To our customers,

Old Company Name in Catalogs and Other Documents

On April 1st, 2010, NEC Electronics Corporation merged with Renesas Technology Corporation, and Renesas Electronics Corporation took over all the business of both companies. Therefore, although the old company name remains in this document, it is a valid Renesas Electronics document. We appreciate your understanding.

Renesas Electronics website: http://www.renesas.com

April 1st, 2010
Renesas Electronics Corporation

Issued by: Renesas Electronics Corporation (http://www.renesas.com)

Send any inquiries to http://www.renesas.com/inquiry.

Preliminary Design Manual

CB-9 Family VX/VM Type

0.35 μm CMOS Cell-Based IC (CBIC)

NB85E, NB85ET

Document No. A14335EJ3V1DM00 (3rd edition)
Date Published January 2002 NS CP(N)

[MEMO]

NOTES FOR CMOS DEVICES

① PRECAUTION AGAINST ESD FOR SEMICONDUCTORS

Note:

Strong electric field, when exposed to a MOS device, can cause destruction of the gate oxide and ultimately degrade the device operation. Steps must be taken to stop generation of static electricity as much as possible, and quickly dissipate it once, when it has occurred. Environmental control must be adequate. When it is dry, humidifier should be used. It is recommended to avoid using insulators that easily build static electricity. Semiconductor devices must be stored and transported in an anti-static container, static shielding bag or conductive material. All test and measurement tools including work bench and floor should be grounded. The operator should be grounded using wrist strap. Semiconductor devices must not be touched with bare hands. Similar precautions need to be taken for PW boards with semiconductor devices on it.

② HANDLING OF UNUSED INPUT PINS FOR CMOS

Note:

No connection for CMOS device inputs can be cause of malfunction. If no connection is provided to the input pins, it is possible that an internal input level may be generated due to noise, etc., hence causing malfunction. CMOS devices behave differently than Bipolar or NMOS devices. Input levels of CMOS devices must be fixed high or low by using a pull-up or pull-down circuitry. Each unused pin should be connected to V_{DD} or GND with a resistor, if it is considered to have a possibility of being an output pin. All handling related to the unused pins must be judged device by device and related specifications governing the devices.

③ STATUS BEFORE INITIALIZATION OF MOS DEVICES

Note:

Power-on does not necessarily define initial status of MOS device. Production process of MOS does not define the initial operation status of the device. Immediately after the power source is turned ON, the devices with reset function have not yet been initialized. Hence, power-on does not guarantee out-pin levels, I/O settings or contents of registers. Device is not initialized until the reset signal is received. Reset operation must be executed immediately after power-on for devices having reset function.

Regional Information

Some information contained in this document may vary from country to country. Before using any NEC product in your application, please contact the NEC office in your country to obtain a list of authorized representatives and distributors. They will verify:

- Device availability
- Ordering information
- Product release schedule
- Availability of related technical literature
- Development environment specifications (for example, specifications for third-party tools and components, host computers, power plugs, AC supply voltages, and so forth)
- Network requirements

In addition, trademarks, registered trademarks, export restrictions, and other legal issues may also vary from country to country.

NEC Electronics Inc. (U.S.)
Santa Clara, California
Tel: 408-588-6000
800-366-9782
Fax: 408-588-6130
800-729-9288

NEC Electronics (Europe) GmbH
Duesseldorf, Germany
Tel: 0211-65 03 01
Fax: 0211-65 03 327

• Branch The Netherlands
Eindhoven, The Netherlands
Tel: 040-244 58 45
Fax: 040-244 45 80

• Branch Sweden
Taeby, Sweden
Tel: 08-63 80 820
Fax: 08-63 80 388

NEC Electronics (France) S.A.
Vélizy-Villacoublay, France
Tel: 01-3067-58-00
Fax: 01-3067-58-99

NEC Electronics (France) S.A.
Representación en España
Madrid, Spain
Tel: 091-504-27-87
Fax: 091-504-28-60

NEC Electronics Italiana S.R.L.
Milano, Italy
Tel: 02-66 75 41
Fax: 02-66 75 42 99

NEC Electronics (UK) Ltd.
Milton Keynes, UK
Tel: 01908-691-133
Fax: 01908-670-290

NEC Electronics Hong Kong Ltd.
Hong Kong
Tel: 2886-9318
Fax: 2886-9022/9044

NEC Electronics Hong Kong Ltd.
Seoul Branch
Seoul, Korea
Tel: 02-528-0303
Fax: 02-528-4411

NEC Electronics Singapore Pte. Ltd.
Novena Square, Singapore
Tel: 253-8311
Fax: 250-3583

NEC Electronics Taiwan Ltd.
Taipei, Taiwan
Tel: 02-2719-2377
Fax: 02-2719-5951

NEC do Brasil S.A.
Electron Devices Division
Guarulhos-SP, Brasil
Tel: 11-6462-6810
Fax: 11-6462-6829

J01.12

Major Revisions in This Edition (1/2)

Pages	Description
Throughout	Deletion of descriptions on NPB peripheral macro
p.19	Modification of **Figure 2-1 Clock Controller Connection Example (Placing Oscillator Inside ASIC and Attaching Resonator Outside)**
p.25	Addition of **Figure 3-1 NB85E ROM Area Setting (1) When one compiled ROM of 32 bits × 8 Kwords is connected**
p.25	Addition of **Note** in **Figure 3-2 ROM Access Timing**
p.26	Addition of **3.2.1 When connecting one ROM**
p.26	Addition of **Figure 3-3 Example of Connecting One Compiled ROM (32 Bits × 8 Kwords) to VFB**
pp.27 to 31	Addition of **3.2.2 When connecting multiple ROMs**
p.28	Modification of **Figure 3-4 Example of Connecting Two Compiled ROMs (32 Bits × 4 Kwords) to VFB**
p.33	Modification of **Figure 3-9 RAM Access Timing**
p.34	Addition of **Figure 3-10 NB85E RAM Area Setting (2) When connecting eight compiled RAMs of 8 bits × 2 Kwords**
p.35	Modification of **Figure 3-11 Example of Connecting Compiled Memory to VDB (Connecting Four Compiled Memories)**
p.36	Modification of **Figure 3-12 Example of Connecting Compiled Memory to VDB (Connecting Eight Compiled Memories)**
pp.37 to 39	Addition of **3.3.1 Operation when eight RAMs are connected**
p.41	Modification of **Figure 4-1 Example of Connecting User Logic to VSB**
p.42	Modification of **4.1.1 Overview of VSB operation**
p.44	Modification of **Figure 4-4 VSB Timing (Address Hold)**
p.51	Modification of **4.2 Connection of Compiled Memory**
p.56	Modification of description in **CHAPTER 5 CONNECTION OF MEMORY CONTROLLER (MEMC)**
p.57	Modification of **Figure 5-1 Example of Connecting NB85E, MEMC, and External Memory (SRAM, SDRAM)**
p.58	Modification of **Figure 5-2 Example of Connection to SRAM**
pp.50 to 53 in 2nd edition	Deletion of **5.1 (2) Register example** and **(3) Operation timing example**
pp.55, 56 in 2nd edition	Deletion of **5.2 (2) Register example** and **(3) Operation timing example**
p.60	Modification of **5.3 Connection to SDRAM**
p.61	Modification of **Figure 5-4 Example of Connection to SDRAM**
pp.59 to 65 in 2nd edition	Deletion of **5.3 (2) Register example** and **(3) Operation timing example**
p.63	Modification of peripheral I/O area in **6.2 (1) Register mapping**
p.64	Modification of **Figure 6-3 Example of Address Decoder HDL Creation**
p.65	Modification of **Figure 6-4 Example of User Logic HDL Creation**

Major Revisions in This Edition (2/2)

Pages	Description
p.67	Modification of **Figure 6-6 Example of HDL Creation for User Logic with Retry Function**
p.69	Modification of **Figure 7-1 Example of Connecting Instruction Cache (NB85E213) to NB85E**
p.71	Addition of **7.1.1 (7) Initial program settings**
p.71	Addition of **7.1.1 (8) Setting BHC register of NB85E**
p.71	Addition of **7.1.1 (9) Test bus auto wiring tool support**
p.72	Addition of **7.1.1 (10) Tag clear procedure**
p.73	Modification of **Figure 7-3 Example of Connecting Data Cache (NB85E263) to NB85E**
p.74	Addition of **7.2.1 (5) Test bus auto wiring tool support**
p.74	Addition of **7.2.1 (6) Other**
p.75	Addition of **7.2.1 (7) Operation during debugging**
p.79	Modification of **Figure 8-4 Circuit Example for In-Circuit Emulator Connection (NB85E + RCU (NB85E901))**
p.80	Modification of **Figure 8-5 Circuit Example for In-Circuit Emulator Connection (NB85ET)**
p.81	Addition of **8.2.4 Design of timing with N-wire type in-circuit emulator**
p.86	Modification of **Figure 9-2 Skew or Hold Violation Countermeasure Execution Example**
p.90	Modification of **9.5 (2) Product specifications**
p.92	Modification of **Figure 10-1 Example of Connecting Peripheral Macro to NB85E**
p.94	Modification of **10.2.1 (1) (b) Input pin processing**
p.94	Modification of **10.2.1 (1) (c) Output pin processing**
p.95	Modification of **10.2.1 (2) Pins for test mode**
p.95	Modification of **10.2.1 (3) Cautions when using NB85E901 (RCU)**
p.95	Modification of **10.2.1 (4) Cautions when using NB85ET**
pp.98 and 99	Addition of **10.3 Test Bus Auto Wiring**
pp.100 to 105	Addition of **10.4 Test Bus PINF File Creation/Editing Methods**
p.106	Addition of description in **CHAPTER 11 (1) Creation of test patterns for checking connections**
p.112	Addition of **APPENDIX REVISION HISTORY**

The mark ★ shows major revised points.

PREFACE

Target Readers This manual is intended for users who design ASICs using NEC Corporation's "CB-9 Family VX/VM Type" high-speed, high-integration CMOS CBIC.

Purpose The purpose of this manual is to give users an understanding of design methods and various restrictions and cautions specific to designing ASICs that have on-chip NB85E or NB85ET 32-bit microprocessor cores.

Be sure to observe the points (general information, cautions, and restrictions) described in this design manual. Failure to do so may lower the quality or performance of the ASIC or lead to malfunctions. Note, however, that these points do not necessarily guarantee a circuit, and necessary functions may not be satisfied as a result of placement and routing. Therefore, be sure to perform development after verifying operation.

Organization This manual is roughly organized into the following sections.

- Connection of clock controller
- Connection of ROM/RAM to VFB/VDB
- Connection to VSB
- Connection of memory controller
- Connection to NPB
- Cache connection
- Connection to in-circuit emulator (IE)
- Cautions
- Test circuit design
- Total chip simulation

How to Use This Manual Note in this manual apply to both the NB85E and NB85ET, but the NB85E is used as the representative microprocessor core (CPU core) unless specifically stated otherwise. When using the NB85ET, read and modify the CPU core name and some of the pin names according to the following table (pin functions are the same in both products).

Item	Using NB85E[Note 1] (Names used in this manual)	Using NB85ET (Names must be changed to the following)
CPU core	NB85E	NB85ET
Pin name	DCRESZ[Note 2]	RESETZ
	DCSTOPZ	STOPZ
	DCNMI2 to DCNMI0	NMI2 to NMI0

Notes 1. Includes systems in which the NB85E901 (run control unit (RCU)) is connected.

2. There is a pin of the same name but with a different function in the NB85ET. Be careful when reading and changing names.

Before using this manual, be sure to read the separate "**CB-9 Family VX/VM Type Design Manual (A12745E)**".

Conventions

Data significance:	Higher digits on the left and lower digits on the right
Active low representation:	xxxZ (Z after pin or signal name)
	or xxxB (B after pin or signal name)
Note:	Footnote for item marked with Note in the text
Caution:	Information requiring particular attention
Remark:	Supplementary information
Numerical representation:	Binary ... xxxx or xxxxB
	Decimal ... xxxx
	Hexadecimal ... xxxxH

Prefix indicating the power of 2 (address space, memory capacity):

K (kilo) ... $2^{10} = 1024$
M (mega) ... $2^{20} = 1024^2$
G (giga) ... $2^{30} = 1024^3$

Related Documents

The related documents indicated in this publication may include preliminary versions. However, preliminary versions are not marked as such.

- CB-9 Family VX/VM Type Design Manual (A12745E)
- CB-9 Family VX/VM Type Core Library CPU Core, Memory Controller Design Manual (A13195E)
- CB-9 Family VX/VM Type Memory Macro (Compiled Type) Design Manual (A12982E)
- NEC SYSTEM LSI DESIGN OPENCAD™ V5.4 OPC_VSHELL User's Manual (A15050E)
- NEC SYSTEM LSI DESIGN OPENCAD V5.4 Verilog-XL™ Interface User's Manual (A15052E)
- NEC SYSTEM LSI DESIGN OPENCAD V5.4 Design Compiler™ Interface User's Manual (A15058E)
- NB85E Hardware User's Manual (A13971E)
- NB85ET Hardware User's Manual (A14342E)
- Memory Controller NB85E, NB85ET User's Manual (A14206E)
- Instruction Cache, Data Cache NB85E, NB85ET User's Manual (A14247E)
- IE-V850E-MC, IE-V850E-MC-A User's Manual (U14487E)
- IE-V850E-MC-EM1-B, IE-V850E-MC-MM2 User's Manual (U14482E)

The related documents listed above are subject to change without notice. Be sure to use the latest version of each document for designing.

CONTENTS

LIST OF FIGURES (1/3)

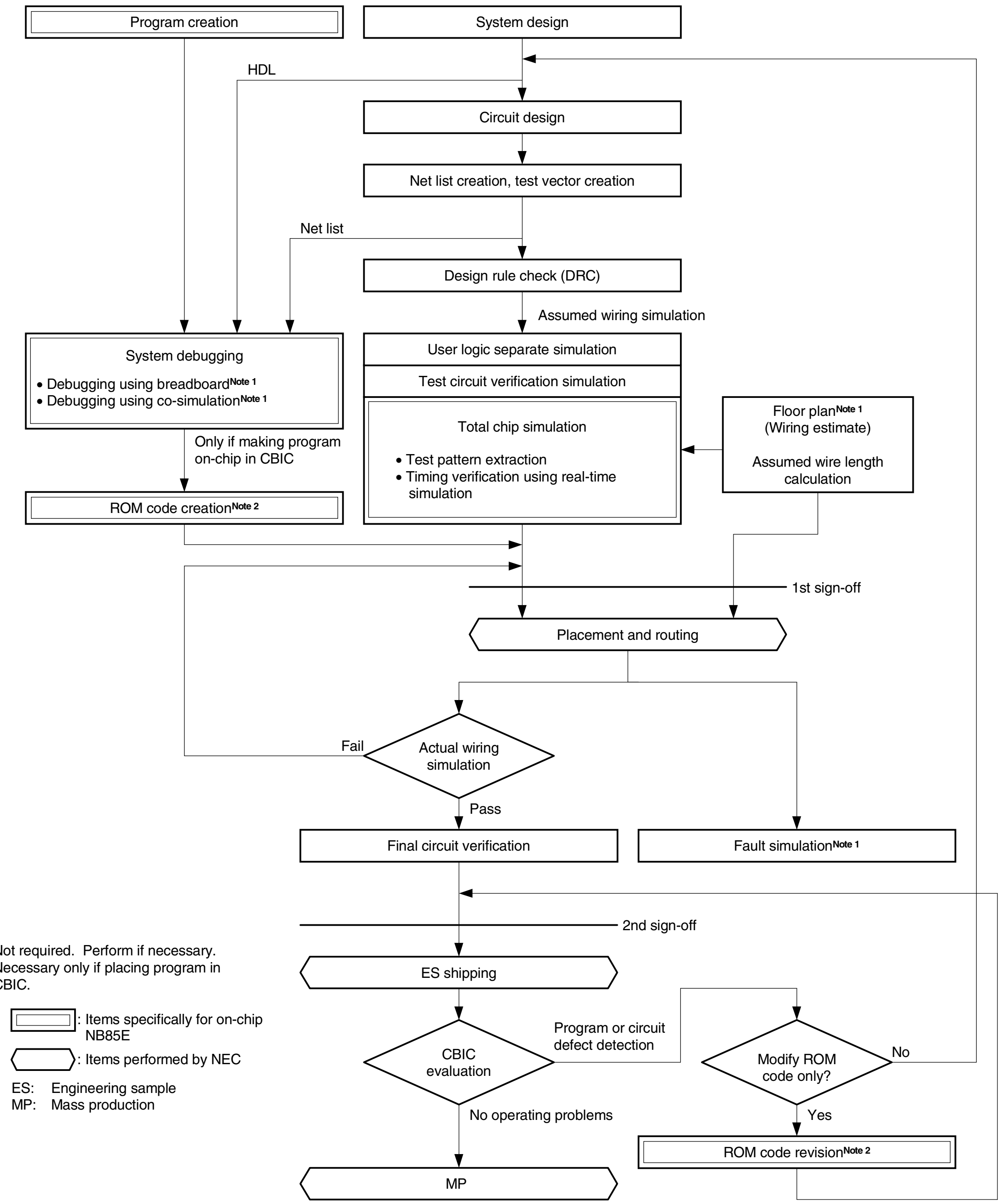
LIST OF FIGURES (2/3)

LIST OF FIGURES (3/3)

Figure No.	Title	Page

LIST OF TABLES

1.1 Design Flow for CBIC with On-Chip NB85E



Notes **1.** Not required. Perform if necessary.
2. Necessary only if placing program in CBIC.

Remarks 1. (double-framed box): Items specifically for on-chip NB85E
(hexagonal box): Items performed by NEC

2. ES: Engineering sample
MP: Mass production

1.2 Items Specifically for On-Chip NB85E

(1) Total chip simulation

Total chip simulation is simulation that checks the connections between macros by operating the NB85E.
See **CHAPTER 11 TOTAL CHIP SIMULATION** for the simulation method and cautions.

(2) System debugging

System debugging debugs the entire target system, including the CBIC under development.

(a) Debugging using breadboard

<1> Debugging using in-circuit emulator for NB85E (IE-V850E-MC-A)

Debug the target system by creating a breadboard using a general-purpose LSI, FPGA, and G/A.
See **8.1 Using In-Circuit Emulator for NB85E (IE-V850E-MC-A)** regarding connecting to the IE-V850E-MC-A.

<2> Debugging using N-Wire type in-circuit emulator (IE-70000-MC-NW-A)

Since the NB85ET has a debug control unit (DCU) that supports an on-chip debug function, real-time debugging using an ES is possible by connecting to the IE-70000-MC-NW-A via a dedicated debug interface based on JTAG.
Similar debugging can be performed in the case of an NB85E if an NB85E901 (run control unit (RCU)) is connected.

Debug	NB85ET	NB85E + NB85E901
Break	√	√
Event detection	√	–
Trace	√	–

See **8.2 Using N-Wire Type In-Circuit Emulator (IE-70000-MC-NW-A)** regarding connecting to the in-circuit emulator.

(b) Debugging using co-simulation

Debug the target system using co-simulation.
Normally, integrated debugging of hardware and software using an in-circuit emulator is not possible before ES completion. However, co-simulation makes possible target-less debugging of both hardware and software at the system design stage before ES completion.
Contact NEC for details.

(3) ROM code creation and revision

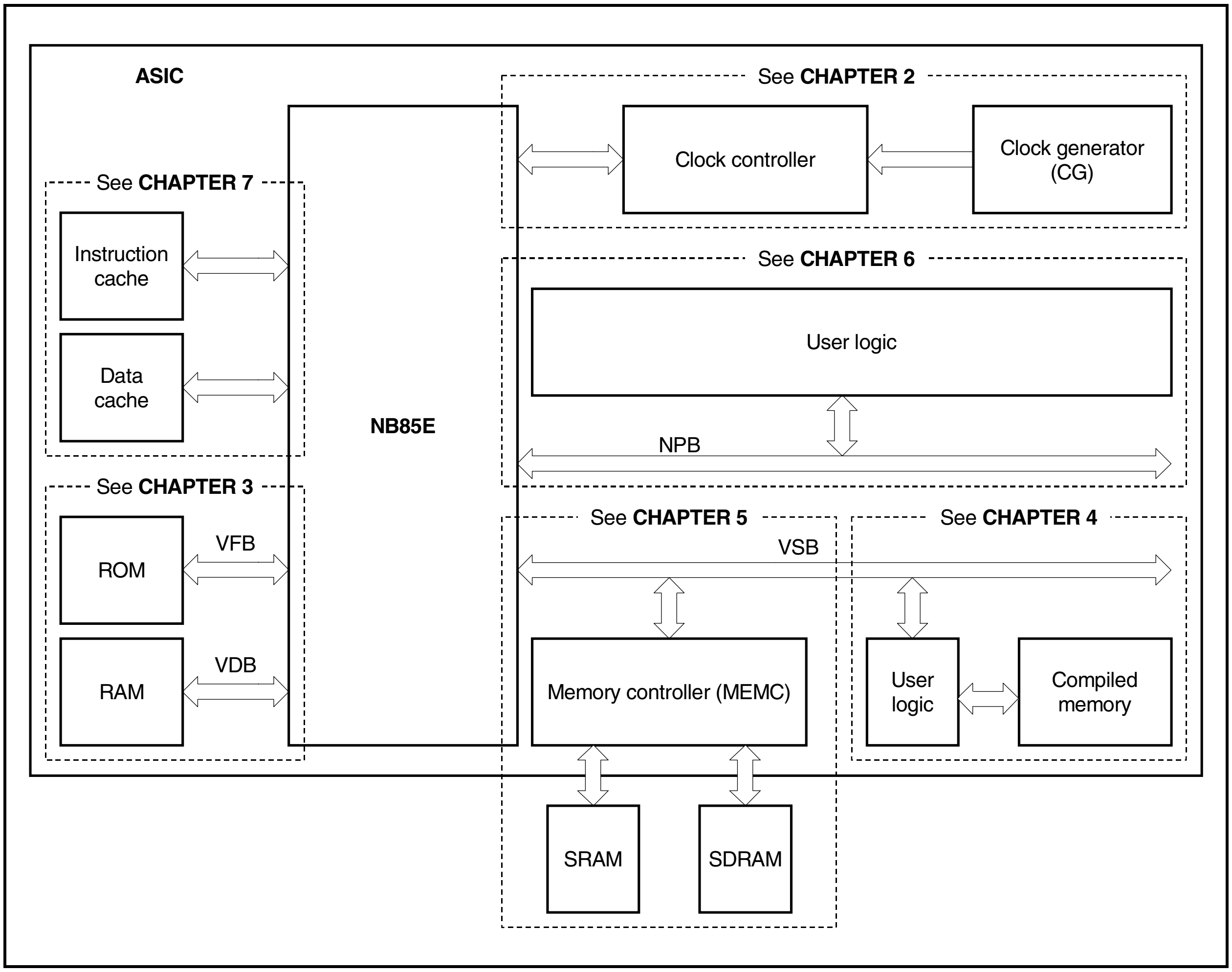
Sign-off to NEC must be performed using a specific format.
See **CHAPTER 12 ROM CODE CREATION** regarding the format.

1.3 System Example

In this manual, the ASIC shown below is used as an example and explained by focussing on the method of connecting each peripheral macro to the NB85E.

Figure 1-1. System Example

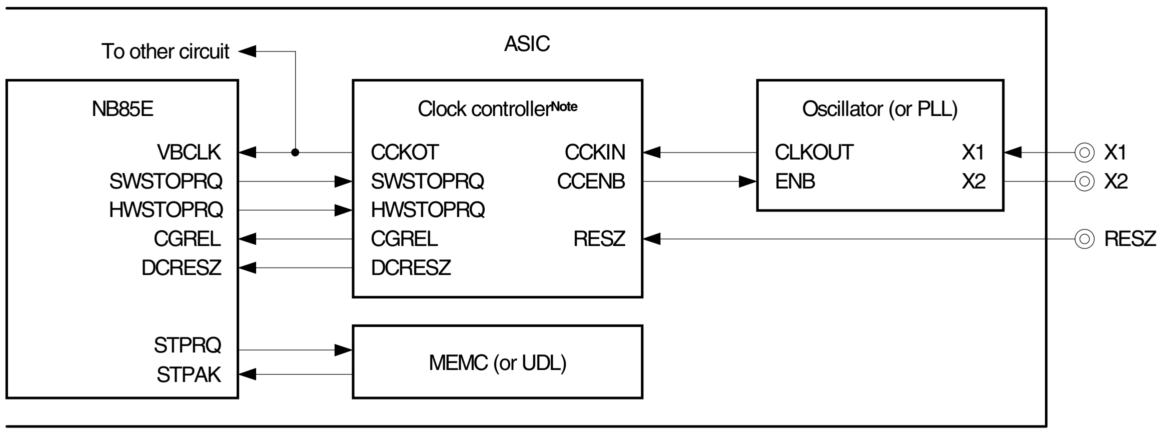
★



CHAPTER 2 CONNECTION OF CLOCK CONTROLLER

When using the NB85E software or hardware STOP mode, connect a clock controller between the oscillator (or external input clock) and NB85E or between the oscillator (or external input clock) and an other circuit related to VBCLK so that sufficient oscillation stabilization time can be guaranteed, allowing a stable clock to be supplied to the NB85E.

★ **Figure 2-1. Clock Controller Connection Example (Placing Oscillator Inside ASIC and Attaching Resonator Outside)**



Note Design the clock controller as user logic. Be sure to also include a circuit to secure the oscillation stabilization time (refer to **Figure 2-2**).

Remark CCKOT: Clock output
CCKIN: Clock input
CCENB: Oscillator control (Oscillate on low-level output)

The oscillator oscillates when a low level is output from the CCENB pin of the clock controller. Oscillation stabilization time is counted by the counter register (CNTR) inside the clock controller. The standard value of CNTR is determined using the following relational expression.

CNTR >> Oscillation stabilization time [s] × Clock frequency [Hz]

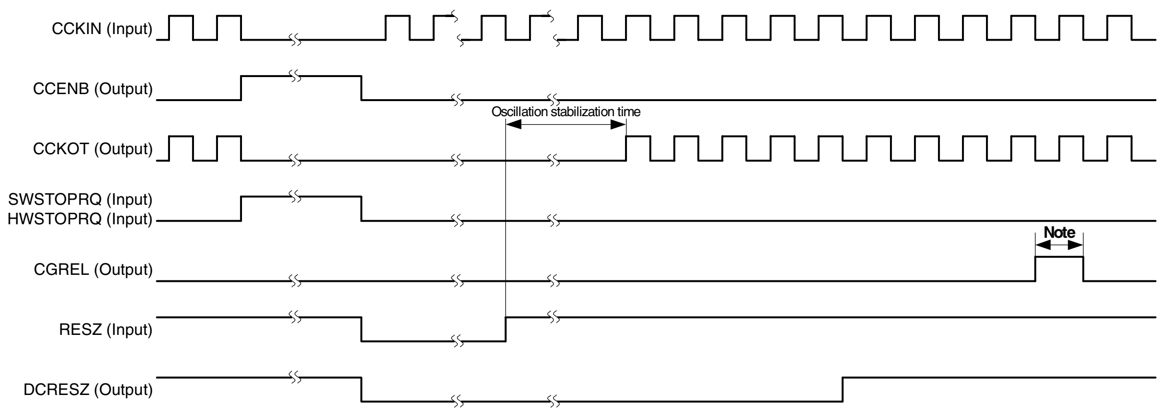
Example Using an oscillator for which the output clock frequency is 20 MHz and the oscillation stabilization time is 1 ms:
CNTR >> $1 \times 10^{-3} \times 20 \times 10^{6}$ = 20000 (4E20H)

Caution If the STPAK pin of the NB85E is unused, input a high level to the STPAK pin. If a high level is not input, the HWSTOPRQ signal and SWSTOPRQ signal do not become active and STOP mode cannot be entered.

