM entries with appropriate values (if SSRAM present)
*/
```

CONTACT

Lara Networks, Inc.
110 Nortech Parkway
San Jose, CA 95134
www.laranetworks.com
Tel: 408 942 2035
Fax: 408 942 2099