Figure 2-2 shows the timing chart of each signal of the clock controller in Figure 2-1 and Figure 2-3 shows an HDL creation example.

Figure 2-2. Clock Controller Timing Chart

(a) Canceling STOP mode by DCRESZ input

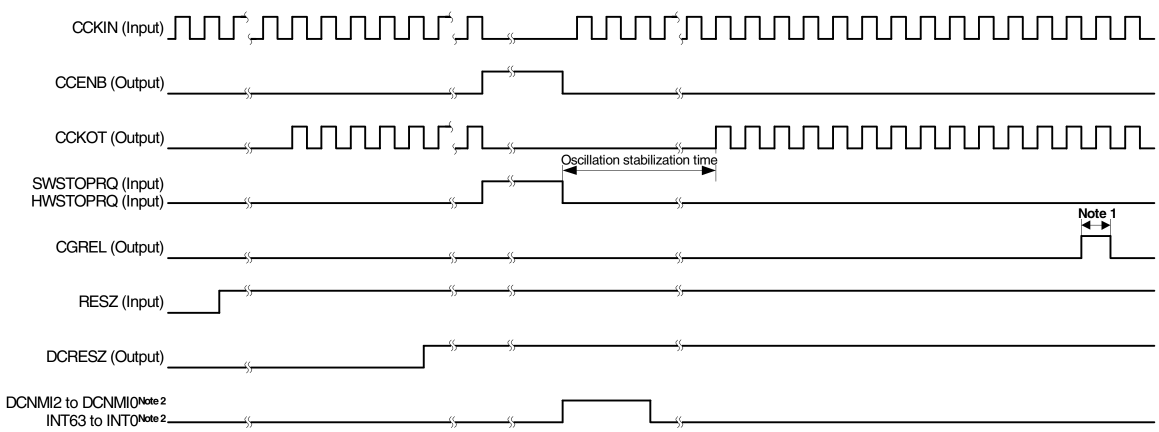


Note Always return to low level after one or more clocks of high-level output.

Remarks 1. Refer to the **NB85E Hardware User's Manual (A13971E)** regarding the relationship to the NB85E.

2. Input and output are shown as seen from the clock controller side.

(b) Canceling STOP mode by interrupt request



Notes 1. Always return to low level after one or more clocks of high-level output.

2. These are NB85E pins.

Remarks 1. Refer to the **NB85E Hardware User's Manual (A13971E)** regarding the relationship to the NB85E.

2. Input and output are shown as seen from the clock controller side.

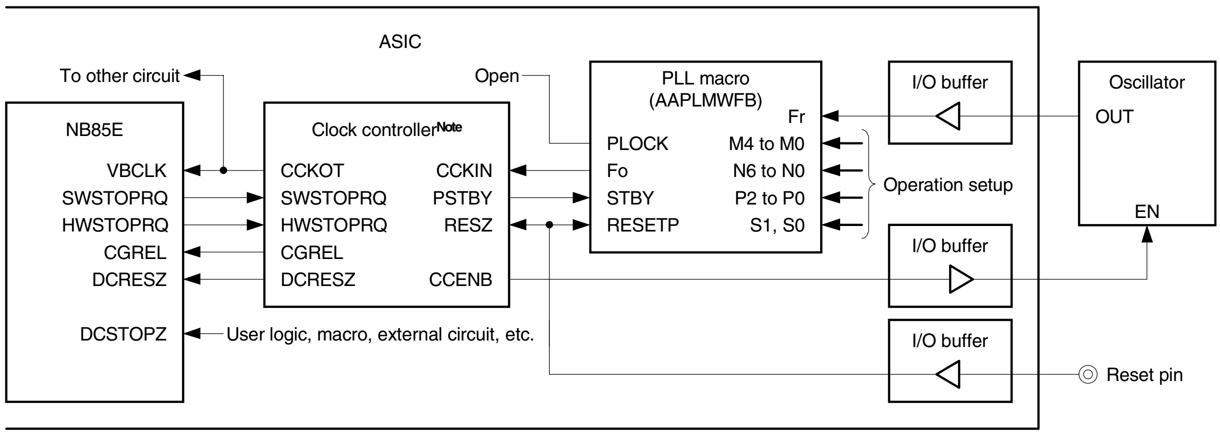
In the example in Figure 2-3, the reset input of the NB85E (DCRESZ) is also controlled. This is in order to reliably supply a stable clock during a reset period. The DCRESZ signal rises after the oscillation stabilization time has been secured following the rise of the RESZ signal (system reset).

Figure 2-3. Example of Clock Controller HDL Creation

```
module CLKCTL(
        CCKOT, SWSTOPRQ, HWSTOPRQ, CGREL,
        CCKIN, CCENB, DCRESZ, RESZ
        ) ;

        output          CCKOT ;         // Output to VBCLK pin of NB85E
        input           SWSTOPRQ ;      // Input from SWSTOPRQ pin of NB85E
        input           HWSTOPRQ ;      // Input from HWSTOPRQ pin of NB85E
        output          CGREL ;         // Output to CGREL pin of NB85E
        input           CCKIN ;         // Input from CLKOUT pin of oscillator (or PLL)
        output          CCENB ;         // Output to ENB pin of oscillator (or PLL)
        output          DCRESZ ;        // Output to DCRESZ pin of NB85E
        input           RESZ ;          // System reset input

        reg [7:0] CNTR ;                // Oscillation stabilization time counter.  The bit width is adjusted in accordance with the oscillation stabilization time.
        reg             CKOTEN ;        // Clock output enable
        reg             DCRESZ ;        // NB85E reset

        wire            STOPRQZ = ~ (SWSTOPRQ | HWSTOPRQ) ;

        wire            CGREL = ( CNTR == 8'hFE ) ;     // CGREL output timing setting
        wire            CCKOT = CKOTEN & CCKIN ;
        wire            CCENB = ~STOPRQZ ;

        // synopsys async_set_reset "DCRESZ"
        always @( negedge CCKIN or negedge STOPRQZ or negedge RESZ ) begin
                if ( ~RESZ ) begin
                        CNTR            <= 8'h00 ;
                        CKOTEN          <= 1'b0 ;
                        DCRESZ          <= 1'b0 ;
                end
                else begin
                        if( ~STOPRQZ ) begin
                                CNTR            <= 8'h00 ;
                                CKOTEN          <= 1'b0 ;
                        end
                        else begin
                                if ( CNTR == 8'hF0 ) CKOTEN     <= 1'b1 ;       // Clock output timing setting.  The oscillation stabilization time is F0H.
                                if ( CNTR == 8'hF5 ) DCRESZ     <= 1'b1 ;       // DCRESZ output timing setting
                                if ( CNTR != 8'hFF ) CNTR       <= CNTR + 1 ;   // Internal counter stop timing setting
                        end
                end
        end

endmodule
```

When using a PLL macro, the configuration is as shown in Figure 2-4.

Figure 2-4. Clock Controller Connection Example (Using PLL Macro)



Note This differs from the clock controller shown in Figure 2-1.

The PLL macro and clock controller in Figure 2-4 are described below.

(1) PLL macro

"AAPLMWFB" is taken as an example (contact NEC for details about macros). The input clock corresponds to 5 MHz to 160 MHz, and the output clock to 50 MHz to 250 MHz.

In Figure 2-4, if each of pins M4 to M0, N6 to N0, and P2 to P0, which set counter values, and each of pins S1 and S0, which set VCO range setting switching, is set as follows, when a 16.384 MHz clock is input to the Fr pin, a 49.152 MHz (50% duty) signal is obtained from the Fo pin.

- M4 to M0: L, L, L, L, H
- N6 to N0: L, L, L, L, L, H, H
- P2 to P0: L, L, H
- S1, S0: L, L

Remark L: Low-level input
H: High-level input

(2) Clock controller

A control pin (PSTBY) must be added for setting the PLL to standby mode when the controller shown in Figure 2-1 is in STOP mode.

In addition, if both the PLL (t_{WPLL}) and the OSC (t_{WOSC}) are made to oscillate, the oscillation stabilization time must take into account the time interval $t_{WPLL}+t_{WOSC}$.

CHAPTER 3 CONNECTION OF ROM/RAM TO VFB/VDB

3.1 Overview

The NB85E provides a V850E fetch bus (VFB) that makes direct connection of ROM possible and a V850E data bus (VDB) that makes direct connection of RAM possible.

The VFB is a bus optimized for fetching instructions, and the VDB is a bus optimized for accessing data.

Since these buses are independent of the VSB, instructions can be fetched, or data accessed, even if the VSB is occupied by another macro.

3.2 Connection of Compiled Memory to VFB

The VFB is a bus for fixed 32-bit access instruction fetches that is optimized for data access and by which one clock access is possible.

When a high level is input to the IFIROME pin, an instruction fetch from a ROM connected to the VFB is possible.

The compiled ROM that can be used by the CB-9 Family VX/VM Type is fast synchronous ROM.

When connecting compiled ROM to the VFB, it is necessary to compute the relationship of the ROM access time and cycle time to the CPU clock frequency and check that the timing fits adequately (see **9.4 Timing Adjustment**).

Figure 3-1 shows the NB85E ROM area setting, Figure 3-2 shows the ROM access timing, Figure 3-3 shows an example in which one compiled ROM of 32 bits × 8 Kwords for the CB-9 Family VX Type is connected, and Figure 3-4 shows an example in which two compiled ROMs of 32 bits × 4 Kwords for the CB-9 Family VX Type are connected.

Remarks 1. For details about compiled memory, refer to the **CB-9 Family VX/VM Type Memory Macro (Compiled Type) Design Manual (A12982E)**.

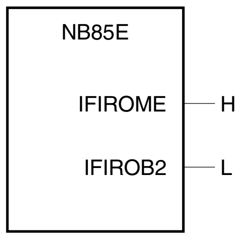
2. For the creation method of compiled memory, refer to **NEC SYSTEM LSI DESIGN OPENCAD V5.4 OPC_VSHELL User's Manual (A15050E).**

3. When using compiled memory, test wiring is needed in order to use the ASIC standard test procedure. Refer to the **CB-9 Family VX/VM Type Design Manual (A12745E)** regarding the test procedure.

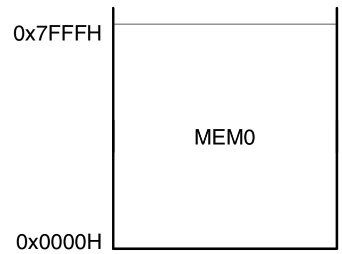
Figure 3-1. NB85E ROM Area Setting

★ **(1) When one complied ROM of 32 bits × 8 Kwords is connected**

(a) Pin settings



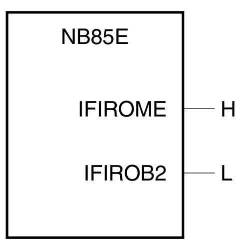
(b) Memory map



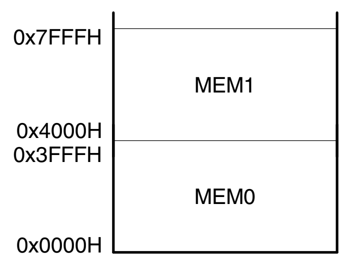
Remark H: High level input
L: Low level input

(2) When two compiled ROMs of 32 bits × 4 Kwords are connected

(a) Pin settings

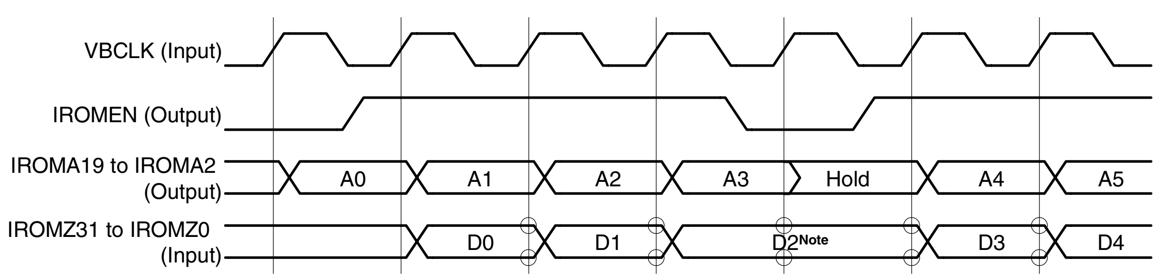


(b) Memory map



Remark H: High level input
L: Low level input

Figure 3-2. ROM Access Timing



★ **Note** Hold the data from the time the IROMEN output becomes high level until the VBCLK signal rises.

Remarks 1. Ax: Arbitrary address
Dx: Data corresponding to address "Ax"

2. The circles indicate the ROM data sampling timing.

3. Input and output are shown as seen from the NB85E.

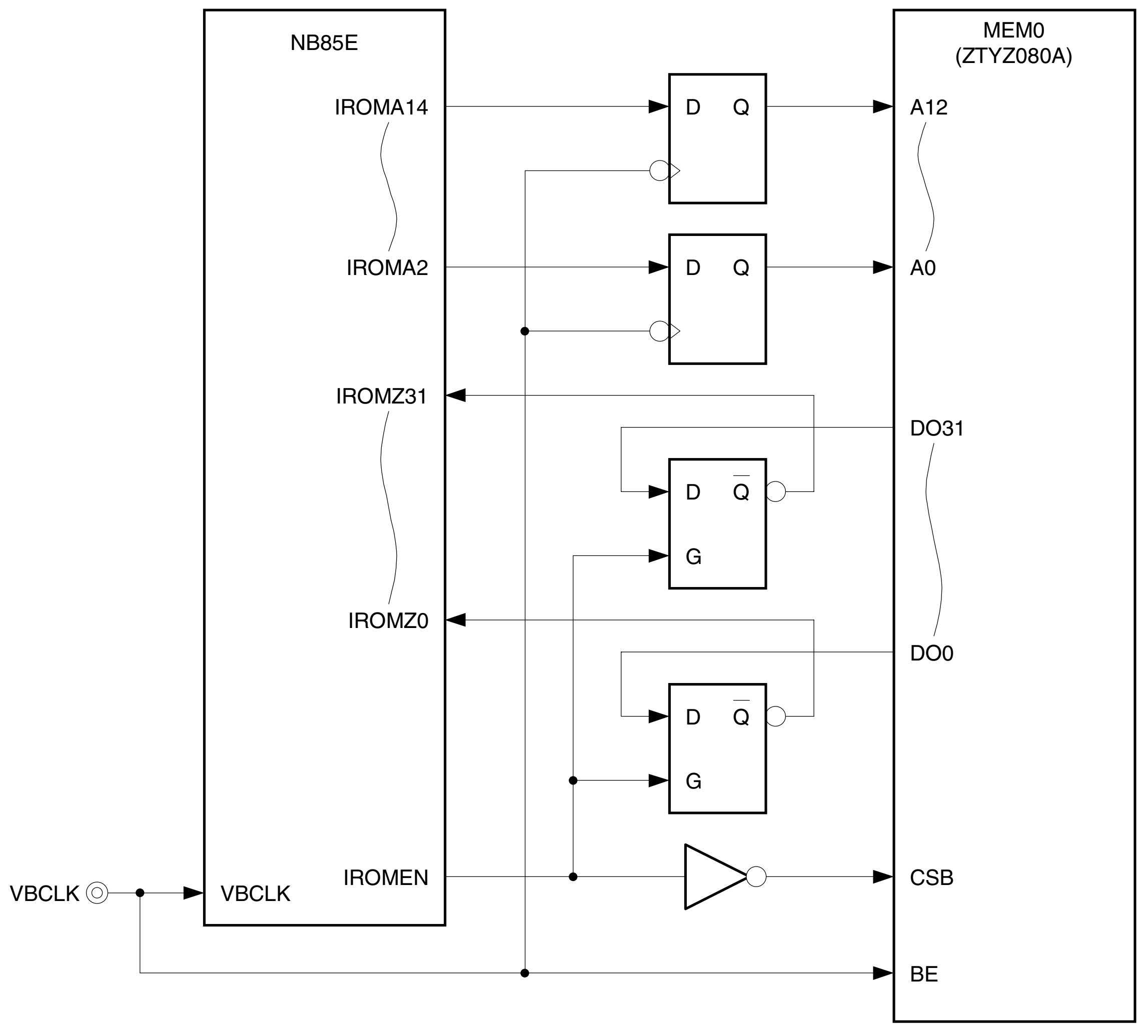
★ 3.2.1 When connecting one ROM

Figure 3-3 shows an example of connecting one compiled ROM of 32 bits × 8 Kwords for the CB-9 Family VX Type.

The operation of the VFB of the NB85E is of the pipeline type, whereby an address and the data that corresponds to that address is shifted by only 1 cycle. Here, to facilitate securing of the address setup/hold time, a flip-flop is inserted between the IROMA14 to IROMA2 pins of the NB85E and the A12 to A0 pins of the compiled ROM. Also, since data is held while the IROMEN is low level, a level latch is inserted between the DO31 to DO0 pins of the compiled ROM and the IROMZ31 to IROMZ0 pins of the NB85E.

The compiled ROM addresses are allocated to 0x00000H to 0x07FFFH.

★ **Figure 3-3. Example of Connecting One Compiled ROM (32 Bits × 8 Kwords) to VFB**



Remarks 1. Since the data bus inverts the actual data logic, be careful when dumping data during simulation.

2. The figure shows the normal pin wiring method. Wire test pins in accordance with the ASIC test procedure (for details, refer to the **CB-9 Family VX/VM Type Design Manual (A12745E)**).

★ ### 3.2.2 When connecting multiple ROMs

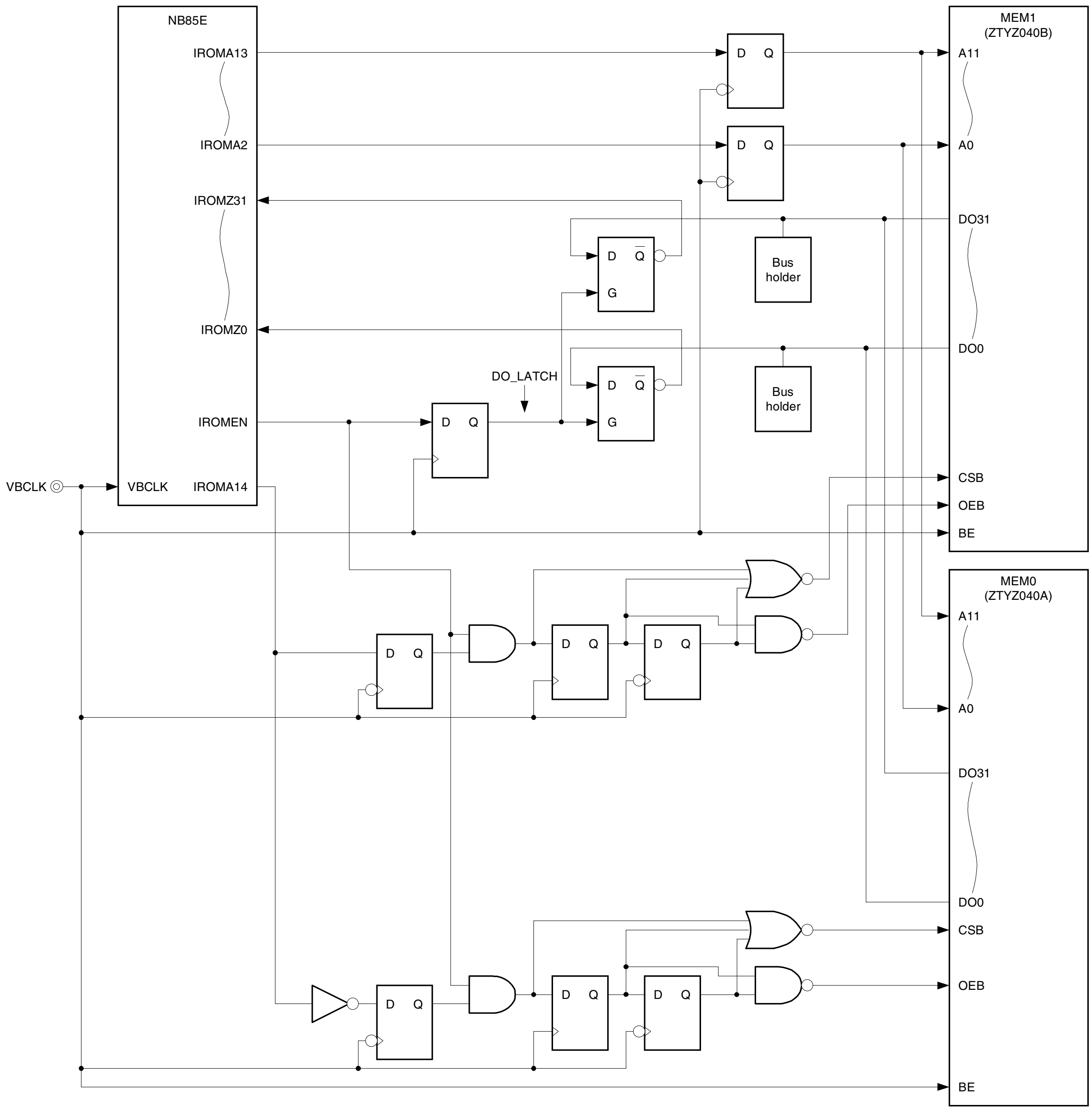
Figure 3-4 shows an example of connecting two compiled ROMs of 32 bits × 4 Kwords for the CB-9 Family VX Type. For the ROM area memory map, refer to **Figure 3-1 (2) When two compiled ROMs of 32 bits × 4 Kwords are connected**.

When multiple ROMs (for example MEM0 and MEM1) are connected, normal read may be prevented because the CSB changes to inactive level prior to the completion of data read from ROM during continuous fetching for the area spanning MEM0 and MEM1 or during branching between MEM0 and MEM1.

To prevent this, in the connection example shown in Figure 3-4, a circuit that adjusts the CSB change timing during area changes is inserted between the IROMEN pin of the NB85E and the CSB pin of the compiled ROM. This circuit also functions as an address decoder between MEM0 and MEM1.

The compiled ROM addresses are allocated to 0x0000H to 0x3FFFH for MEM0 and 0x4000H to 0x7FFFH for MEM1.

★ **Figure 3-4. Example of Connecting Two Compiled ROMs (32 Bits × 4 Kwords) to VFB**



Remarks 1. Since the data bus inverts the actual data logic, be careful when dumping data during simulation.

2. The figure shows the normal pin wiring method. Wire test pins in accordance with the ASIC test procedure (for details, refer to the **CB-9 Family VX/VM Type Design Manual (A12745E)**).

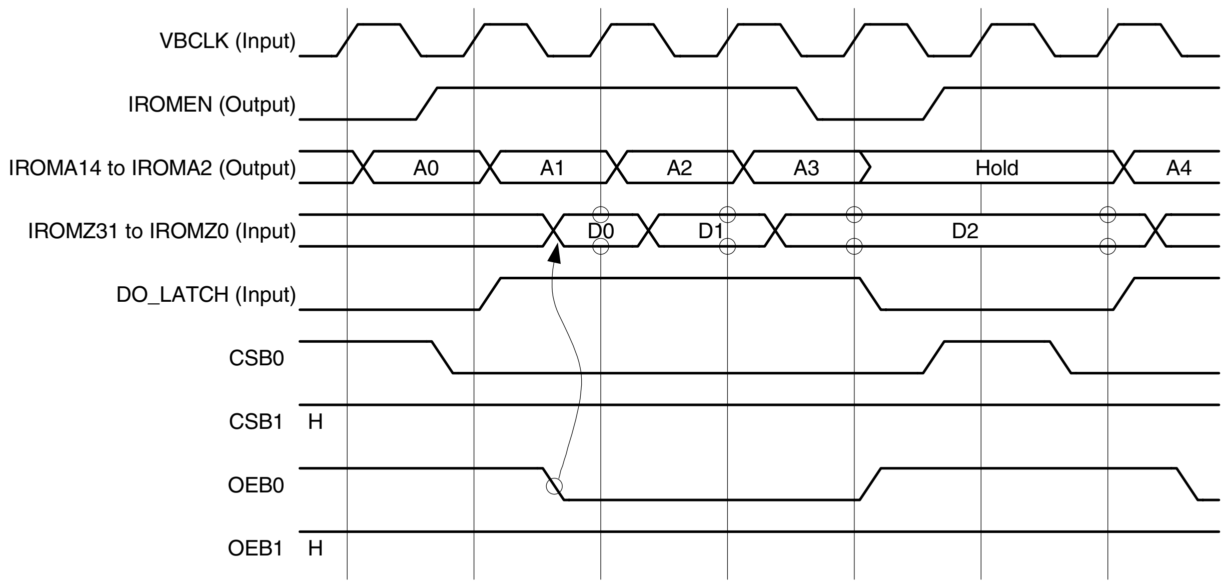
(1) ROM read from the same MEM area

When the result of decoding addresses is MEM0 area (in Figure 3-4, IROMA14 is low level), CSB0 becomes active as soon as IROMEN becomes active. The address on the MEM0 side is latched at the rising edge of the next clock after CSB0 becomes active, OEB0 becomes active at the next falling edge, and the valid data is input to IROMZ31 to IROMZ0.

At the next rising edge after IROMEN becomes inactive, at the same time as OEB0 becomes inactive, IROMZ31 to IROMZ0 hold the value at that time. This is because the values are held without performing ROM read while IROMEN is inactive, through level latching between D0 and IROMZ31 to IROMZ0. At the next falling edge of the clock after OEB0 becomes inactive, CSB0 becomes inactive.

Figure 3-5 shows the timing when the MEM0 area is accessed continually.

Figure 3-5. ROM Read Timing (For Same MEM Area)



Remarks 1. Ax: Arbitrary address

Dx: Data corresponding to address “Ax”

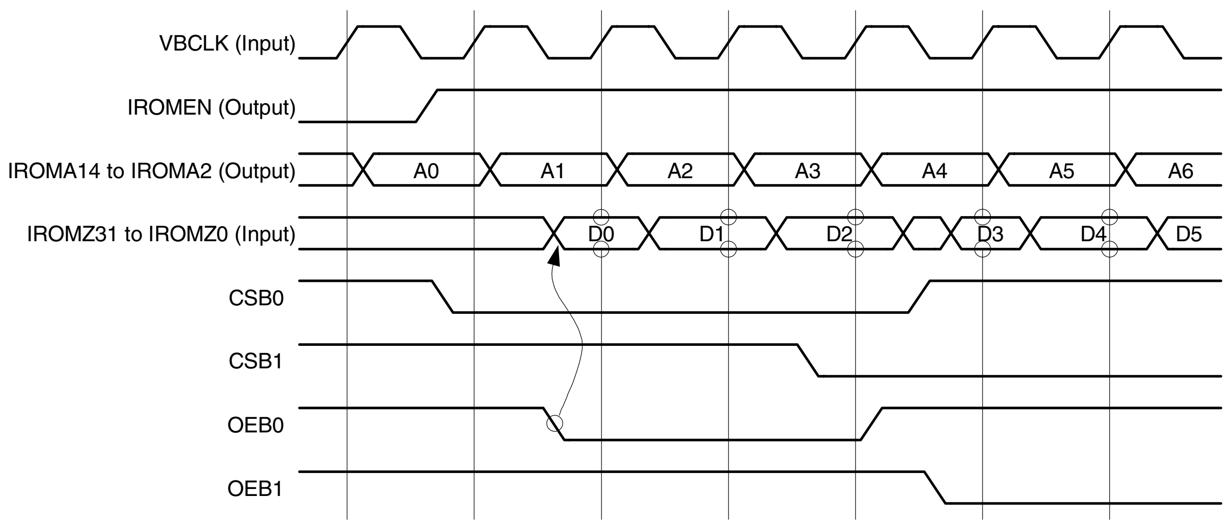
2. The circles indicate the ROM data sampling timing.

(2) ROM read when MEM area has changed

Figure 3-6 shows the CSB0, CSB1, OEB0, and OEB1 timings during continuous access to A0 to A2 (MEM0 area) and A3 to A6 (MEM1 area).

Even if IROMA14 to IROMA2 become A3 and the ROM address changes from the MEM0 area to the MEM1 area, since the data (D2) read timing for A2 is one clock later, at the rising edge of VCLK, OEB0 is extended by 1 clock, and CSB0 by 1.5 clocks. CSB1 is made active at the falling edge of A3 for preparation to read D3. To avoid data conflict between MEM0 and MEM1, OEB1 becomes active 0.5 clocks after OEB0 becomes inactive.

Figure 3-6. ROM Read Timing (When MEM Area Has Changed)



Remarks 1. Ax: Arbitrary address

Dx: Data corresponding to address “Ax”

2. The circles indicate the ROM data sampling timing.

Figure 3-7 shows the CSB0, OEB0 generation block, and Figure 3-8 shows the CSB0 and OEB0 timings. (<a> to <c> in Figure 3-7 correspond to <a> to <c> on the MEM0 side in Figure 3-8.)

Figure 3-7. CSB, OEB Generation Block

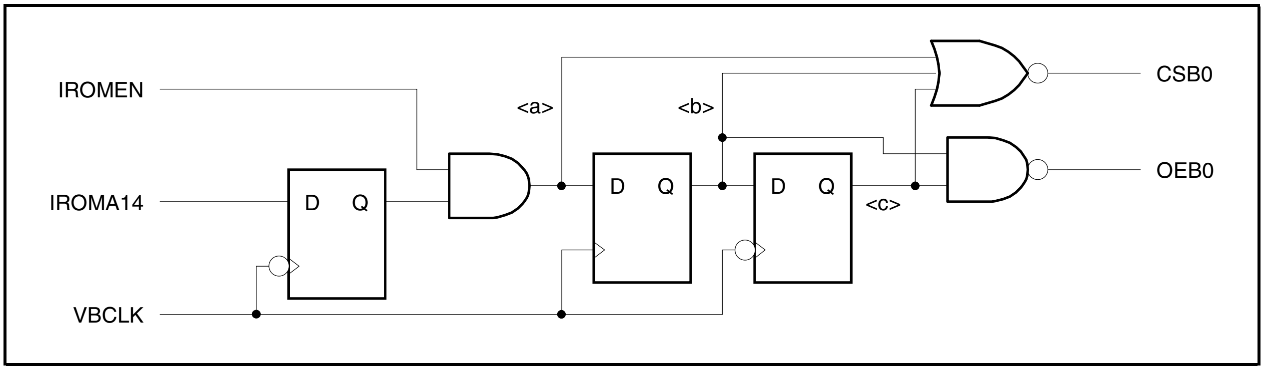
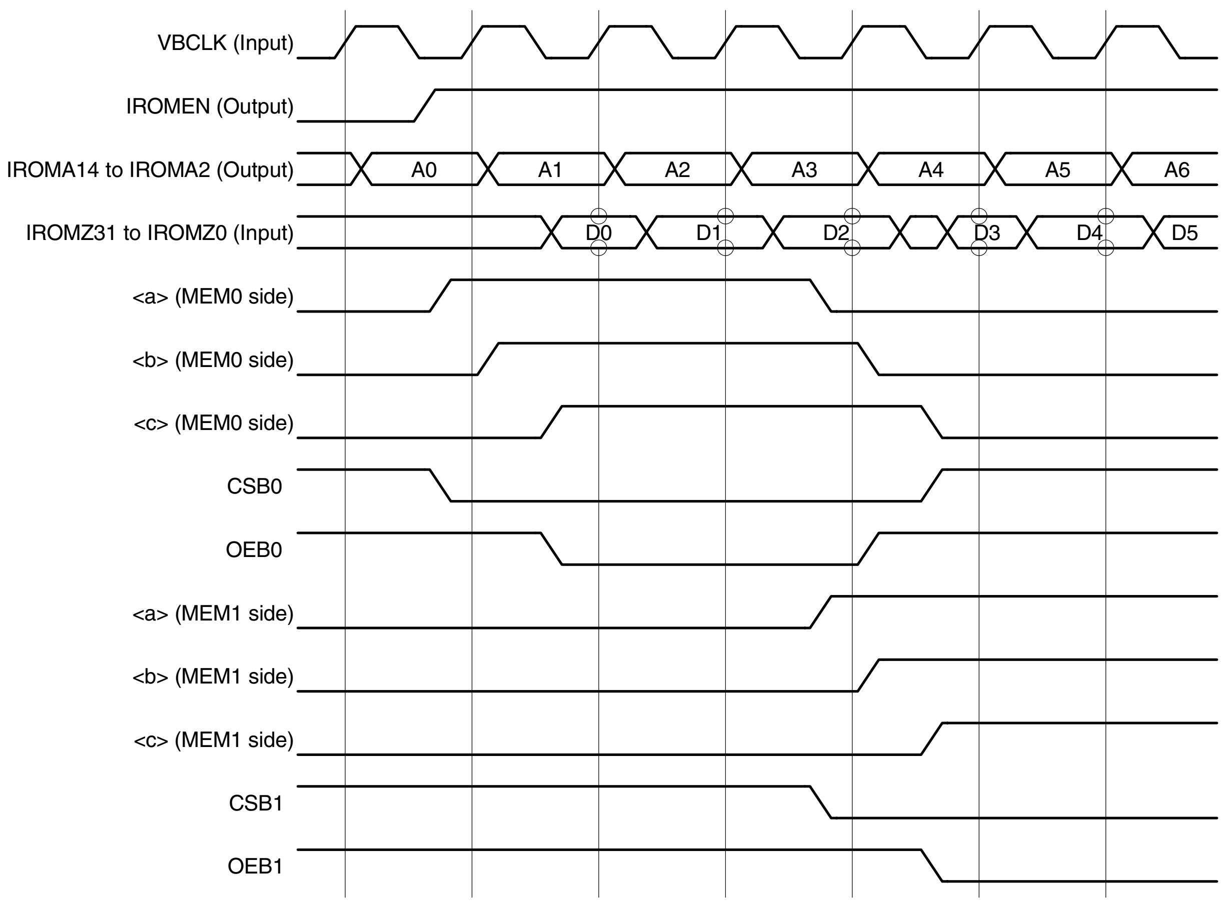


Figure 3-8. CSB and OEB Timings



Remarks 1. Ax: Arbitrary address

Dx: Data corresponding to address “Ax”

2. The circles indicate the ROM data sampling timing.

3.3 Connection of Compiled Memory to VDB

The VDB is a bus that is optimized for data access and can perform access in as fast as one clock.

Access is possible using 8-bit, 16-bit, or 32-bit units.

When connecting compiled RAM to the VDB, it is necessary to compute the relationship of the compiled RAM access time and cycle time to the CPU clock frequency and check that the timing fits adequately (see **9.4 Timing Adjustment**).

Figure 3-9 shows the RAM access timing, Figure 3-10 shows the NB85E RAM area setting, Figure 3-11 shows an example in which four compiled RAMs of 8 bits × 2 Kwords for the CB-9 Family VX Type are connected, and Figure 3-12 shows an example in which eight compiled RAMs of 8 bits × 2 Kwords are connected.

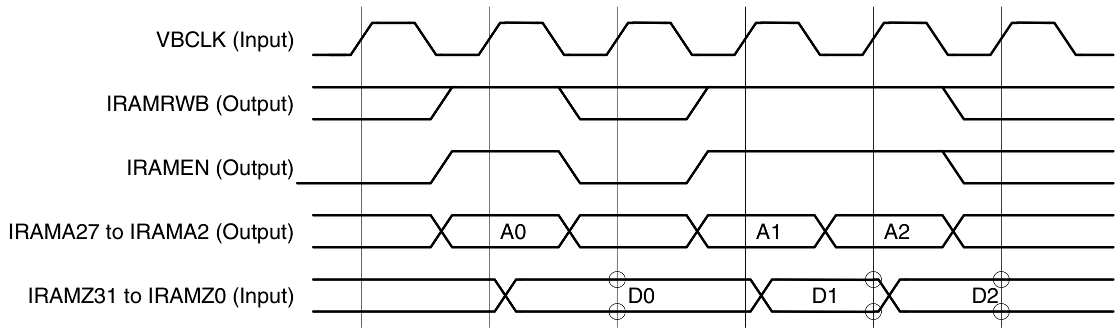
Remarks 1. For details about compiled memory, refer to the **CB-9 Family VX/VM Type Memory Macro (Compiled Type) Design Manual (A12982E)**.

2. For the creation method of compiled memory, refer to **NEC SYSTEM LSI DESIGN OPENCAD V5.4 OPC_VSHELL User's Manual (A15050E).**

3. When using compiled memory, test wiring is needed in order to use the ASIC standard test procedure. Refer to the **CB-9 Family VX/VM Type Design Manual (A12745E)** regarding the test procedure.

Figure 3-9. RAM Access Timing

(a) Read timing



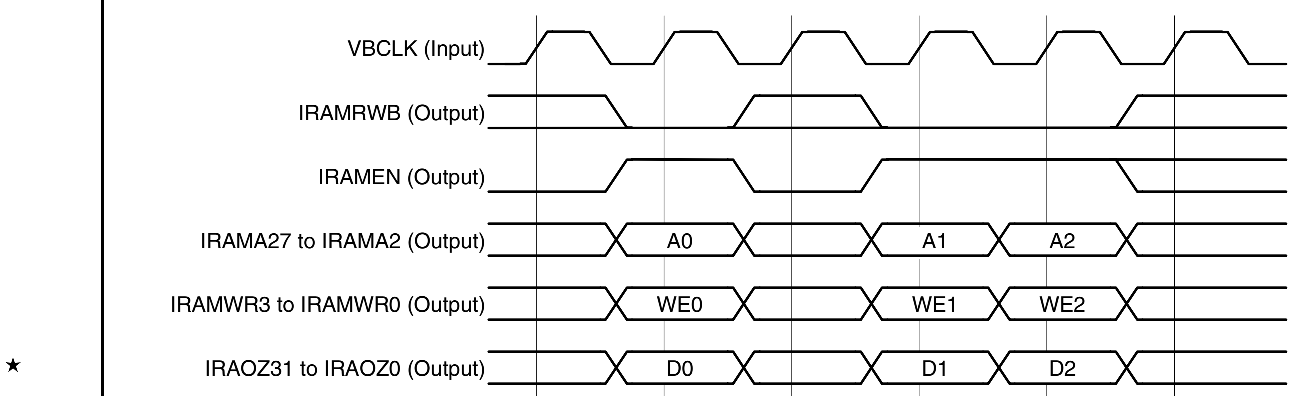
Remarks 1. Ax: Arbitrary address

Dx: Data corresponding to address “Ax”

2. The circles indicate the RAM data sampling timing.

3. Input and output are shown as seen from the NB85E.

(b) Write timing



Remarks 1. Ax: Arbitrary address

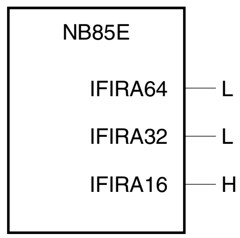
Dx: Data corresponding to address “Ax”

2. Input and output are shown as seen from the NB85E.

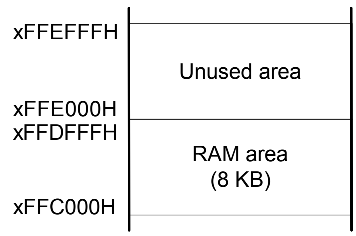
Figure 3-10. NB85E RAM Area Setting

(1) When connecting four compiled RAMs of 8 bits × 2 Kwords

(a) Pin settings



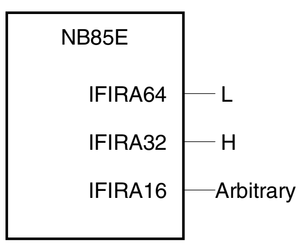
(b) Memory map



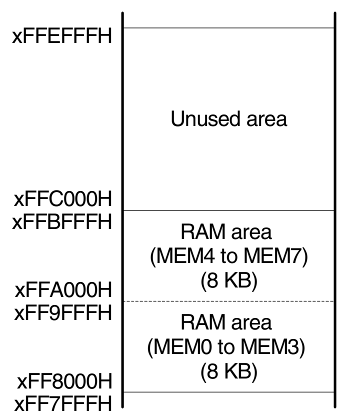
Remark L: Low level input
H: High level input

★ **(2) When connecting eight compiled RAMs of 8 bits × 2 Kwords**

(a) Pin settings

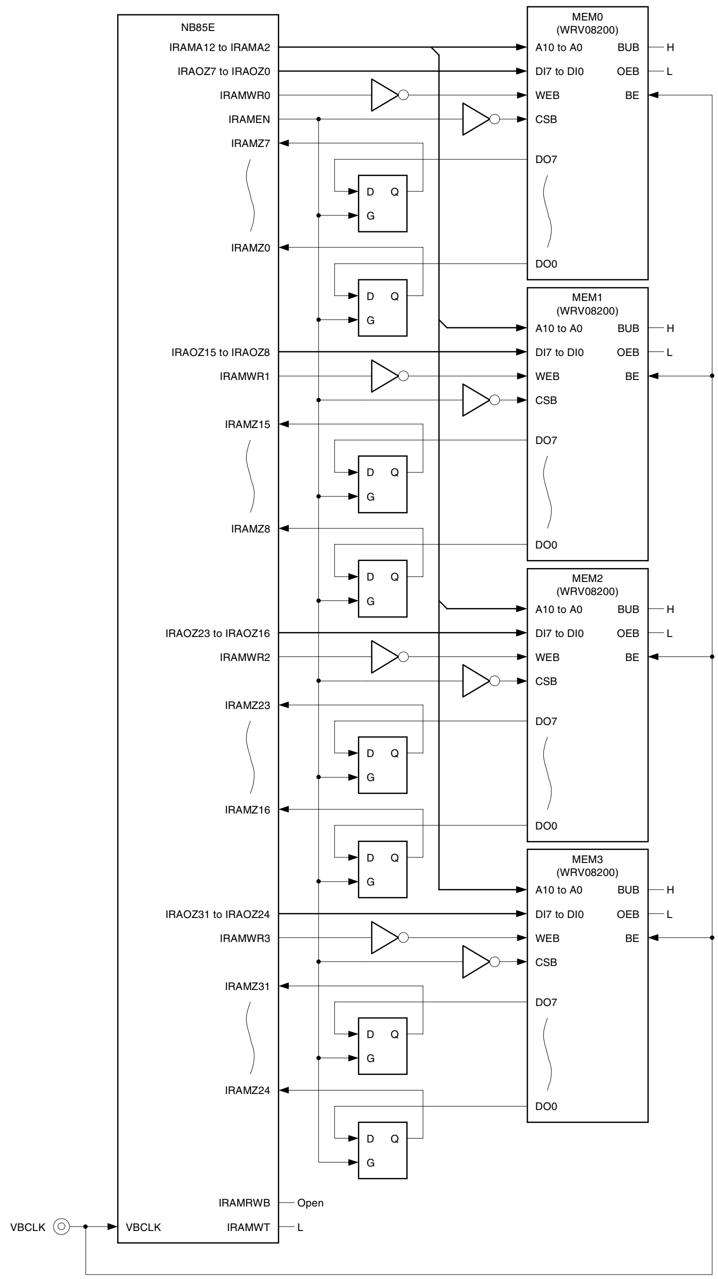


(b) Memory map



Remark L: Low level input
H: High level input

Figure 3-11. Example of Connecting Compiled Memories to VDB (Four Memories)



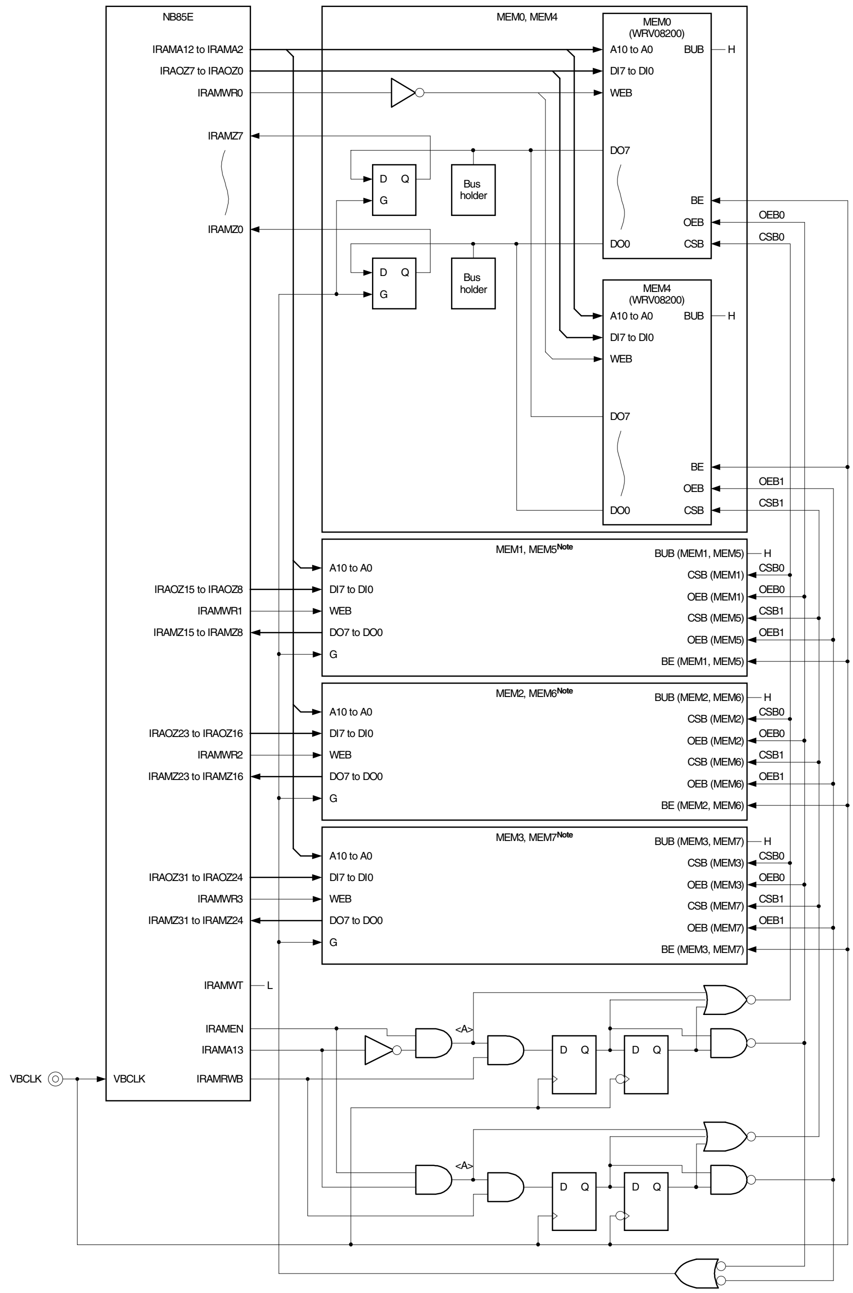
Remarks 1. In this connection example, backup mode is set not to be used. If backup mode is used, measures such as the separation of power supplies must be implemented. For details, refer to the **CB-9 Family VX/VM Type Memory Macro (Compiled Type) Design Manual (A12982E)**.

★ **2.** Since the data bus inverts the actual data logic (0 → 1 or 1 → 0), be careful when dumping data during simulation.

3. The figure shows the normal pin wiring method. Wire test pins in accordance with the ASIC test procedure (for details, refer to the **CB-9 Family VX/VM Type Design Manual (A12745E)**).

4. L: Low level input H: High level input Open: Open

★ **Figure 3-12. Example of Connecting Compiled Memories to VDB (Eight Memories)**



Note Same circuit configuration as MEM0 and MEM4.

Remarks 1. In this connection example, backup mode is set not to be used. If the backup mode is used, measures such as separation of power supplies must be implemented. For details, refer to the **CB-9 Family VX/VM Type Memory Macro (Compiled Type) Design Manual (A12982E)**.

2. Since the data bus inverts the actual data logic (0 → 1 or 1 → 0), be careful when dumping data during simulation.

3. The figure shows the wiring method for normal pins. In the case of test pins, perform wiring in accordance with ASIC test methods (for details, refer to the **CB-9 Family VX/VM Type Design Manual (A12745E)**).

4. L: Low level input, H: High level input

★ ### 3.3.1 Operation when eight RAMs are connected

(1) Read from same RAM area

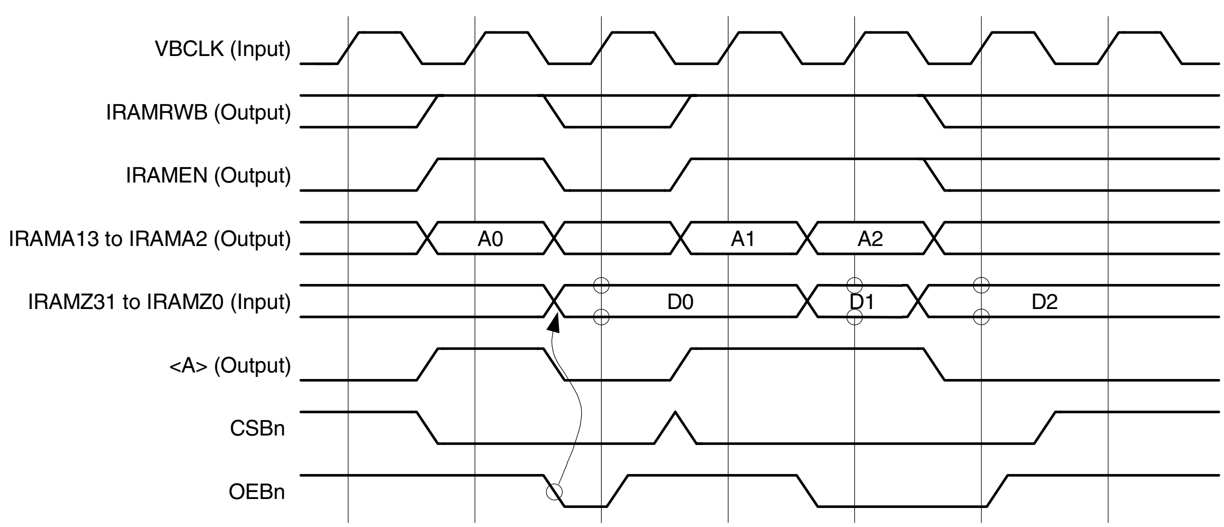
Figure 3-13 shows the timing when the RAM area of either MEM0 to MEM3 or MEM4 MEM7 is read. To support read access in the VDB pipeline method, OEB is generated with the result of decoding IRAMA13 delayed by 1 clock. To prevent data conflicts, CSB is generated by ORing the address decoding result and OEB.

As soon as the result of inputting and decoding IRAMEN and IRAMA13 (refer to **<A>** in **Figure 3-12**) becomes active, CSBn becomes active. RAM starts being read at the next rising edge of the clock, but output is not performed because OEBn is inactive at that time. OEBn becomes active at the next falling edge of the clock, and valid data is input to IRAMZ31 to IRAMZ0 (n = 1, 0).

At the next rising edge after the decoded result becomes inactive, at the same time that OEBn becomes inactive, IRAMZ31 to IRAMZ0 hold the value at that time. This is because they hold the value without reading RAM while OEBn is inactive. At the next falling edge of the clock after OEBn becomes inactive, CSBn becomes inactive.

Since the pipeline method is not used during write access, the result of decoding IRAMA13 is transferred as is to CSB.

Figure 3-13. RAM Read Timing (Read from Same RAM Area)



Remarks 1. Ax: Arbitrary address
Dx: Data corresponding to address "Ax"

2. The circles indicate the RAM data sampling timing

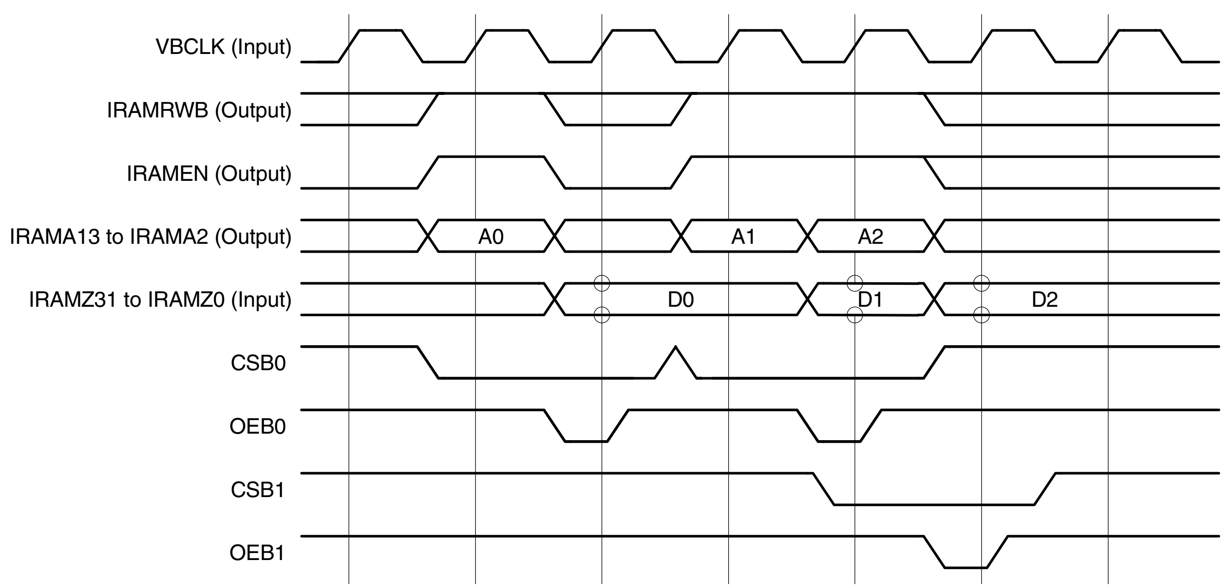
3. n = 1, 0

(2) Read when RAM area has changed

Figure 3-14 shows the CSB0, OEB0, CSB1, and OEB1 timings during continuous read access to A0 and A1 (MEM0 to MEM3 area) and A2 (MEM4 to MEM7 area).

Even if IRAMA13 to IRAMA2 change from A1 to A2, and the RAM address changes from the MEM0 to MEM3 area to the MEM4 to MEM7 area, the timing for reading the data (D1) corresponding to A1 is the rising edge of the clock 0.5 clocks later, so OEB0 and CSB0 for the MEM0 to MME3 area are extended 0.5 clocks and 1 clock, respectively. Since CSB1 reads D2, it becomes active when IRAMA13 to IRAMA2 become A2. In order to prevent data conflict, OEB1 becomes active 0.5 clocks after OEB0 becomes inactive.

Figure 3-14. RAM Read Timing (When RAM Area Has Changed)



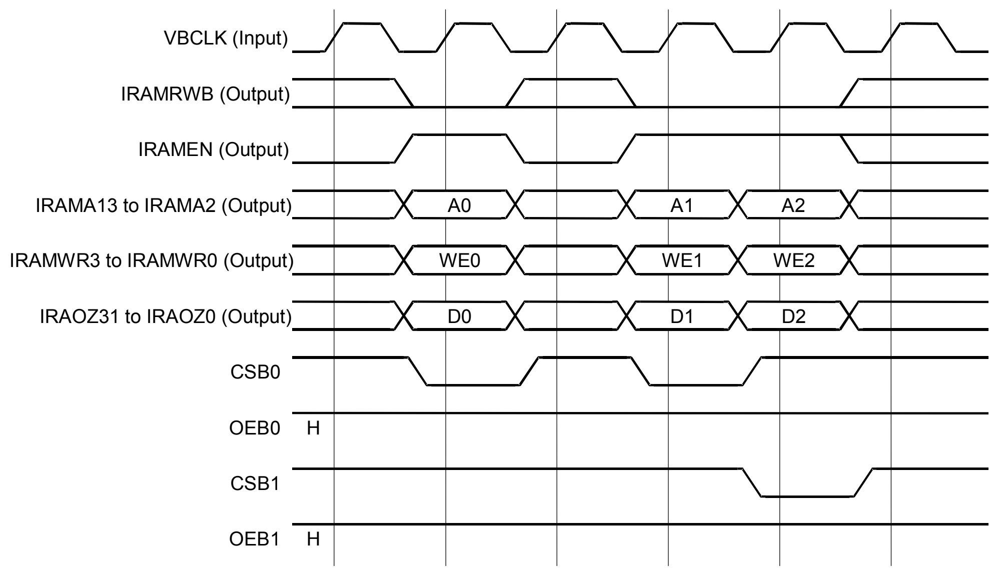
Remarks 1. Ax: Arbitrary address
Dx: Data corresponding to address "Ax"

2. The circles indicate the ROM data sampling timing.

Figure 3-15 shows the CSB0 and CSB1 timings during continuous write access to A0 and A1 (MEM0 to MEM3 area) and A2 (MEM4 to MEM7 area).

In the case of write access, since pipeline processing is not required, CSB is generated based on IRAMEN and the address decoding result.

Figure 3-15. RAM Write Timing



Remark Ax: Arbitrary address

Dx: Data corresponding to address “Ax”

CHAPTER 4 CONNECTION TO VSB

4.1 Connection of User Logic

This section describes how to connect user logic to the VSB.

Three kinds of read/write 32-bit registers are taken as examples of user logic. These registers operate using no wait, one wait, and address hold.

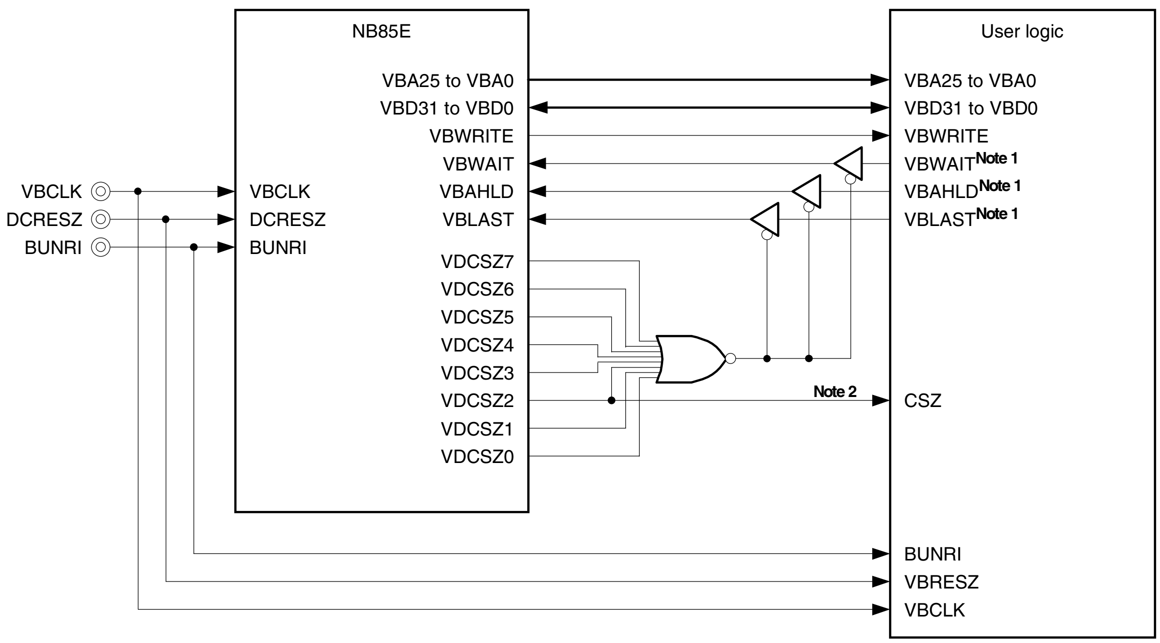
Each register can be written in word units only and undefined values are written in bits that are written by byte or halfword instructions. Moreover, all bits are cleared (0) on reset input.

Signals that user logic uses in the interface to the outside are as follows.

Table 4-1. User Logic Interface Signals

Signal Name	I/O	Function
VBCLK	Input	System clock
VBRESZ	Input	System reset
CSZ	Input	Select signal (one of VDCSZ7 to VDCSZ0)
VBA25 to VBA0	Input	Address input
VBD31 to VBD0	I/O	Data input/output
VBWRITE	Input	Write status
VBWAIT	Output	Wait response output
VBAHLD	Output	Address hold response output
VBLAST	Output	Last response output
BUNRI	Input	Normal/test mode selection input (used to make output high impedance status when testing)

★ **Figure 4-1. Example of Connecting User Logic to VSB**



Notes 1. Drive the user logic response signal only when the VSB slave is selected (when any of the VDCSZn signals is active) (n = 7 to 0).

2. The user logic CSZ is an example when the CS2 area is used.

4.1.1 Overview of VSB operation

★ The basic VSB timing is 1-clock access, as shown in Figure 4-2. Also, the USB is a pipeline type bus that shifts data corresponding to addresses by one clock.

Timing adjustment is performed by inserting a wait cycle or address hold cycle as needed by manipulating the VBWAIT signal or VBAHLD signal.

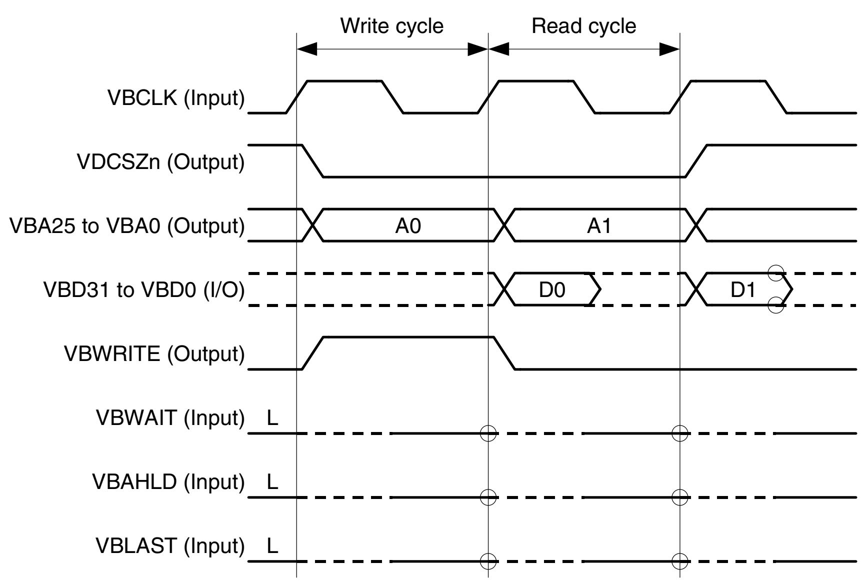
(1) For no wait (basic timing)

Output an address during one clock to perform a read or write during the next half clock.

The data bus (VBD31 to VBD0) drives only while the clock (VBCLK) is high level. While the clock is low level, it has undefined status in which the NB85E internal bus holder is driving (Weak unknown).

The VBWAIT, VBAHLD, and VBLAST signals drive only while the clock is low level. While the clock is high level, it has undefined status in which the NB85E internal bus holder is driving (Weak unknown).

Figure 4-2. VSB Timing (No Wait)



Remarks 1. Ax: Arbitrary address

Dx: Data corresponding to address "Ax"

2. Levels of broken-line portions indicate undefined status in which the (NB85E internal) bus holder is driving (Weak unknown).

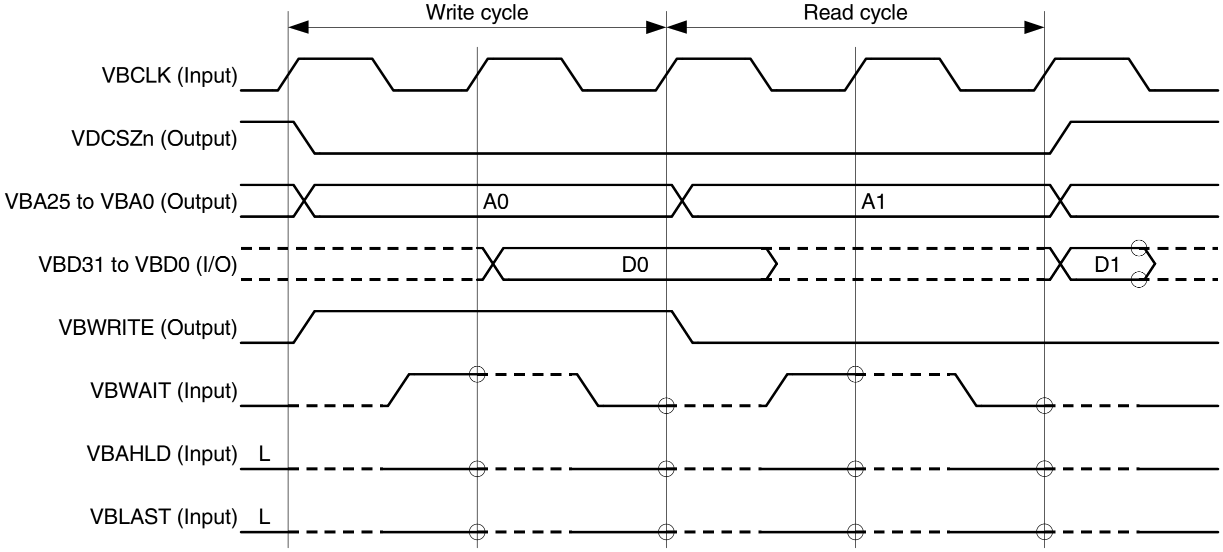
3. Input and output are shown as seen from the NB85E.

4. The circles indicate the sampling timing.

(2) For one wait

Output an address in two clock periods to perform a read or write in the next half clock. By inserting a wait, a low-speed circuit can be connected.

Figure 4-3. VSB Timing (One Wait)



Remarks 1. Ax: Arbitrary address
Dx: Data corresponding to address “Ax”

2. Levels of broken-line portions indicate undefined status in which the (NB85E internal) bus holder is driving (Weak unknown).

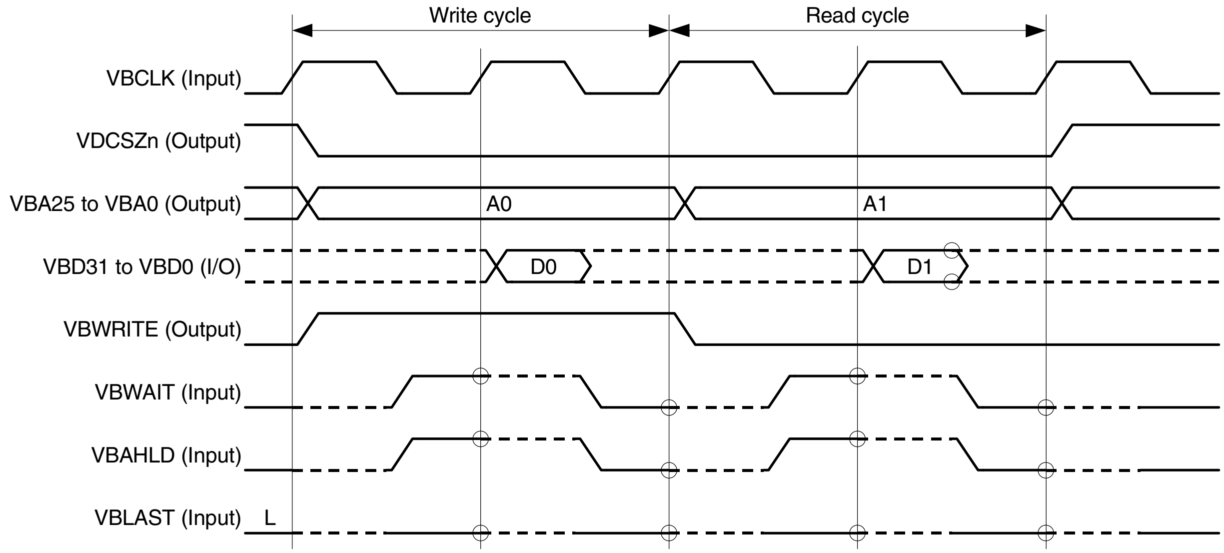
3. Input and output are shown as seen from the NB85E.

4. The circles indicate the sampling timing.

(3) For address hold

Similar to the case for one wait, two clock periods are used for one data transfer. The time interval from address output until data input or output is the same as for basic timing, but since the address is held until the end of the read cycle or write cycle, designing the interface circuit is easy.

★ **Figure 4-4. VSB Timing (Address Hold)**



Remarks 1. Ax: Arbitrary address

Dx: Data corresponding to address “Ax”

2. Levels of broken-line portions indicate undefined status in which the (NB85E internal) bus holder is driving (Weak unknown).

3. Input and output are shown as seen from the NB85E.

4. The circles indicate the sampling timing.

4.1.2 Circuit example

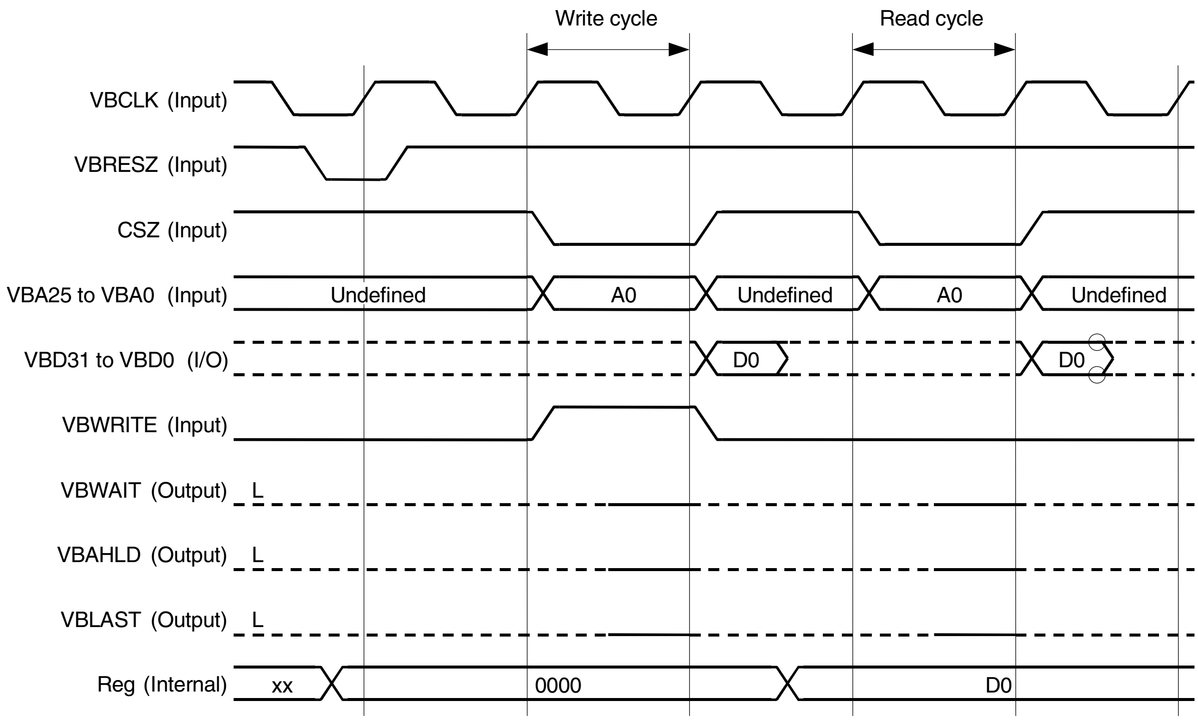
(1) For no wait

An example of connecting user logic that operates using no wait is shown below. In this example, reading or writing is performed at high speed, but when the VBCLK frequency is high, guaranteeing the data setup time or hold time is difficult.

Figure 4-5. HDL Creation Example for User Logic Operating with No Wait

```
module  UDL_NOWAIT( VBCLK, VBRESZ, CSZ, VBA, VBD, VBWRITE, VBWAIT, VBAHLD, VBLAST, BUNRI ) ;
        input           VBCLK ;
        input           VBRESZ ;
        input           CSZ ;
        input [25:0]    VBA ;
        input [31:0]    VBD ;
        input           VBWRITE ;
        output          VBWAIT ;
        output          VBAHLD ;
        output          VBLAST ;
        input           BUNRI ;

// nowait
        reg [31:0]      reg1 ;
        reg             sel_reg_rd ;
        reg             sel_reg_wr ;

        wire            sel = (CSZ==1'b0 && VBA[7:2]==6'b0000_00) ? 1'b1 : 1'b0 ;

// wait control
        assign          VBWAIT = (~VBCLK & sel) ? 1'b0 : 1'bZ ;
        assign          VBAHLD = (~VBCLK & sel) ? 1'b0 : 1'bZ ;
        assign          VBLAST = (~VBCLK & sel) ? 1'b0 : 1'bZ ;

// read
        assign          VBD = ( sel_reg_rd & VBCLK & ~BUNRI ) ? reg1 : 32'hZZZZZZZZ ;

        always @( VBCLK )
        begin
                if( VBCLK == 1'b0 )
                begin
                        sel_reg_rd <= sel & ~VBWRITE ;
                end
        end

// write
        always @( negedge VBCLK or negedge VBRESZ )
        begin
                if( ~VBRESZ )
                        reg1 <= 32'h00000000 ;
                else
                begin
                        if( sel_reg_wr )
                                reg1 <= VBD ;
                        sel_reg_wr <= sel & VBWRITE ;
                end
        end
endmodule
```

Figure 4-6. Timing Chart for No Wait



Remarks 1. A0: Arbitrary address

D0: Data corresponding to address “A0”

2. Levels of broken-line portions indicate undefined status in which the (NB85E internal) bus holder is driving (Weak unknown).

3. Input and output are shown as seen from the user logic.

(2) For insertion of one wait

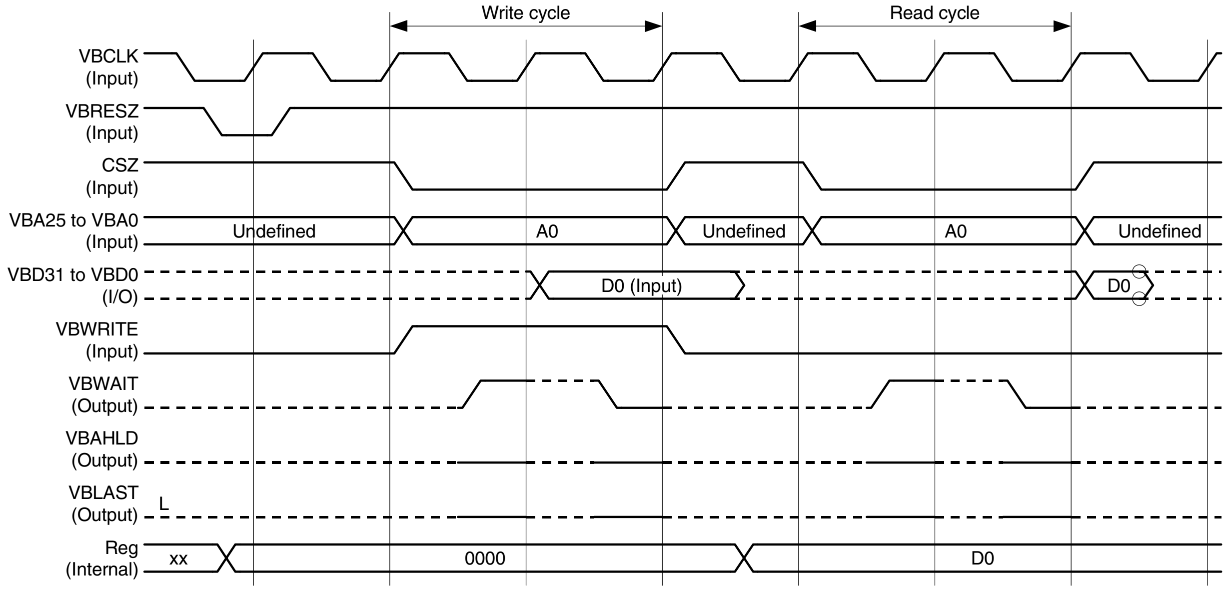
An example of connecting user logic that operates using one wait is shown below. In this example, delaying the data read or write timing 1 clock makes it possible to have extra access time.

To increase the number of wait insertions, use a VBSTZ signal after extending it using a configuration like that of a shift register.

Figure 4-7. HDL Creation Example for User Logic Operating with One Wait

```
module  UDL_1WAIT( VBCLK, VBRESZ, CSZ, VBA, VBD, VBSTZ, VBWRITE, VBWAIT, VBAHLD, VBLAST, BUNRI ) ;
        input           VBCLK ;
        input           VBRESZ ;
        input           CSZ ;
        input [25:0]    VBA ;
        input [31:0]    VBD ;
        input           VBSTZ ;
        input           VBWRITE ;
        output          VBWAIT ;
        output          VBAHLD ;
        output          VBLAST ;
        input           BUNRI ;

        reg [31:0]      reg2 ;
        reg             sel_reg_rd ;
        reg             sel_reg_wr ;

        wire            sel = (CSZ==1'b0 && VBA[7:2]==6'b0000_01) ? 1'b1 : 1'b0 ;

// wait control
        wire            wa = ~VBSTZ ;   // 1wait
        assign          VBWAIT = (~VBCLK & sel) ? wa : 1'bZ ;
        assign          VBAHLD = (~VBCLK & sel) ? 1'b0 : 1'bZ ;
        assign          VBLAST = (~VBCLK & sel) ? 1'b0 : 1'bZ ;

// read
        assign          VBD = ( sel_reg_rd & VBCLK & ~BUNRI) ? reg2 : 32'hZZZZZZZZ ;
        always @( VBCLK )
        begin
                if( VBCLK == 1'b0 )
                begin
                        sel_reg_rd <= sel & ~VBWRITE ;
                end
        end

// write
        always @( negedge VBCLK or negedge VBRESZ )
        begin
                if( ~VBRESZ )
                        reg2 <= 32'h00000000 ;
                else
                begin
                        if( sel_reg_wr )
                                reg2 <= VBD ;
                        sel_reg_wr <= sel & VBWRITE & ~wa ;
                end
        end
endmodule
```

Figure 4-8. Timing Chart for One Wait Insertion



Remarks 1. A0: Arbitrary address
D0: Data corresponding to address "A0"

2. Levels of broken-line portions indicate undefined status in which the (NB85E internal) bus holder is driving (Weak unknown).

3. Input and output are shown as seen from the user logic.

(3) For address hold

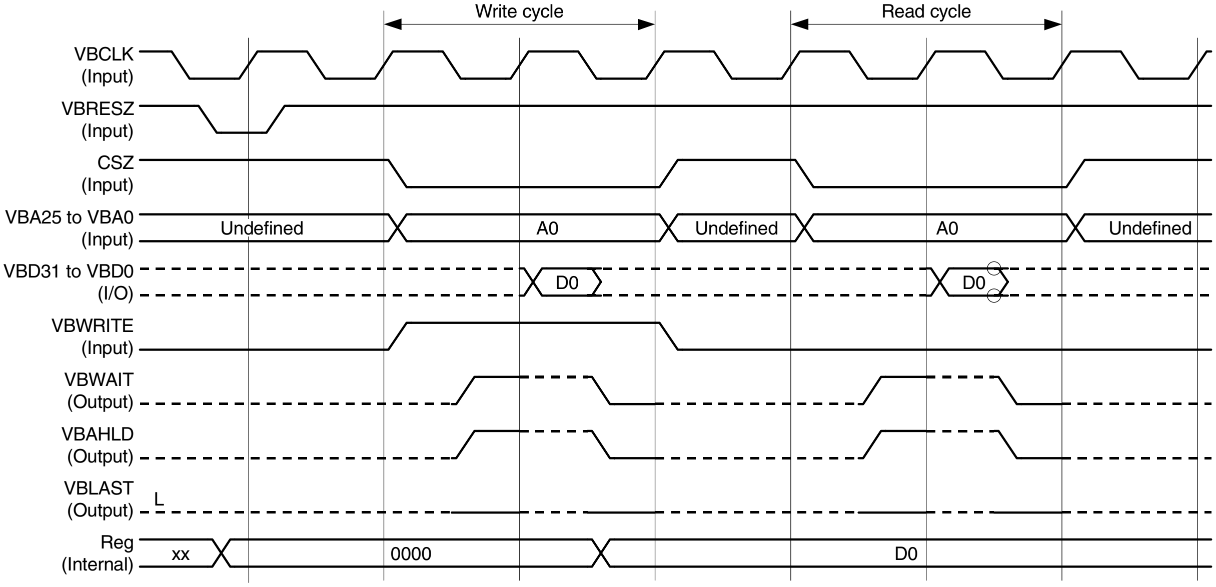
An example of connecting user logic in which the address hold function is used is shown below. In this example, the address does not change in a data read or write cycle. Therefore, address latch is not necessary and the circuit can be simplified.

This also can be combined with wait insertion to connect long-access-time I/O.

Figure 4-9. Example of HDL Creation for User Logic in Which Address Hold Function Is Used

```
module  UDL_AHLD( VBCLK, VBRESZ, CSZ, VBA, VBD, VBSTZ, VBWRITE, VBWAIT, VBAHLD, VBLAST, BUNRI ) ;
        input           VBCLK ;
        input           VBRESZ ;
        input           CSZ ;
        input [25:0]    VBA ;
        input [31:0]    VBD ;
        input           VBSTZ ;
        input           VBWRITE ;
        output          VBWAIT ;
        output          VBAHLD ;
        output          VBLAST ;
        input           BUNRI ;

        reg [31:0]      reg3 ;

        wire            sel = (CSZ==1'b0 && VBA[7:2]==6'b0000_10) ? 1'b1 : 1'b0 ;

// wait control
        wire            wa = ~VBSTZ ;   // 1wait
        assign          VBWAIT = (~VBCLK & sel) ? wa : 1'bZ ;
        assign          VBAHLD = (~VBCLK & sel) ? wa : 1'bZ ;
        assign          VBLAST = (~VBCLK & sel) ? 1'b0 : 1'bZ ;

// read
        reg             VBSTZ_1E
        always @( posedge VBCLK)
                        VBSTZ_1E = VBSTZ;
        wire            sel_reg_rd = (sel & ~VBWRITE) ;
        assign          VBD = ( sel_reg_rd & VBCLK & ~VBSTZ_1E & BUNRI ) ? reg3 : 32'hZZZZZZZZ ;

// write
        wire            sel_reg_wr = (sel & VBWRITE & ~wa) ;
        always @( negedge VBCLK or negedge VBRESZ )
        begin
                if( ~VBRESZ )
                        reg3 <= 32'h00000000 ;
                else
                begin
                        if( sel_reg_wr )
                                reg3 <= VBD ;
                end
        end
endmodule
```

Figure 4-10. Timing Chart for Address Hold



Remarks 1. A0: Arbitrary address

D0: Data corresponding to address "A0"

2. Levels of broken-line portions indicate undefined status in which the (NB85E internal) bus holder is driving (Weak unknown).

3. Input and output are shown as seen from the user logic.

4.2 Connection of Compiled Memory

This section describes how to connect high-density synchronous 1-port RAM to the VSB without using a memory controller (MEMC).

Figure 4-11 shows a connection example, Figure 4-12 shows a timing chart, and Figure 4-13 shows an example of the creation of the HDL for the compiled memory access controller (VSB_VDL_MEMCBC9).

The following high-density synchronous 1-port RAM is used in this connection example.

- Macro block name: W8K08100
- Memory capacity (total): 4 KB (1024 words × 8 bits × 4[Note])
- Operating frequency: 66 MHz

Note The bit width of a high-density synchronous 1-port RAM can be selected from 1 to 32 bits in 1-bit units. The VSB data bus width is 32 bits, but it must be possible to access RAM in 8-bit units. Therefore, connect four RAMs with a bit width of 8 bits to configure a 32-bit width.

★ For a normal access, the VSB performs a “pipeline method” bus operation, which shifts the data to an arbitrary address by one cycle. In the connection example in Figure 4-11, compiled memory access control is simplified by controlling the VBWAIT, VBAHLD, and VBLAST signals and responding to the NB85E in address hold status when reading and in wait status when writing.

Moreover, CB-9 Family VX/VM Type synchronous RAM has the following timing restrictions.

- Address change that uses the same timing as a clock input signal (BE) rise is prohibited
- CSB signal change when the clock input signal (BE) is high level is prohibited

To observe these restrictions, measures like the following are taken in the connection example in Figure 4-11.

- The clock (VBCLK) is input to the NB85E at the clock input pin (BE)
- An address that changes at the rise of VBCLK and CSB (VDCSZ) are used by latching at the fall of VBCLK
- OEB is used by latching so that it has the same timing as CSB
- For WEB, VBBENZ signal, which is the byte enable signal from the NB85E, is used under the conditions CSB = 0 and OEB = 1

Figure 4-11. Example of Connecting Compiled Memory

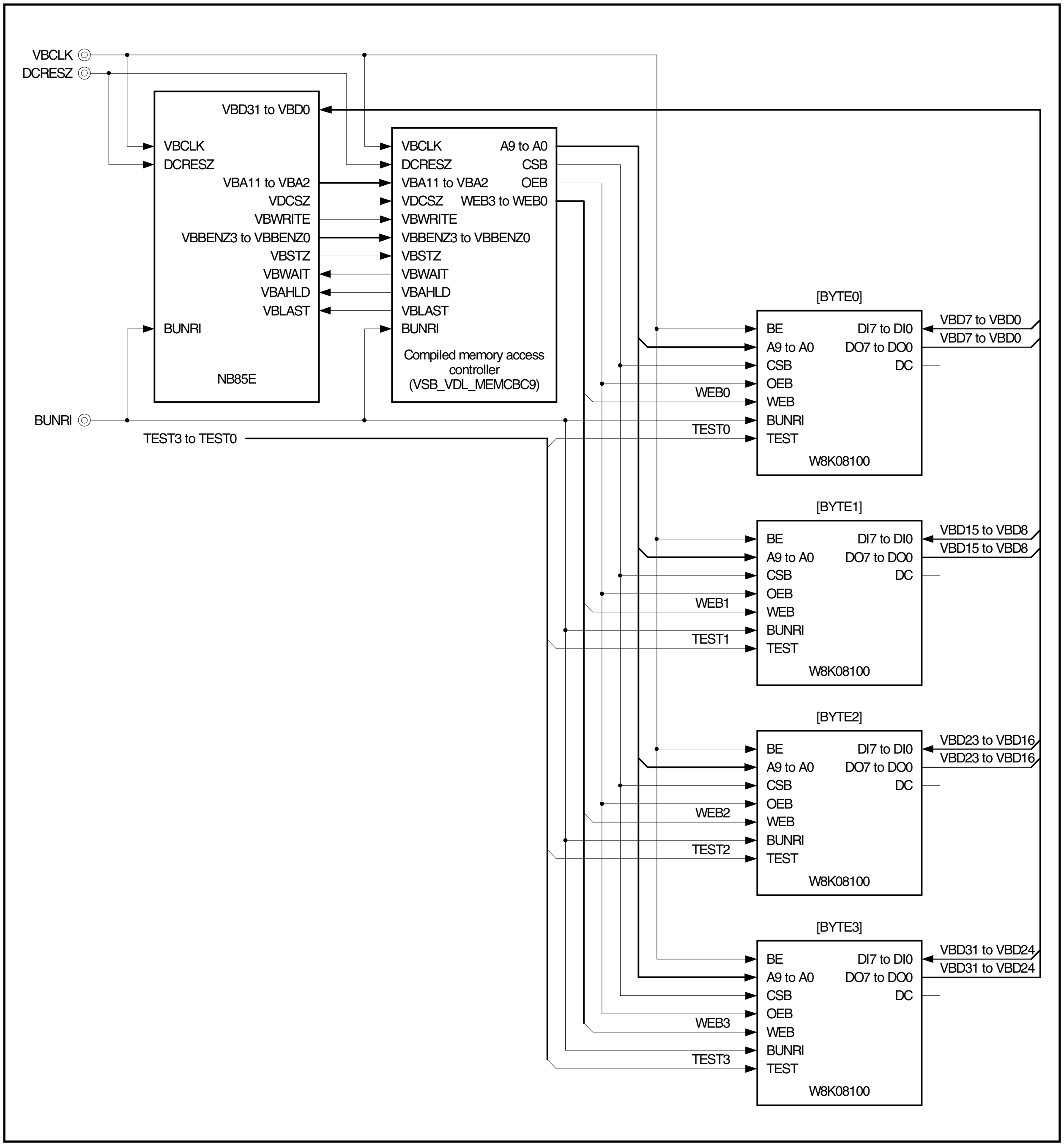
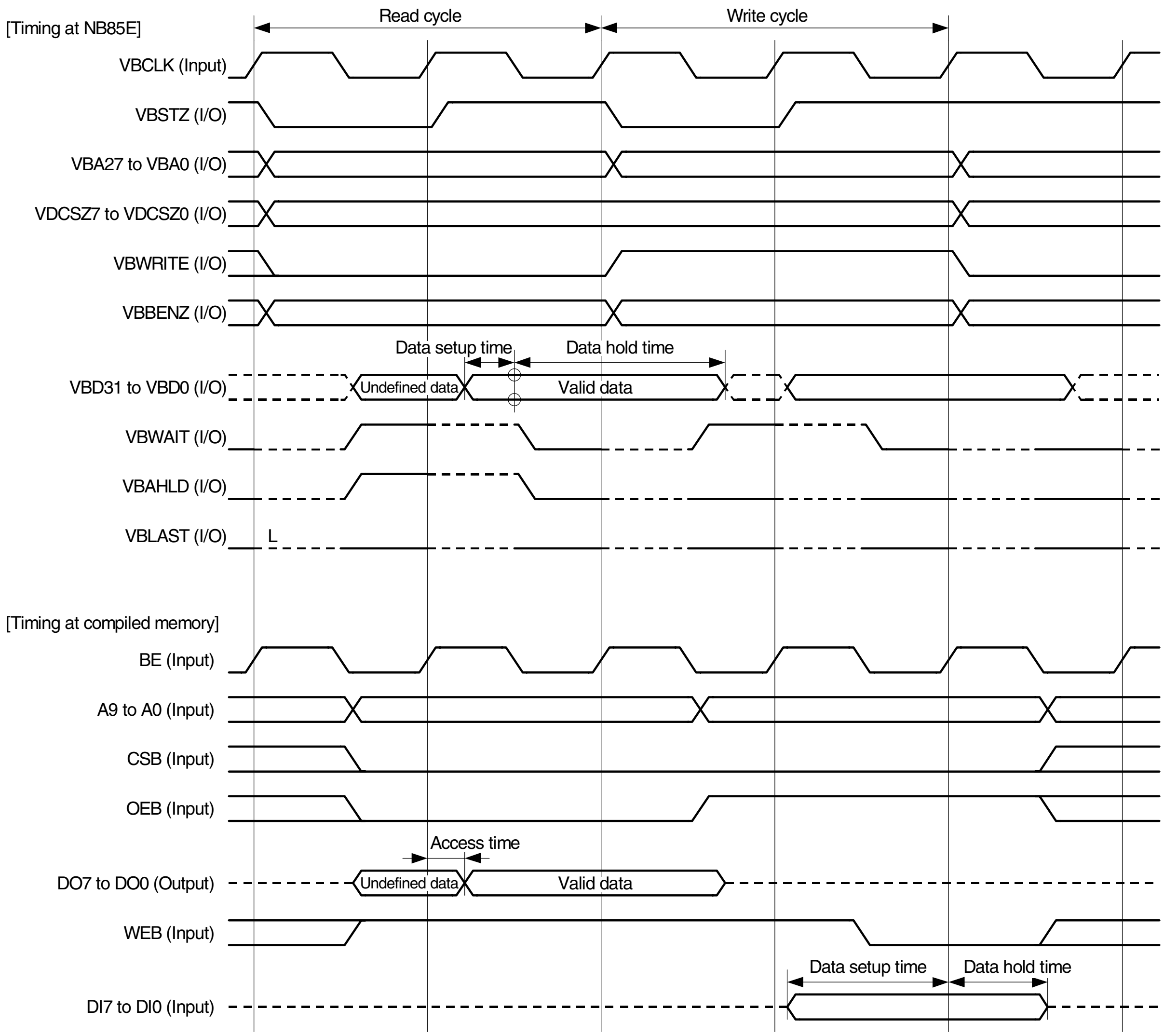


Figure 4-12. Timing Chart for Compiled Memory Access



Remarks 1. Levels of broken-line portions of the VBWAIT, VBAHLD, VBLAST, and VBD31 to VBD0 signals indicate an undefined status in which the (NB85E internal) bus holder is driving (Weak unknown). Broken-line portions of the DO7 to DO0 and DI7 to DI0 signals indicate high impedance.

2. The circles indicate the sampling timing.

Figure 4-13. Example of HDL Creation for Compiled Memory Access Controller (1/2)

```
module VSB_UDL_MEMCBC9 (
        VPRESZ,VBCLK,VBA,A,VDCSZ,VBWRITE,CSB,OEB,VBBENZ,WEB,VBSTZ,
        VBWAIT,VBAHLD,VBLAST,BUNRI );

        input           VPRESZ ;        // Reset
        input           VBCLK ;         // System clock
        input   [11:2]  VBA ;           // Address bus from VSB
        output  [9:0]   A ;             // Address bus to Memory macro
        input           VDCSZ ;         // Chip area select
        input           VBWRITE ;       // Transfer direction
        input   [3:0]   VBBENZ ;        // Byte enable
        output          CSB ;           // Chip area select
        output          OEB ;           // RAM data output enable
        output  [3:0]   WEB ;           // Write enable each byte
        input           VBSTZ ;         // Transfer start
        output          VBWAIT ;        // Wait response
        output          VBAHLD ;        // Address hold response
        output          VBLAST ;        // Last data response
        input           BUNRI ;         // Test mode

        wire    [11:2]  VBA ;
        reg     [9:0]   A ;
        reg     [3:0]   WEB ;
        reg             CSB,OEB ;
        reg             ResOut ;
        wire    [3:0]   VBBENZ ;
        wire            VBWAIT,VBAHLD,VBLAST ;
        reg     [1:0]   State ;
        reg     [2:0]   Response ;
        reg             WRITE ;

// Define status & output for Response({VBWAIT,VBAHLD,VBLAST})
        parameter [1:0] S_READY = 2'b00, S_WAIT = 2'b10, S_AHOLD = 2'b11 ;
        parameter [2:0] O_READY = 3'b000,O_WAIT = 3'b100,O_AHOLD = 3'b110 ;

        // synopsys async_set_reset "VPRESZ"
        always @( negedge VBCLK or negedge VPRESZ ) begin
                if( ~VPRESZ ) begin
                        A       = 10'h0 ;
                        CSB     = 1'b1 ;
                        OEB     = 1'b1 ;
                end else begin
                        A       = VBA ;
                        CSB     = VDCSZ ;
                        OEB     = VBWRITE ;
                end
        end

        always @( negedge VBCLK or negedge VPRESZ ) begin
                if( ~VPRESZ ) begin
                        WEB     = 4'b1111 ;
                        WRITE   = 0 ;
                end else if( ~VBSTZ && VBWRITE ) begin          // write cycle 1
                        WEB     = 4'b1111 ;
                        WRITE   = 1 ;
                end else if( WRITE ) begin                      // write cycle 2
                        if( ~CSB  ) WEB = VBBENZ ;
                        else        WEB = 4'b1111 ;
                        WRITE = 0 ;
                end else begin
                        WEB    = 4'b1111 ;
                        WRITE = 0 ;
                end
        end
```

Figure 4-13. Example of HDL Creation for Compiled Memory Access Controller (2/2)

```
        always @( negedge VBCLK or negedge VPRESZ ) begin
                if( ~VPRESZ ) begin
                        State       = S_READY ;
                        Response = O_READY ;
                end else if( ~VBSTZ && ! VBWRITE ) begin            // read cycle
                        State       = S_AHOLD ;
                        Response = O_AHOLD ;
                end else if( ~VBSTZ && VBWRITE ) begin              // write cycle
                        State       = S_WAIT ;
                        Response = O_WAIT ;
                end else begin
                        State       = S_READY ;
                        Response = O_READY ;
                end
        end

        always @( VDCSZ or VBCLK or BUNRI) ResOut = ~VDCSZ & ~VBCLK & ~BUNRI ;
        assign  { VBWAIT,VBAHLD,VBLAST } = ResOut ? Response : 3'bzzz ;

endmodule
```

CHAPTER 5 CONNECTION OF MEMORY CONTROLLER (MEMC)

★ The NB85E500 is a memory controller for the NB85E and NB85ET, and the NU85E500 is a memory controller for the NB85E.

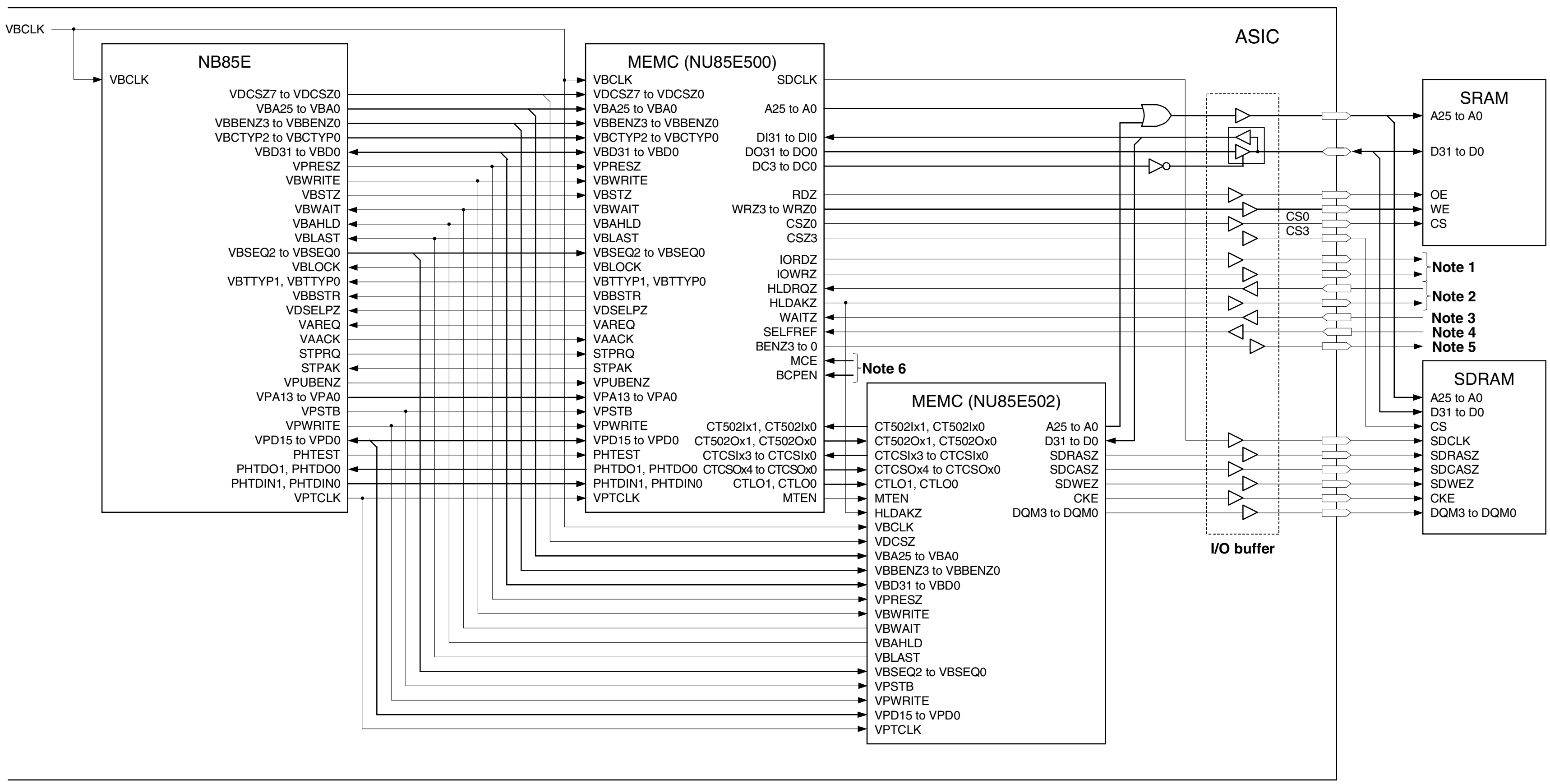
The Nx85E500 is used as follows according to the type of external memory to be connected.

Target CPU Core	Type of External Memory to Be Connected	Memory Controller (MEMC)
NB85E	SRAM, ROM, page ROM, flash memory	NB85E500/NU85E500
	SDRAM	NB85E500/NU85E500 + NU85E502
NB85ET	SRAM, ROM, page ROM, flash memory	NB85E500
	SDRAM	NB85E500 + NU85E502

Memory with a width of 8, 16, or 32 bits can be connected, but in the case of the latter two, writing must be performed in byte units. Write to SRAM and SDRAM in byte units using the WE and DQM signals, respectively.

For details, refer to the **Memory Controller NB85E, NB85ET User's Manual (A14206E)**.

★ **Figure 5-1. Example of Connecting NB85E, MEMC, and External Memory (SRAM, SDRAM)**



Notes 1. Used for DMA flyby transfer

2. Used for bus hold cycle

3. Used for external wait control

4. Used for self-refresh request

5. When byte enable is used

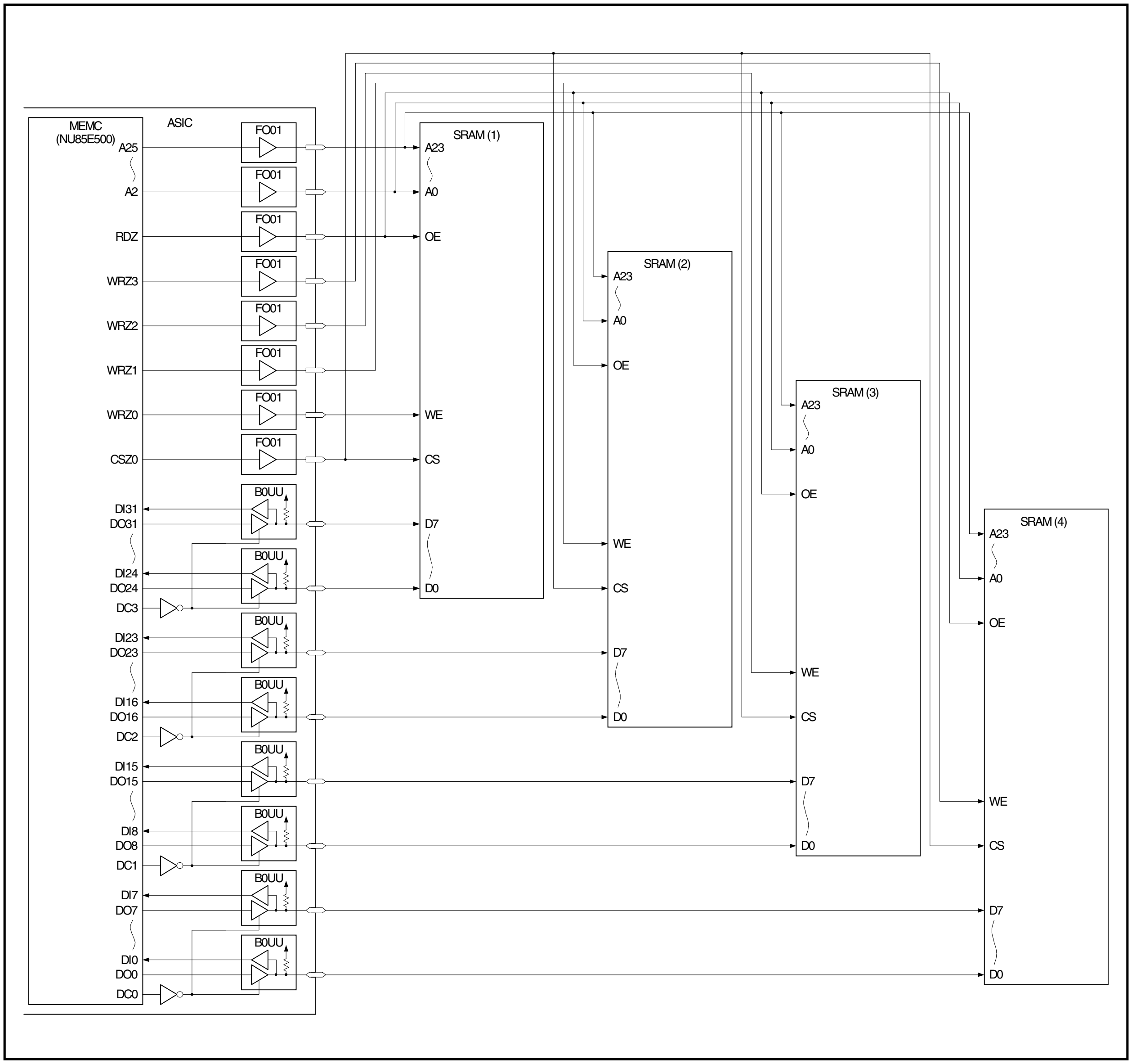
6. For initial settings

5.1 Connection to SRAM

Figure 5-2 shows an example of connection to SRAM.

In this connection example, four 8-bit width SRAMs are connected, making 32-bit width access possible.

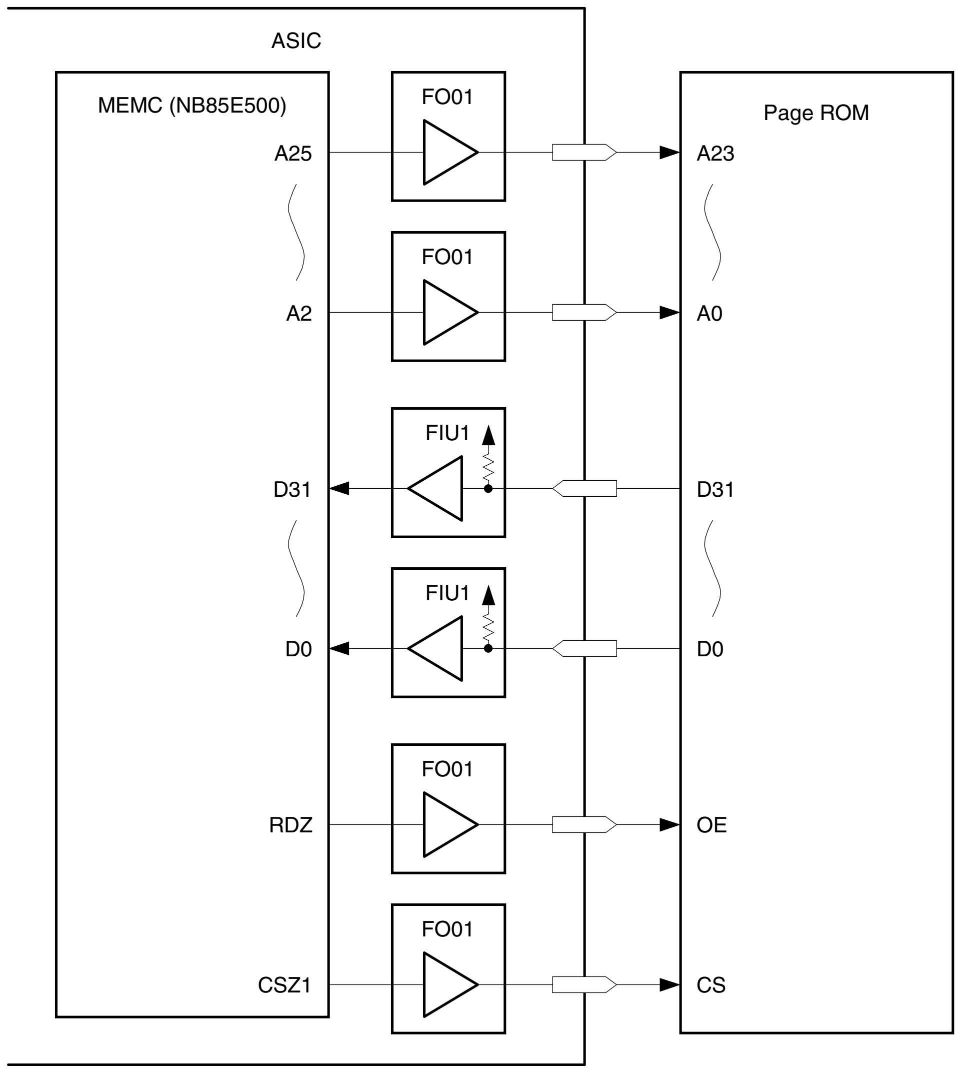
★ **Figure 5-2. Example of Connection to SRAM**



5.2 Connection to Page ROM

Figure 5-3 shows an example of connection to a 32-bit width page ROM.

Figure 5-3. Example of Connection to Page ROM



Remark If only ROM is externally connected to the ASIC, it is not necessary to make the data bus bi-directional.

5.3 Connection to SDRAM

When connecting an SDRAM to the MEMC (NU85E502), the SDRAM must satisfy the following conditions.

- Read latency: 2 or 3
- Data bus width: 8, 16, or 32 bits
- Row address width: 11 bits or 12 bits
- Address multiplex width (column address width): 8, 9, or 10 bits
- Refresh: CBR refresh and self-refresh

Figure 5-4 shows an example of connecting SDRAM and the MEMC. In this example, two 64 Mb (1 Mword × 16 bits × 4 banks) SDRAMs are connected, and access is in 32 bits.

Refresh is 4096 cycles/64 ms (corresponding to 64 Mb SDRAM).

★ The SDRAM in this connection example is 64 Mb (1 Mword × 16 bits × 4 banks), and the refresh cycle is 4096 cycles, resulting in the following address configuration.

- Row address input: A0 to A11 (12)
- Column address input: A0 to A7 (8)
- Bank select: A12, A13 (2)

Also, since the data bus width is 32 bits, addresses are connected to the NU85E502 with a 2-bit shift. Therefore, the NU85E502 and SDRAM addresses are connected as follows.

Row address and bank address during active command execution

A25	A24	A23	A22	A21	A20	A19	A18	A17	A16	A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
a25	a24	a23	a22	a21	a20	a19	a18	a25	a24	a23	a22	a21	a20	a19	a18	a17	a16	a15	a14	a13	a12	a11	a10	a9	a8

Bank addresses (A23 to A22) Row addresses (A13 to A2)

Column address output during read/write command execution

A25	A24	A23	A22	A21	A20	A19	A18	A17	A16	A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
a25	a24	a23	a22	a21	a20	a19	a18	a17	a16	a15	a14	a13	0	a11	a10	a9	a8	a7	a6	a5	a4	a3	a2	a1	a0

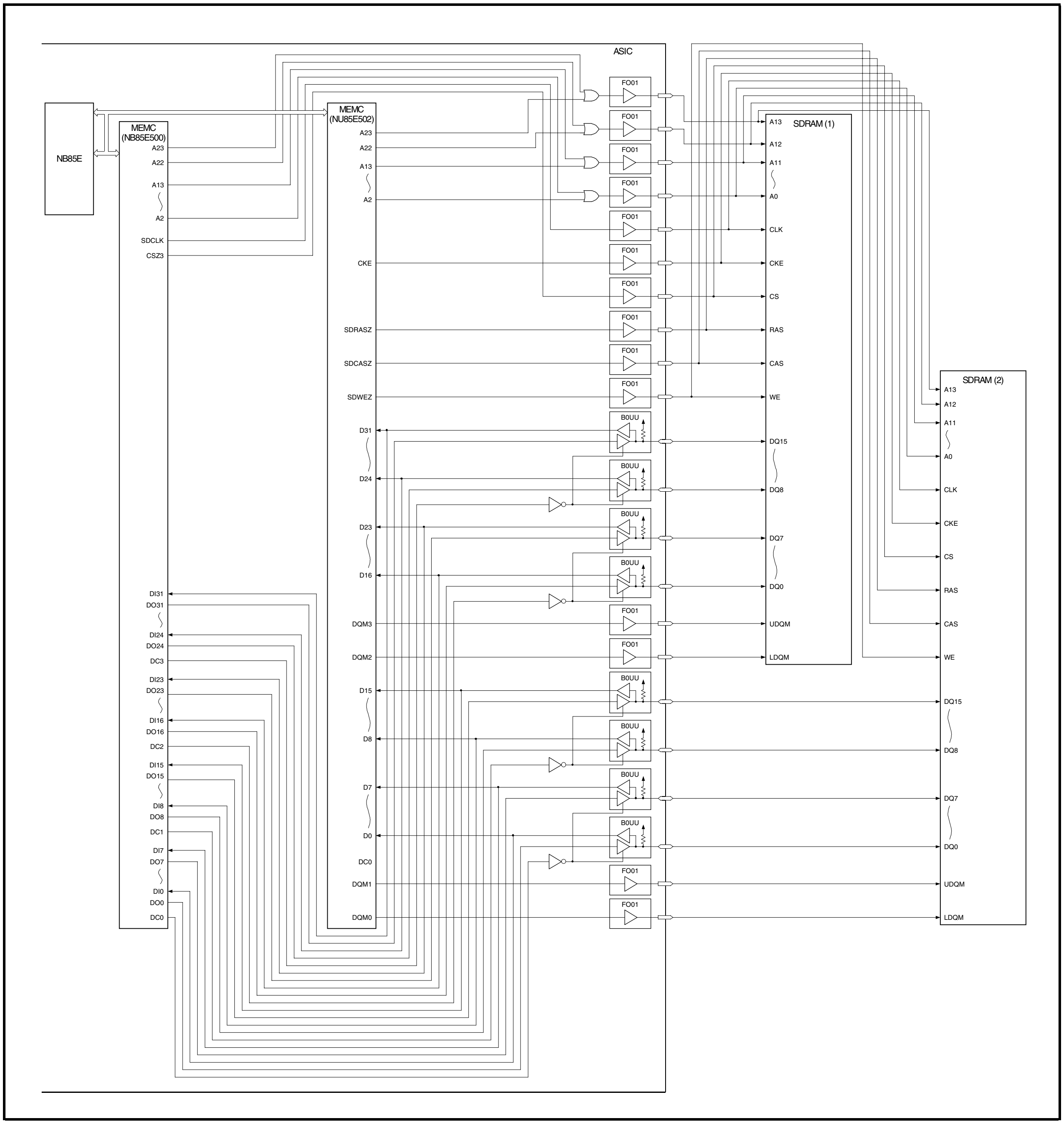
Column addresses (A9 to A2)

- Connection of NU85E502 and SDRAM

  A23, A22 (NU85E502) → BA0 (A13), BA1 (A12) (SDRAM)

  A13 to A2 (NU85E502) → A11 to A0 (SDRAM)

★ **Figure 5-4. Example of Connection to SDRAM**

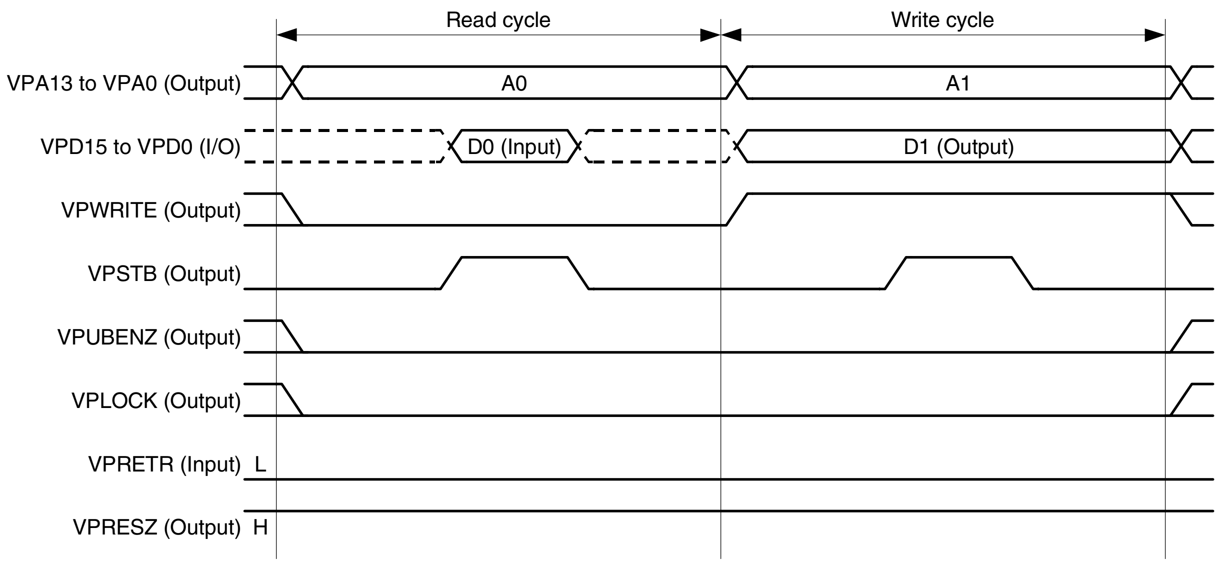


6.1 Overview of NPB

★ The NPB (NEC peripheral bus) of the NB85E is a 16-bit width asynchronous bus that operates at a bus speed of 10 MHz or less. User logic operating at a frequency of 10 MHz or less can be connected.

Remark A macro connected to the VSB such as the MEMC, instruction cache, or data cache must also be connected to the NPB.

Figure 6-1. NPB Operation Timing Chart



Remarks 1. Ax: Arbitrary address
Dx: Data corresponding to address "Ax"

2. Levels of broken-line portions indicate an undefined status in which the (NB85E internal) bus holder is driving (Weak unknown).

3. Input and output are shown as seen from the NB85E.

6.2 Connection of User Logic

This section shows an example of connecting user logic to the NPB.

An example of user logic in which two 8-bit registers and one 16-bit register are lined up has been created. The user logic macro name is UDL1 and the names of the registers are UDL11, UDL12, and UDL13.

(1) Register mapping

The memory address at which the user logic is placed is in a programmable peripheral I/O area or peripheral I/O area within the following ranges (for details, refer to the **NB85E Hardware User's Manual (A13971E)**).

- xFFF200H to xFFF47FH
- ★ xFFF520H to xFFF7BFH
- ★ xFFF800H to xFFFFFFH

An example in which the user logic registers are placed at addresses starting at address FFFFF880H is shown below.

Table 6-1. Example of User Logic Address Assignment

Address	Macro Name	Register Name	Bit Width
FFFFF880	UDL1	UDL11	8 bits
FFFFF881	UDL1	UDL12	8 bits
FFFFF882	UDL1	UDL13	16 bits

(2) Creation of address decoder

A connection example of user logic, an example of HDL creation for the address decoder and user logic, and a timing chart are shown below.

This example is designed so that UDL11, UDL12, and UDL13 are processed together in one module. Since four addresses are assigned, the address bus handles them using the two signals VPA0 and VPA1.

Caution If user logic is connected to the NPB, design it so that the data bus is high impedance when in test mode (active level input to BUNRI pin) (see 10.2.1 Processing NB85E and NB85ET pins and the NB85E Hardware User's Manual (A13971E)).

Figure 6-2. Example of Connecting User Logic

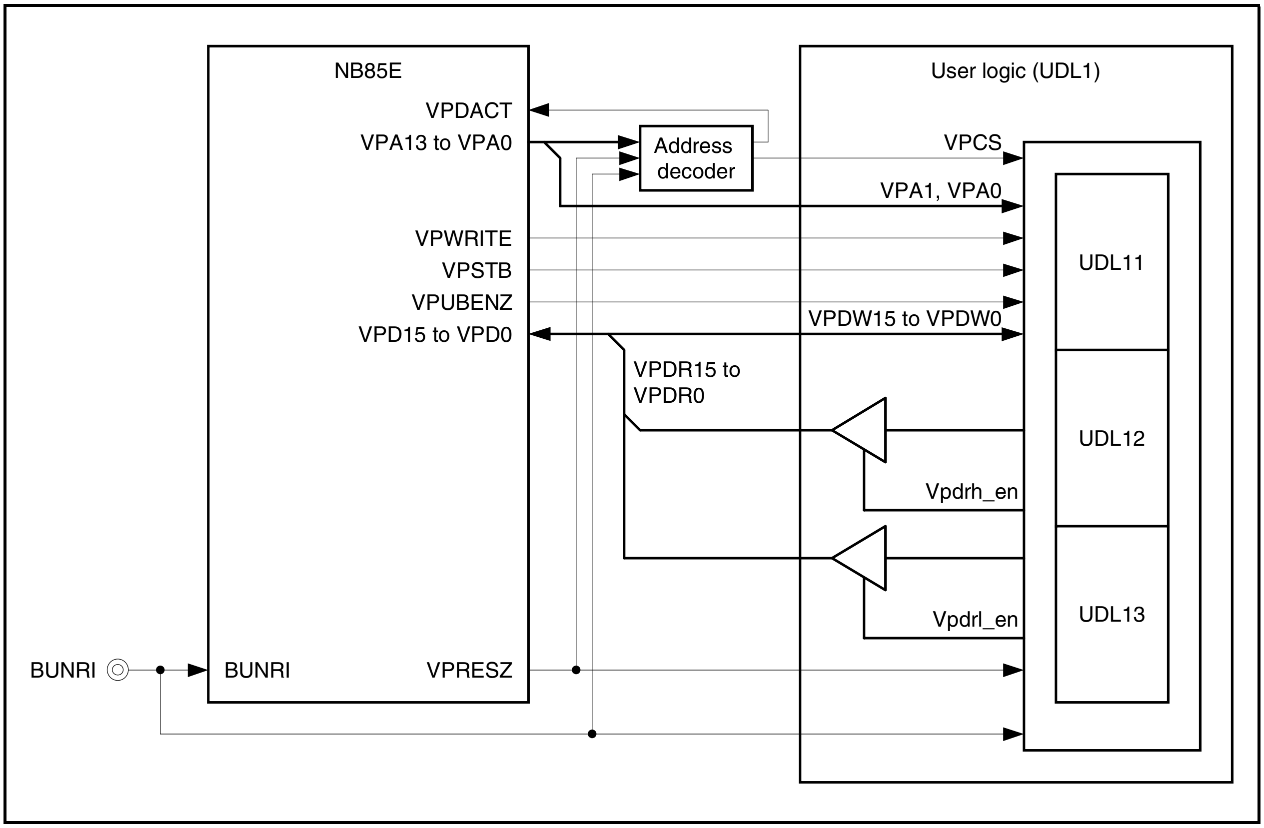


Figure 6-3. Example of Address Decoder HDL Creation

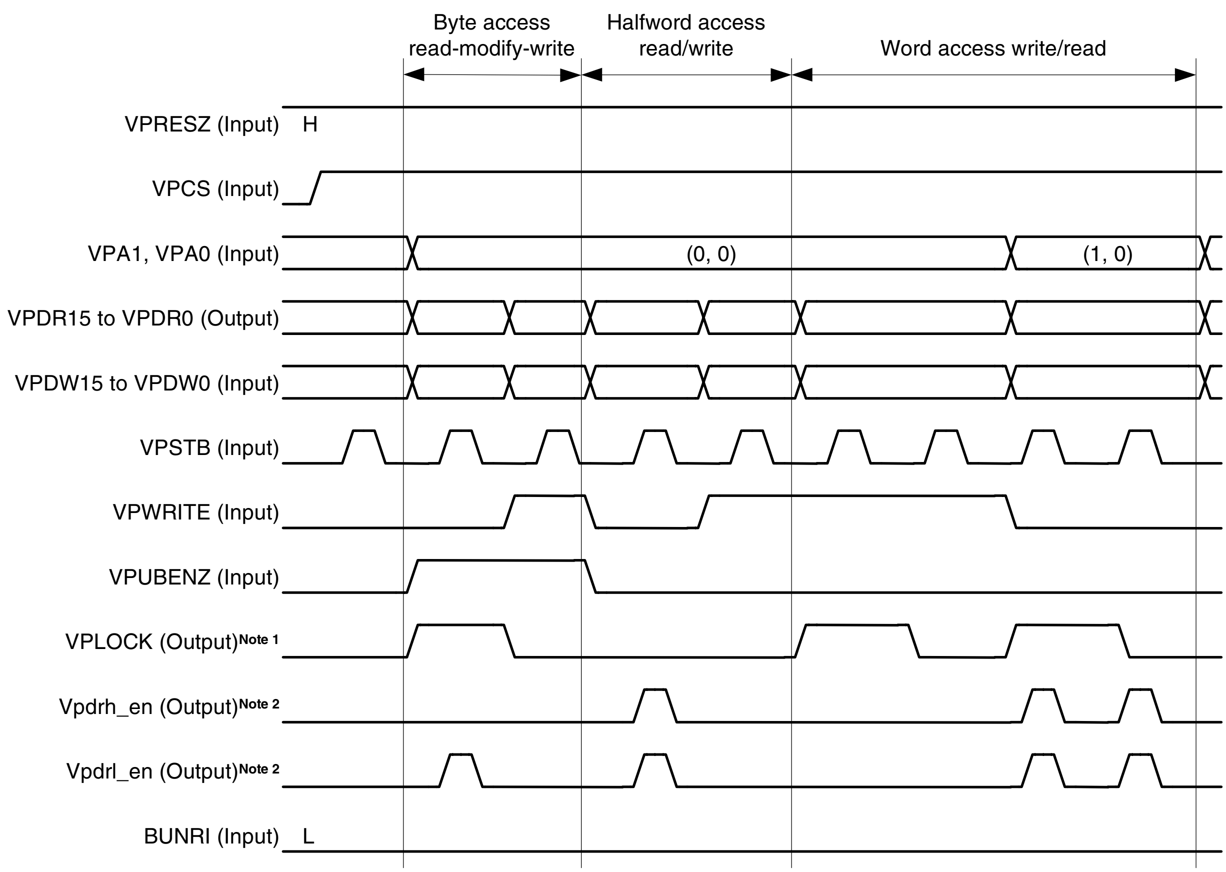
```
module ADRSDEC_UDL ( VPA, VPDACT, VPRESZ, UDL1_VPCS, BUNRI
) ;

parameter     UDL1_ADR        = 12'b11_1000_1000_00 ;   // 3880H

input  [13:0]  VPA ;
input          VPRESZ, BUNRI ;

output         VPDACT ;
output         UDL1_VPCS ;

wire   [13:0]  VPA ;

★       assign  VPDACT          = ~( UDL1_VPCS ) ;
        assign  UDL1_VPCS       =   ( UDL1_ADR == VPA[13:2] ) & ~BUNRI ;

endmodule
```

★

Figure 6-4. Example of User Logic HDL Creation

```
module NPB_UDL1(
                    VPA, VPDR, VPDW, VPCS, VPWRITE, VPSTB, VPUBENZ,
                    VPRESZ, BUNRI
                    ) ;

input    [1:0]      VPA ;
input    [15:0]     VPDW ;
input               VPCS, VPWRITE, VPSTB, VPUBENZ, VPRESZ, BUNRI ;

output   [15:0]     VPDR ;

reg      [15:0]     dout ;

//-- VPDR driver --

wire     vpdrl_en, vpdrh_en ;

         assign  vpdrl_en = VPCS & ~VPWRITE & VPSTB & ~VPA[0] & ~BUNRI ;
         assign  vpdrh_en = VPCS & ~VPWRITE & VPSTB & ~VPUBENZ & ~BUNRI ;

         assign  VPDR[7:0] = ( vpdrl_en ) ? dout[7:0] : 8'bzzzz_zzzz ;
         assign  VPDR[15:8] = ( vpdrh_en ) ? dout[15:8] : 8'bzzzz_zzzz ;

//-- User Logic Registor --

reg      [7:0]      udl11, udl12 ;
reg      [15:0]     udl13 ;

         // synopsys async_set_reset "VPRESZ"

         always @( VPSTB or VPRESZ ) begin
                 if ( ~VPRESZ ) begin
                         dout    <= 16'h0000 ;
                         udl11   <= 8'h00 ;
                         udl12   <= 8'h00 ;
                         udl13   <= 16'h0000 ;
                 end
                 else begin
                         if ( VPSTB & VPCS ) begin
                                 if ( ~VPA[1] ) begin
                                         if ( ~VPWRITE ) begin       // read action
                                                 dout  <= { udl12, udl11 } ;
                                         end
                                         else begin                  // write action
                                                 if ( VPA[0] & ~VPUBENZ ) udl12 <= VPDW[15:8] ;
                                                 else if ( ~VPA[0] & VPUBENZ ) udl11 <= VPDW[7:0] ;
                                                 else if ( ~VPA[0] & ~VPUBENZ ) begin
                                                         udl12 <= VPDW[15:8] ;
                                                         udl11 <= VPDW[7:0] ;
                                                 end
                                         end
                                 end
                                 else begin
                                         if ( ~VPWRITE ) dout <= udl13 ;
                                         else            udl13 <= VPDW ;
                                 end
                         end
                 end
         end

endmodule
```

Figure 6-5. User Logic Operation Timing Chart



Notes 1. This is an NB85E signal.

2. These are internal signals of user logic (UDL1).

Remark Input and output are shown as seen from the user logic (UDL1).

(3) Application example (example of creating user logic with retry function)

To use the user logic shown in the above design example as an intermediate buffer between a very low-speed macro and the NPB, a retry function can be attached to the user logic.

Figure 6-6 shows an example of creating HDL for user logic with a retry function and Figure 6-7 shows an operation timing chart.

This example is designed so that a wait of one VPRETR signal is inserted when reading UDL13.

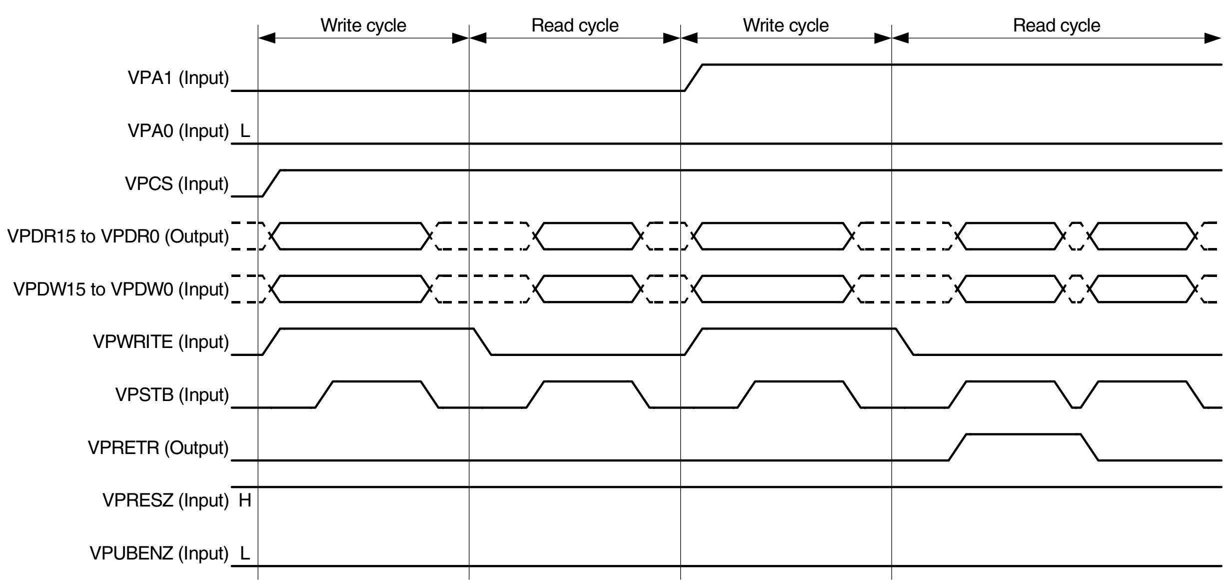
Figure 6-6. Example of HDL Creation for User Logic with Retry Function (1/2)

```
module NPB_UDL2(
                    VPA, VPDR, VPDW, VPCS, VPWRITE, VPSTB, VPUBENZ,
                    VPRESZ, VPRETR, BUNRI
                    ) ;

input     [1:0]     VPA ;
input     [15:0]    VPDW ;
input               VPCS, VPWRITE, VPSTB, VPUBENZ, VPRESZ, BUNRI ;

output    [15:0]    VPDR ;
output              VPRETR ;

reg       [15:0]    dout ;
reg                 VPRETR ;

//-- VPDR driver --

wire      vpdrl_en, vpdrh_en ;

          assign    vpdrl_en = VPCS & ~VPWRITE & VPSTB & ~VPA[0] & ~BUNRI ;
          assign    vpdrh_en = VPCS & ~VPWRITE & VPSTB & ~VPUBENZ & ~BUNRI ;

          assign    VPDR[7:0] = ( vpdrl_en ) ? dout[7:0] : 8'bzzzz_zzzz ;
          assign    VPDR[15:8] = ( vpdrh_en ) ? dout[15:8] : 8'bzzzz_zzzz ;

//-- User Logic Registor --

reg       [7:0]     udl11, udl12 ;
reg       [15:0]    udl13 ;

reg                 retract ;

          // synopsys async_set_reset "VPRESZ"

          always @( VPSTB or VPRESZ ) begin
                    if ( ~VPRESZ ) begin
                              dout      <= 16'h0000 ;
                              udl11     <= 8'h00 ;
                              udl12     <= 8'h00 ;
★                             udl13     <= 16'h000 ;
                              VPRETR <= 1'b0 ;
                    end
                    else begin
                              if ( VPSTB & VPCS ) begin
                                        if ( ~VPA[1] ) begin
                                                  if ( ~VPWRITE ) begin        // read action
                                                            dout  <= { udl12, udl11 } ;
                                                  end
                                                  else begin                   // write action
                                                            if ( VPA[0] & ~VPUBENZ ) udl12 <= VPDW[15:8] ;
                                                            else if ( ~VPA[0] & VPUBENZ ) udl11 <= VPDW[7:0] ;
                                                            else if ( ~VPA[0] & ~VPUBENZ ) begin
                                                                      udl12 <= VPDW[15:8] ;
                                                                      udl11 <= VPDW[7:0] ;
                                                            end
                                                  end
                                        end
```

Figure 6-6. Example of HDL Creation for User Logic with Retry Function (2/2)

```
                    else begin
                            if ( ~VPWRITE ) begin
                                    VPRETR <= retract ;
                                    Dout      <= udl13 ;
                            end
                            else    udl13     <= VPDW ;
                    end
               end
          end
     end
     always @( posedge VPSTB or negedge VPRESZ ) begin
          if ( ~VPRESZ ) retract <= 1'b0 ;
          else if ( ~VPWRITE & VPCS & VPA[1] ) retract <= ~retract ;
     end
endmodule
```

Figure 6-7. Operation Timing Chart of User Logic with Retry Function



Remarks 1. Levels of broken-line portions indicate an undefined status in which the (NB85E internal) bus holder is driving (Weak unknown).

2. Input and output are shown as seen from the user logic (UDL1).

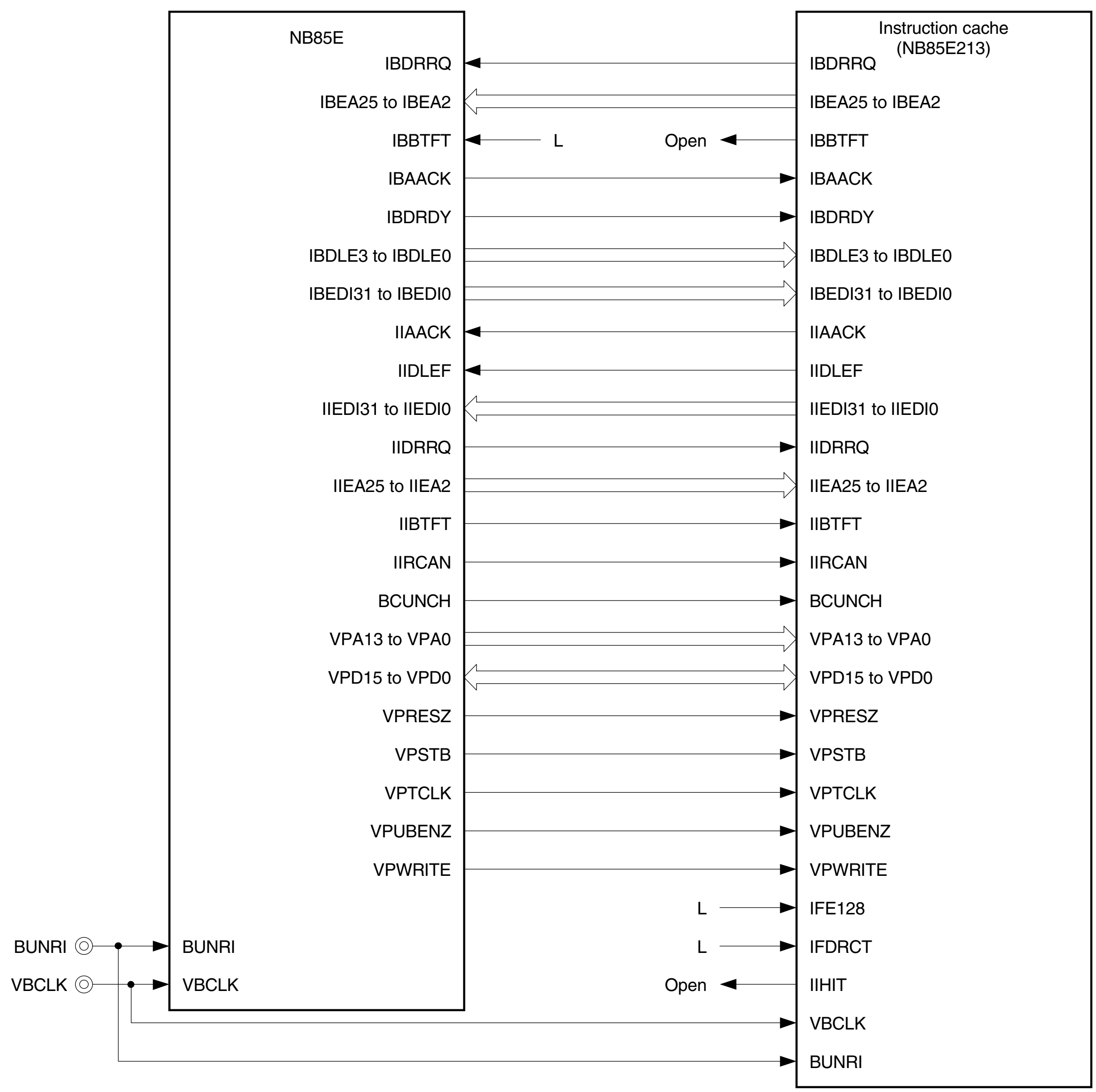
CHAPTER 7 CACHE CONNECTION

7.1 Connection of Instruction Cache

The following products are instruction caches that can be connected to the NB85E.

- NB85E212 ... 4 KB 2-way set associative instruction cache (4 words × 128 entries × 2 ways = 4 KB)
- NB85E213 ... 8 KB 2-way set associative instruction cache (4 words × 256 entries × 2 ways = 8 KB)

★ **Figure 7-1. Example of Connecting Instruction Cache (NB85E213) to NB85E**



Remark L: Low level input

7.1.1 Cautions when using instruction cache

(1) Connection to NB85E

Connect pins that have the same pin name. However, leave the IBBTFT pin of the instruction cache open and fix the IBBTFT pin of the NB85E to low level.

(2) Cache type selection pin setup

Input the following levels to cache type selection pins whose pin names begin with “IF”.

Pin Name	Input Level	
	NB85E212	NB85E213
IFE128	High level	Low level
IFDRCT	Low level	Low level

(3) Bus cycle status

In an area for which the instruction cache setting is set to cacheable by the cache configuration register (BHC) of the NB85E, the VBCTYP2 to VBCTYP0 signals of the NB85E always indicate a normal operation code fetch and do not indicate an operation code fetch of the target address in a branch instruction.

(4) Operation on reset

On a reset, a tag is automatically cleared (invalidated) and the next data replacement is performed from way 0. Therefore, if there is an access to the instruction cache in a period of as many clock cycles as the number of lines after reset, the CPU stops until the tag is cleared (becomes invalid).

(5) Register settings

Be sure to set the NB85E registers shown below in non-cacheable areas. However, set bit 4 of the instruction cache control register (ICC) in a cacheable area.

- Chip area select control registers (CSC0, CSC1)
- Peripheral I/O area select control register (BPC)
- Bus size configuration register (BSC)
- Endian configuration register (BEC)
- Cache configuration register (BHC)
- Instruction cache control register (ICC)[Note]
- Instruction cache data configuration register (ICD)

Note Excluding bit 4

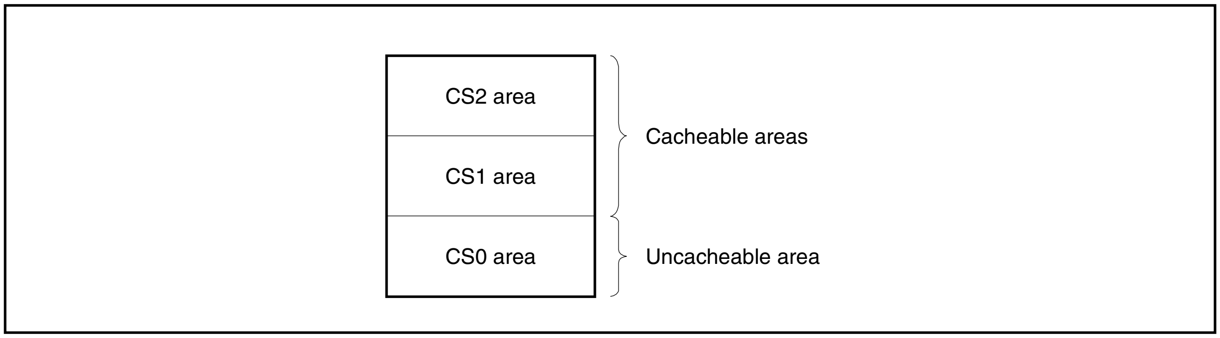
(6) Access to memory boundaries

If adjoining chip select (CSn) areas are a cacheable area and a non-cacheable area, the memory boundary between them can be continuously accessed only by a branch instruction (n = 7 to 0). Operation is not guaranteed if this memory boundary is continuously accessed by other than a branch instruction. An example is shown below.

Example Set cache areas as shown in Figure 7-2. In this case, access to memory areas is as follows.

- From CS0 area to CS1 area is only accessible using a branch instruction.
- From CS1 area to CS2 area is continuously accessible.

Figure 7-2. Cache Area Setting Example



★ **(7) Initial program settings**

Always execute the following instruction before setting the BHC register of the NB85E with the initial settings of the user program immediately following system reset.

```
st.h   r0, 0xfffff072[r0]
```

Following execution of this instruction, the cache is enabled by setting “cacheable” (BHn0 bit = 1) as the instruction cache setting with the BHC register (n = 7 to 0).

★ **(8) Setting BHC register of NB85E**

In the case of CSn areas for which an instruction to set the BHC register exists, cacheable/uncacheable settings for the instruction cache using this instruction cannot be performed (n = 7 to 0). Instruction cache cacheable/uncacheable settings are possible only for CSn areas for which no instruction for setting the BHC register exists.

For example, if a BHC register setting instruction exists in the CS0 area, the instruction cache of the CS0 area cannot be set (cacheable/uncacheable settings). In this case, only the instruction cache settings for areas CS1 to CS7 are possible.

However, instruction cache settings for all CSn areas from instructions that exist in memory areas connected to VFB or VDB are possible.

Remark VFB: Dedicated bus used to directly connect ROM (V850E fetch bus)
VDB: Dedicated bus used to directly connect RAM (V850E data bus)

★ **(9) Test bus auto wiring tool support**

This instruction cache does not support test bus auto wiring tools because although it has a BUNRI pin, it does not have test buses (TBOx, TBIx).

★ **(10) Tag clear procedure**

Way 0 shares the counter to clear tags with way 1.

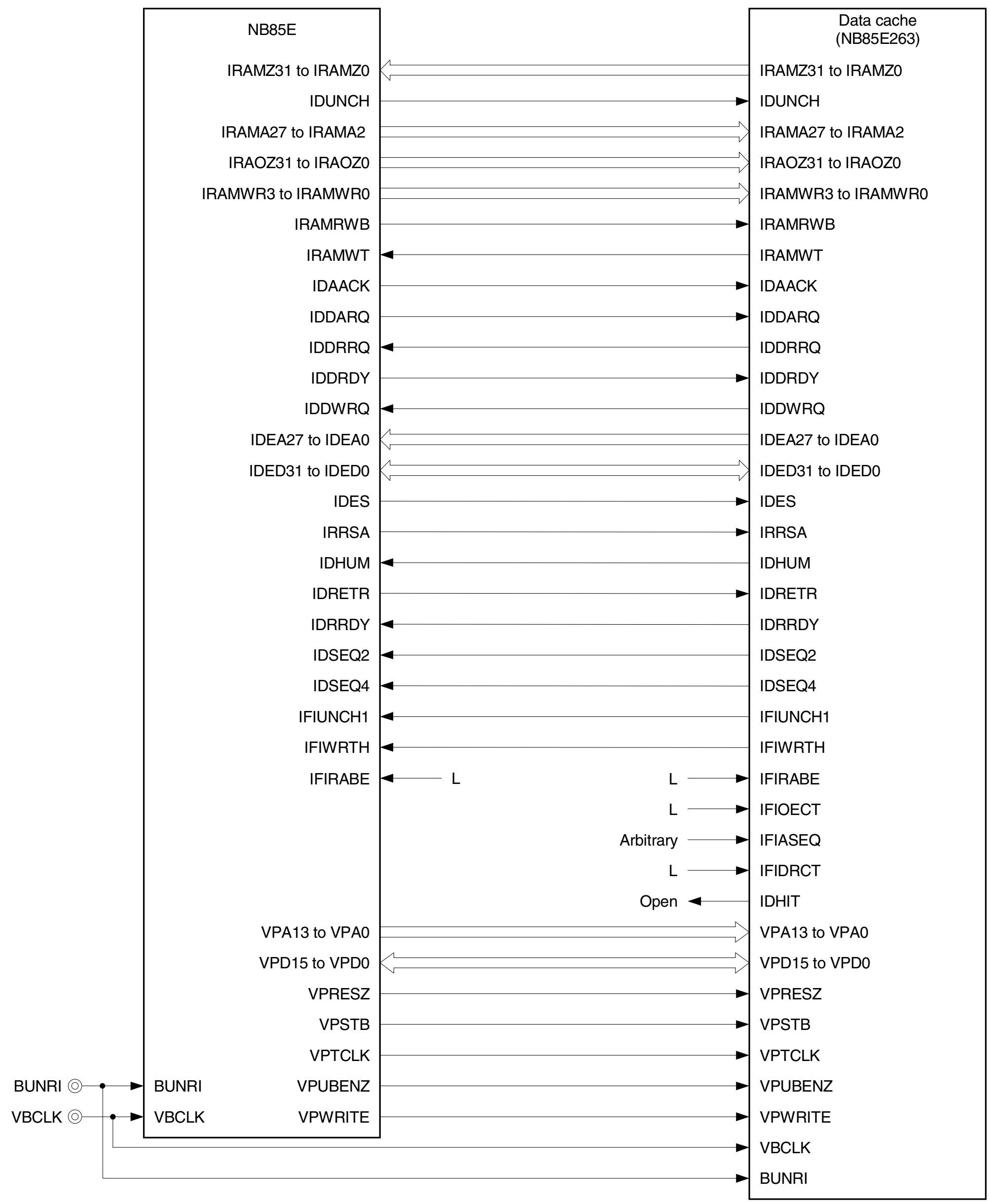
Thus, clear tags (set (1) the TCLR0 bit or TCLR1 bit of the ICC register) when the counter for tag clearing is stopped (TCLR0 = TCLR1 = 0). When clearing the tags of way 0 and way 1 individually, if tag clearing for either way is executed during tag clear execution for the other way (TCLR0 or TCLR1 = 1), the counter stops in the middle of tag clearing. Consequently, normal tag clearing cannot be performed because the counter switches to perform the other tag clear operation still indicating the value it had when stopped halfway. Be sure to confirm that tag clearing for one way is completed (TCLR0 or TCLR1 = 0) before performing tag clearing for the other way.

7.2 Connection of Data Cache

The following products are data caches that can be connected to the NB85E.

- NB85E252 ... 4 KB direct map data cache (4 words × 256 entries = 4 KB)
- NB85E263 ... 8 KB 2-way set associative data cache (4 words × 256 entries × 2 ways = 8 KB)

★ **Figure 7-3. Example of Connecting Data Cache (NB85E263) to NB85E**



Caution **Since the data cache is connected to the NB85E using the V850E data bus (VDB), it is not possible to use RAM and the data cache simultaneously. Connect either the RAM or the data cache to the VDB.**

Remark L: Low level input

7.2.1 Cautions when using data cache

(1) Connection to NB85E

Connect pins that have the same pin names.

(2) Cache type selection pin setup

Input the levels shown below to the cache type selection pins that have pin names beginning with “IFI”. Connect the IFIUNCH1 and IFIWRTH pins to the NB85E.

Pin Name	Input Level	
	NB85E252	NB85E263
IFIASEQ	Arbitrary[Note 1]	Arbitrary[Note 1]
IFIRABE	Low level	Low level
IFIDRCT[Note 2]	–	Low level
IFIOECT	Low level	Low level

Notes 1. Set IFIASEQ depending on the system. For details, refer to the **Instruction Cache, Data Cache NB85E, NB85ET User’s Manual (A14247E)**.

2. NB85E263 only

(3) Bus cycle status

In all read cycles of an area for which the data cache setting is set to cacheable by the cache configuration register (BHC) of the NB85E and in write cycles when in write back mode (write allocate enabled), the VBCTYP2 to VBCTYP0 signals of the NB85E always indicate a data access and do not indicate a misalign access.

(4) Operation on reset

On a reset, a tag is automatically cleared (invalidated) and the next data replacement is performed from way 0. Therefore, if there is an access to the data cache in a period of as many clock cycles as the number of lines after reset, the CPU stops until the tag is cleared (becomes invalid).

★ **(5) Test bus auto wiring tool support**

This data cache does not support test bus auto wiring tools because although it has a BUNRI pin, it does not have test buses (TBOx, TBIx).

★ **(6) Other**

This data cache does not have a bus snoop circuit (which monitors the bus operation). Note that data in the data cache in the cases shown in the following examples is dirty data even when there is no write access to the data cache, and is data that has lost its coherency. To avoid this status, be sure to clear tags.

Examples 1. When DMA transfer is performed to the external memory of a cacheable area from RAM (Transfer data is not reflected in the data cache)

2. When the external memory contents of a cacheable area are overwritten by the external bus master

★ **(7) Operation during debugging**

The data cache does not operate during debugging using an N-wire type in-circuit emulator.

Since access is performed directly only to external memory even if the data cache is enabled, when accessing external memory in a cacheable area during debugging, coherency is degraded. To prevent this, be sure to perform data cache tag clearing. Also, when using an in-circuit emulator (IE-V850E-MC-A), data cache debugging cannot be performed.

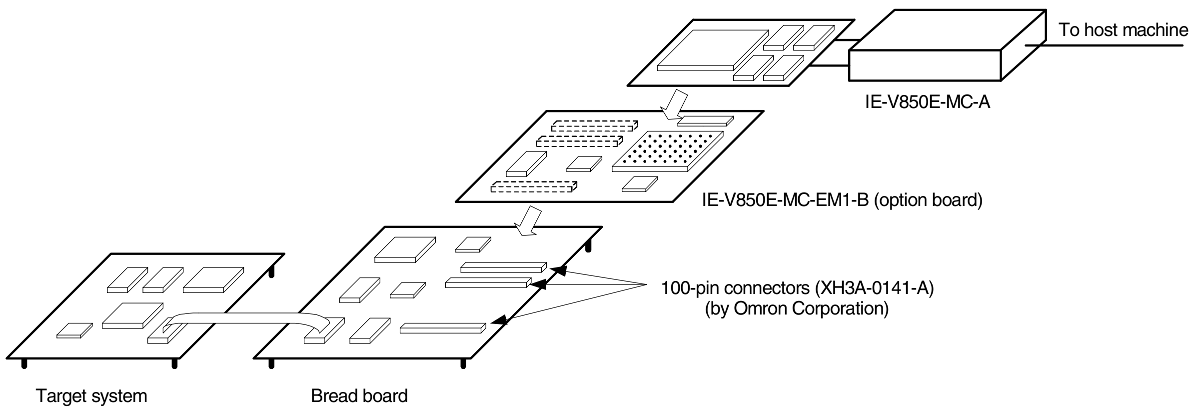
CHAPTER 8 CONNECTION TO IN-CIRCUIT EMULATOR (IE)

8.1 Using In-Circuit Emulator for NB85E (IE-V850E-MC-A)

When using an in-circuit emulator for an NB85E (IE-V850E-MC-A), three 100-pin connectors[Note] must be mounted on a breadboard in order to connect the in-circuit emulator for NB85E (IE-V850E-MC-A) to the option board (IE-V850E-MC-EM1-B) (for connector placement and dimensions and a signal list, refer to the **IE-V850E-MC-EM1-B, IE-V850E-MC-MM2 User's Manual (U14482E)**).

Note Recommended connector: XH3A-0141-A (by Omron Corporation)

Figure 8-1. Connection of In-Circuit Emulator for NB85E (IE-V850E-MC-A)



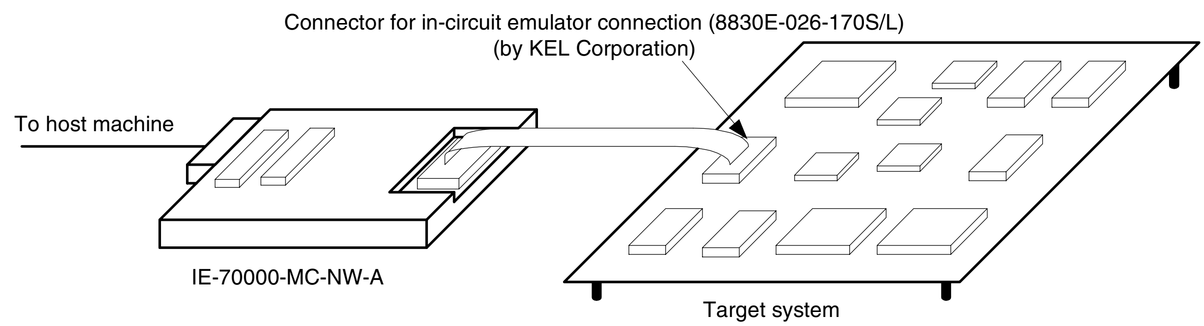
Cautions 1. Since instruction and data cache pins, test mode pins, VFB and VDB pins, RCU pins, and peripheral evaluation chip mode pins are not needed in debugging using the IE-V850E-MC-A, these pins are not on the option board.
There also is no pin for setting the operating mode on the option board, but the initial value of the operating mode can be set using the CPU control register of the IE-V850E-MC-A.

2. When performing debugging, be sure to read the restrictions attached to the in-circuit emulator main unit (IE-V850E-MC-A) and to the option board (IE-V850E-MC-EM1-B).

8.2 Using N-Wire Type In-Circuit Emulator (IE-70000-MC-NW-A)

When using an N-Wire type in-circuit emulator (IE-70000-MC-NW-A), a connector for IE connection and a circuit for connection must be implemented in the target system.

Figure 8-2. Connection of N-Wire Type In-Circuit Emulator



8.2.1 Connector for in-circuit emulator connection (in target system)

Figure 8-3 shows a pin placement diagram of a connector for in-circuit emulator connection (in target system) and Table 8-1 shows the pin functions.

Remark The recommended connectors are as follows.

- 8830E-026-170S (by KEL Corporation): 26-pin straight type
- 8830E-026-170L (by KEL Corporation): 26-pin right-angle type

Figure 8-3. Pin Placement Diagram of Connector for In-Circuit Emulator Connection (in Target System)

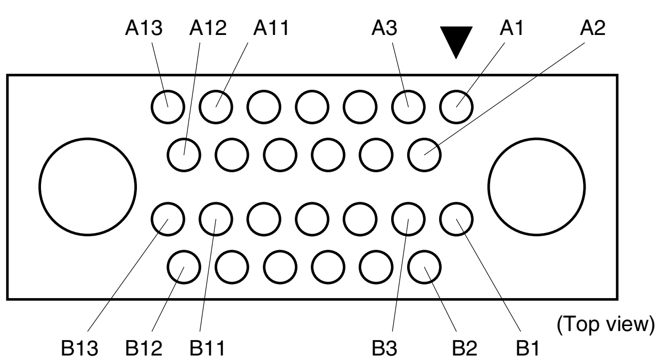


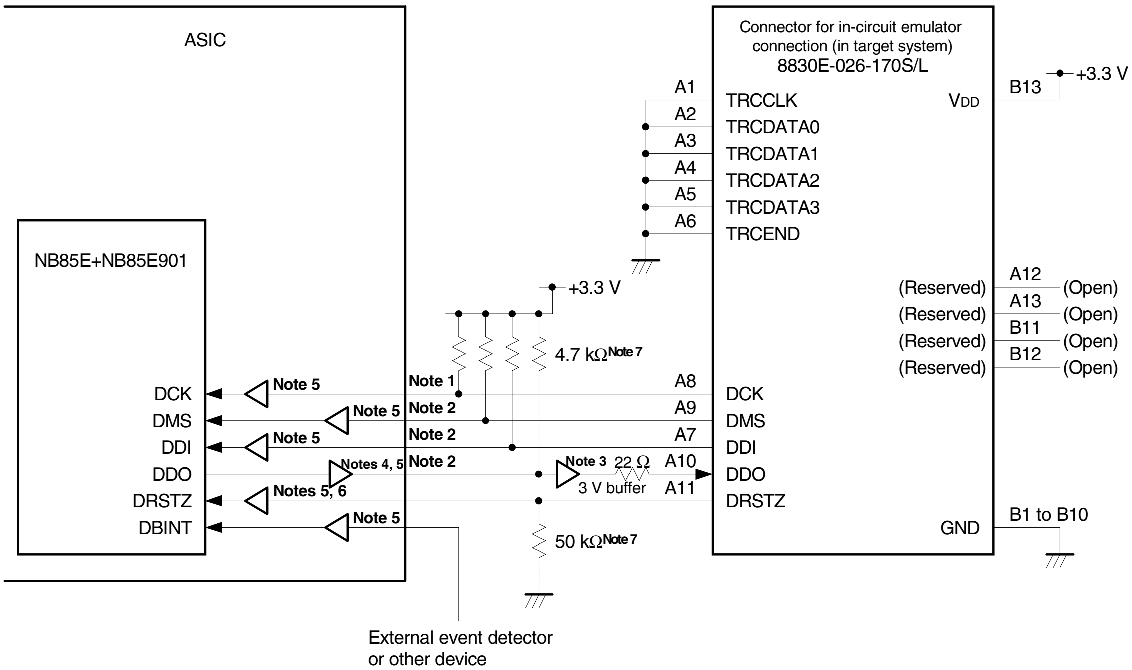
Table 8-1. Pin Functions of Connector for In-Circuit Emulator Connection (in Target System)

Pin Number	Pin Name	I/O	Pin Function
A1	TRCCLK	Input	Trace clock input
A2	TRCDATA0	Input	Trace data 0 input
A3	TRCDATA1	Input	Trace data 1 input
A4	TRCDATA2	Input	Trace data 2 input
A5	TRCDATA3	Input	Trace data 3 input
A6	TRCEND	Input	Trace data end input
A7	DDI	Output	Debug serial interface data output
A8	DCK	Output	Debug serial interface clock output
A9	DMS	Output	Debug serial interface transfer mode selection output
A10	DDO	Input	Debug serial interface data input
A11	DRSTZ	Output	DCU reset output
A12	(Reserved)	–	(Leave open)
A13	(Reserved)	–	(Leave open)
B1	GND	–	–
B2	GND	–	–
B3	GND	–	–
B4	GND	–	–
B5	GND	–	–
B6	GND	–	–
B7	GND	–	–
B8	GND	–	–
B9	GND	–	–
B10	GND	–	–
B11	(Reserved)	–	(Leave open)
B12	(Reserved)	–	(Leave open)
B13	V_{DD}	–	+3.3 V input (for monitoring target power application)

8.2.2 Circuit example when RCU (NB85E901) is connected to NB85E

Figure 8-4 shows a circuit example for the connector for in-circuit emulator connection (in the target system).

★ **Figure 8-4. Circuit Example for In-Circuit Emulator Connection (NB85E + RCU (NB85E901))**



Notes 1. Make the clock pattern length the shortest and shield by surrounding with GND. Make the pattern length such that it does not exceed 100 mm.

2. Make the pattern length as short as possible. Make the pattern length such that it does not exceed 100 mm.

3. Recommended buffer: SN74LVC541A (by TI Corporation) or TC74LCX541F (by Toshiba Corporation)

4. A type having 6 mA or more drive power is recommended for the pin output buffer.

5. When using an NEC in-circuit emulator (IE-70000-MC-NW-A), use a 3.3 V buffer. (To use a third-party in-circuit emulator, consult the tool vendor.)

6. Recommended buffer: A Schmitt type input buffer is recommended for the DRSTZ.

7. The resistors used for pulling up/down outside the ASIC can be provided inside the ASIC without problem.

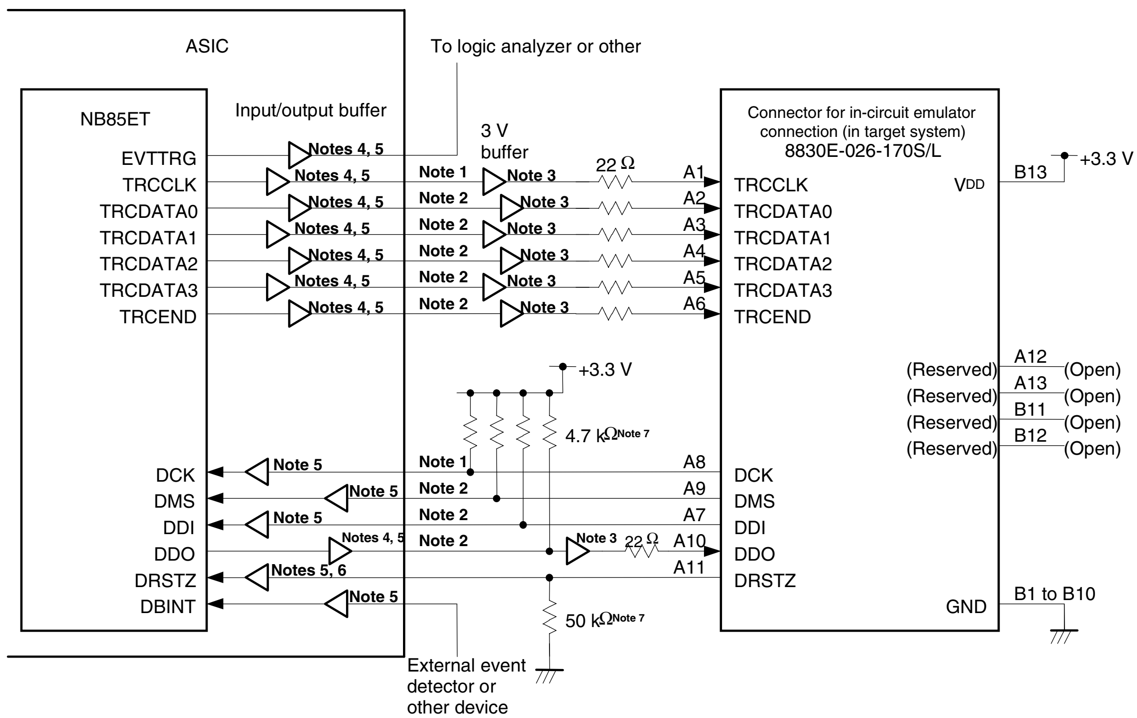
Remarks 1. The V_{DD} (pin B13) of the connector for in-circuit emulator connection (in the target system) is used only for detecting whether power is applied to the target system.

2. The DBINT pin is optional. When there is no need to input debug interrupts from outside, input a low level.

8.2.3 Circuit example when NB85ET is connected

Figure 8-5 shows a circuit example for the connector for in-circuit emulator connection (in the target system).

★ **Figure 8-5. Circuit Example for In-Circuit Emulator Connection (NB85ET)**



Notes 1. Make the clock pattern length the shortest and shield by surrounding with GND. Make the pattern length such that it does not exceed 100 mm.

2. Make the pattern length as short as possible. Make the pattern length such that it does not exceed 100 mm.

3. Recommended buffer: SN74LVC541A (by TI Corporation) or TC74LCX541F (by Toshiba Corporation)

4. A type having 6 mA or more drive power is recommended for the pin output buffer.

5. When using an NEC in-circuit emulator (IE-70000-MC-NW-A), use a 3.3 V buffer. (To use a third-party in-circuit emulator, consult the tool vendor.)

6. Recommended buffer: A Schmitt type input buffer is recommended for the DRSTZ.

7. The resistors used for pulling up/down outside the ASIC can be provided inside the ASIC without problem.

Remarks 1. The V_{DD} (pin B13) of the connector for in-circuit emulator connection (in the target system) is used only for detecting whether power is applied to the target system.

2. The DBINT pin is optional. When there is no need to input debug interrupts from outside, input a low level.

3. The EVTTRG pin is optional. It is mainly used as the trigger output for measurement devices such as a logic analyzer. When trigger output is not needed, leave it open.

★ 8.2.4 Design of timing with N-wire type in-circuit emulator

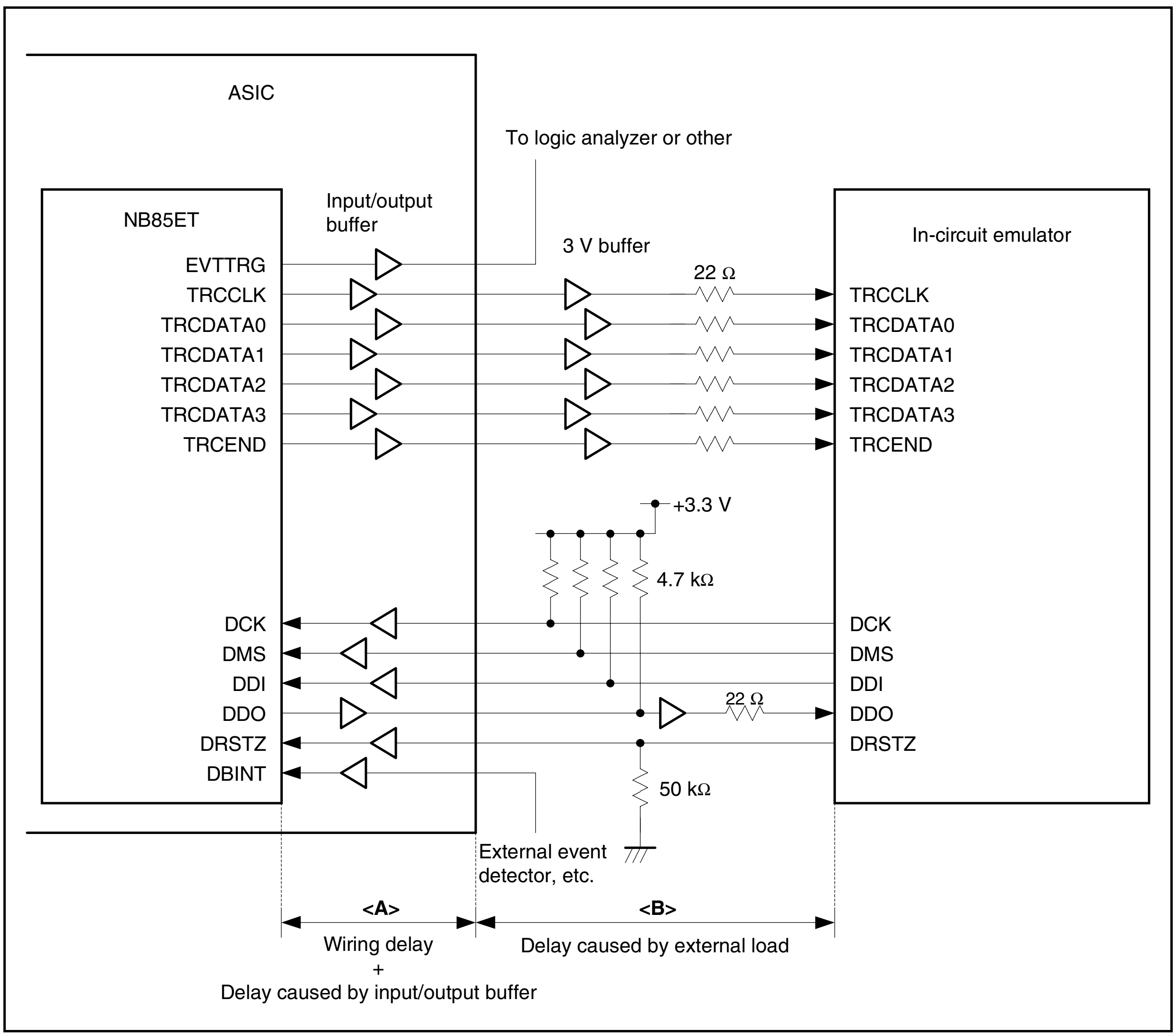
Design the timing between the NB85ET (or NB85E901) and N-wire type in-circuit emulator so as to satisfy the NB85ET (or NB85E901) and N-wire type in-circuit emulator timing specs, taking into consideration the following two delays.

- Wiring delay + delay caused by output buffer
  (From NB85ET (or NB85E901) macro to ASIC pins (refer to **<A>** in **Figure 8-6**))
- Delay caused by external load
  (From ASIC pins to N-wire type in-circuit emulator (refer to **<B>** in **Figure 8-6**))

Remark For the NB85ET (or NB85E901) timing specs, refer to the **CB-9 Family VX/VM Type CPU Core, Memory Controller Design Manual (A13195E)**. For the N-wire type in-circuit emulator specs, consult NEC if using the IE-70000-MC-NW-A, or the third party tool vendor if using an N-wire type in-circuit emulator made by a company other than NEC.

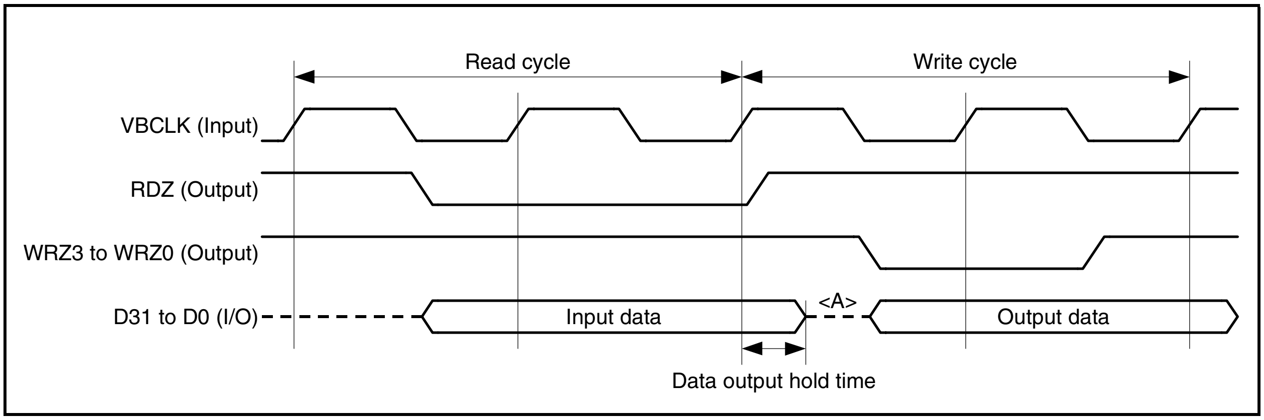
Figure 8-6 shows the delays between the NB85ET and N-wire type in-circuit emulator.

Figure 8-6. Delays Between NB85ET and N-Wire Type In-Circuit Emulator



9.1 Bus Contention When Moving from Read Cycle to Write Cycle (Using MEMC (NB85E500))

In cases in which a write cycle occurs after a read cycle (such as when writing data read from external ROM to external RAM), the bus timing is normally as follows.

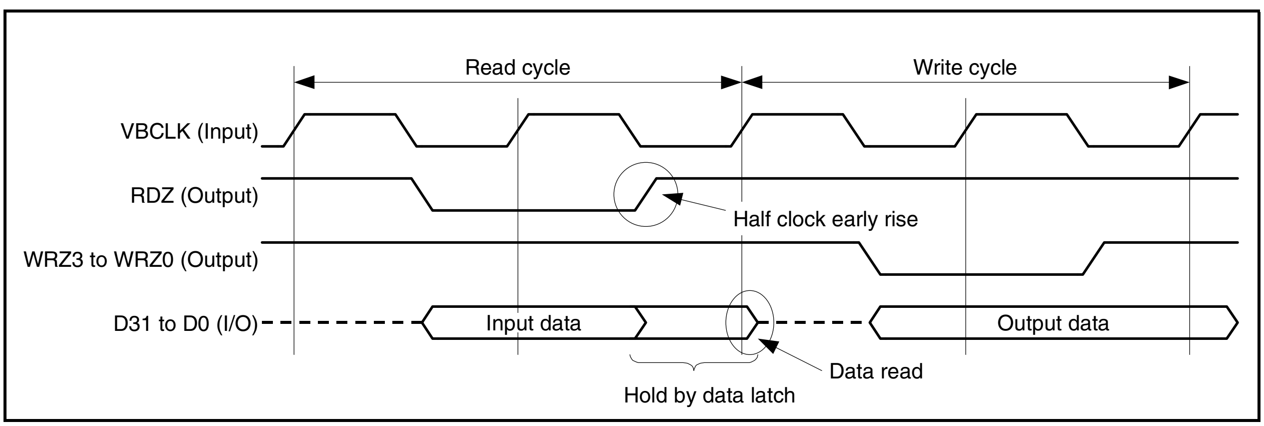


In this case, if the external ROM data output hold time is one half clock of VBCLK or longer, the input data from the external ROM and the output data to the external RAM may conflict (<A> in the figure above).

The following two countermeasures are available for avoiding this.

(1) Insert an idle cycle according to the setting of the bus cycle control register (BCC). Set it so that an idle cycle is inserted behind the read cycle. However, the performance of ROM access (read cycle) declines.

(2) Raise a read strobe signal (RDZ) one half clock early. However, since the NB85E data read is at the rise of the last clock of the read cycle, it is necessary to add a data latch inside the chip to prevent data deletion.



9.2 Cautions on Verilog Simulation

To set data in general-purpose register “reg1” using the instruction “MOV imm32, reg1” in the Verilog simulation model of the NB85E, the write destination reg1 (all bits) must be initialized to “0” in advance. Note that if reg1 contains an undefined value (“x” or “z”), the result of executing the instruction is not correctly reflected in reg1 (restriction).

This occurs only in Verilog simulation and is not a problem in the actual chip. In addition, other MOV instructions (“MOV reg1, reg2”, “MOV imm5, reg2”) do not have such a restriction.

The measure taken to avoid this is to initialize all general-purpose registers at the beginning of the initialization routine. Initialize all registers using the program shown below before using registers.

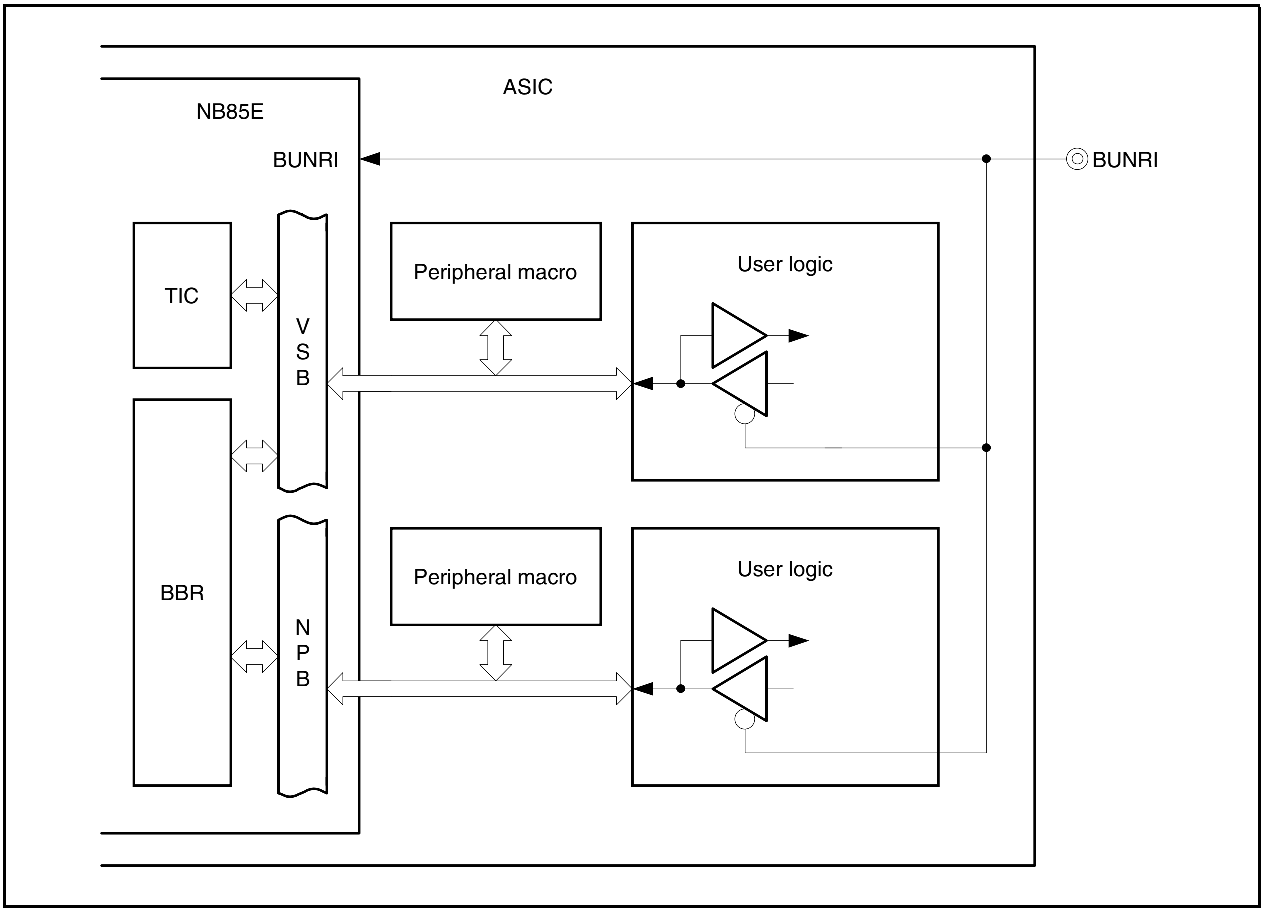
```
mov  r0, r1
mov  r0, r2
mov  r0, r3
mov  r0, r4
mov  r0, r5
mov  r0, r6
mov  r0, r7
mov  r0, r8
mov  r0, r9
mov  r0, r10
mov  r0, r11
mov  r0, r12
mov  r0, r13
mov  r0, r14
mov  r0, r15
mov  r0, r16
mov  r0, r17
mov  r0, r18
mov  r0, r19
mov  r0, r20
mov  r0, r21
mov  r0, r22
mov  r0, r23
mov  r0, r24
mov  r0, r25
mov  r0, r26
mov  r0, r27
mov  r0, r28
mov  r0, r29
mov  r0, r30
mov  r0, r31
```

9.3 Cautions on BUNRI-Testing Chip

When BUNRI-testing a chip, ensure that user logic (macro that does not have a test function) is not selected.

However, for a macro in which a test function is implemented, it is necessary to ensure that it is selected even at the time of BUNRI testing. For details, see **CHAPTER 10 TEST CIRCUIT DESIGN**.

Figure 9-1. User Logic Design Example



9.4 Timing Adjustment

When designing a circuit that performs high-speed operations, skew adjustment and hold time adjustment is required.

If there is a problem with skew or hold time, the following symptoms occur in simulation.

In particular, clock skew and data line hold timing require caution.

- After reset release, all is normal until an instruction fetch and all is undefined after several instructions are read from ROM.
- Data read from outside takes invalid values.
- Simulation cannot be performed at all (all output pins are undefined).
- A “Timing Violation” occurs (particularly a “$hold” error).

9.4.1 Adjustment of clock skew or data line hold violation

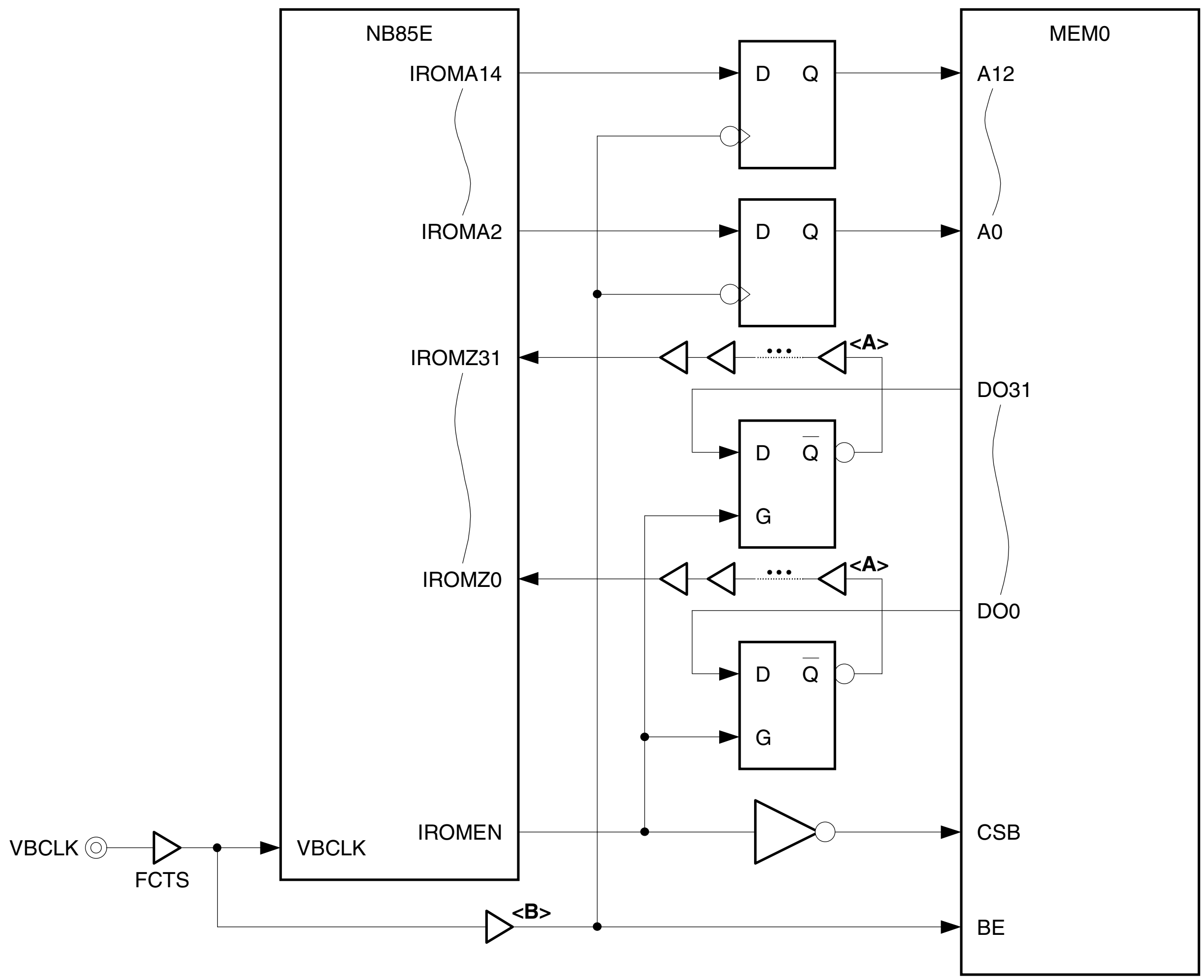
To guarantee the clock skew or data line hold time, use the following two methods together.

- Insertion of buffer in data bus (see **<A>** in **Figure 9-2**)
- Insertion of buffer in clock line (see **<B>** in **Figure 9-2**)

Remarks 1. To insert a buffer for clock skew adjustment in a circuit using HDL, specify it in the structure description.

2. For cautions on logic synthesis, refer to **NEC SYSTEM LSI DESIGN OPENCAD V5.4 Design Compiler Interface User’s Manual (A15058E).**

★ **Figure 9-2. Skew or Hold Violation Countermeasure Execution Example**

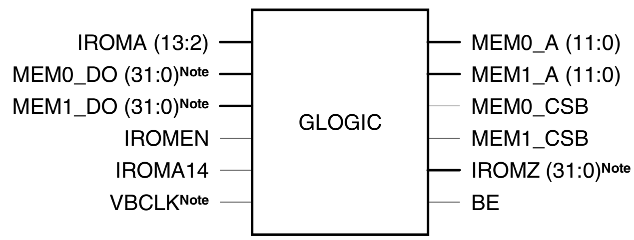


Remark This figure is an example of connecting a compiled memory to the VFB. For details, see **3.2 Connection of Compiled Memory to VFB**.

9.4.2 Resolution of hold violation using design compiler

The methods of adjusting a hold violation using the design compiler are to directly insert it in a circuit in the structure description in HDL or to describe it in a logic synthesis script. An example of a case handled using a logic synthesis script is shown below.

The module block in which portions that tie compiled memory to the CPU are cut from the circuit in Figure 9-2 is as follows (module name: GLOGIC).



Note Port related to hold adjustment

For example, the script is as follows when the CPU clock frequency is 33 MHz (period 30 ns) and an IROMZ hold time of 4 ns or more is needed for VBCLK (input_delay must also be specifically defined). In this example, a 5 ns to 7 ns delay is inserted in IROMZ.

```
/* Set the current_design */
current_design GLOGIC

create_clock –period 30 –waveform {0 15} find (port, “VBCLK”)
set_fix_hold find (clock, “VBCLK”)
set_output_delay 23 –max –clock “VBCLK” find (port, “IROMZ*”)
set_output_delay –5 –min –clock “VBCLK” find (port, “IROMZ*”)
```

9.4.3 Clock skew adjustment

If the clock input to the BE pin is standardized to the clock signal input to the CPU (VBCLK), there is clock skew. This section describes how to calculate the clock skew adjustment taking the case of connecting compiled ROM to the VFB as an example.

(1) How to calculate clock skew (skew of BE signal against VBCLK signal)

From the relationship of the compiled ROM cycle time (t_{RC}), access time (t_{ACC}), and output hold time (t_{OH}) to the CPU clock frequency (f_{VBCLK}), data input setup time (t_{SIDK}), and hold time (t_{HKID}), the degree to which skew of the BE signal against the VBCLK signal is permitted can be found using the following calculation.

$\Delta t_{sk}^{-} < t_{OH} - t_{HKID}$: Data hold condition

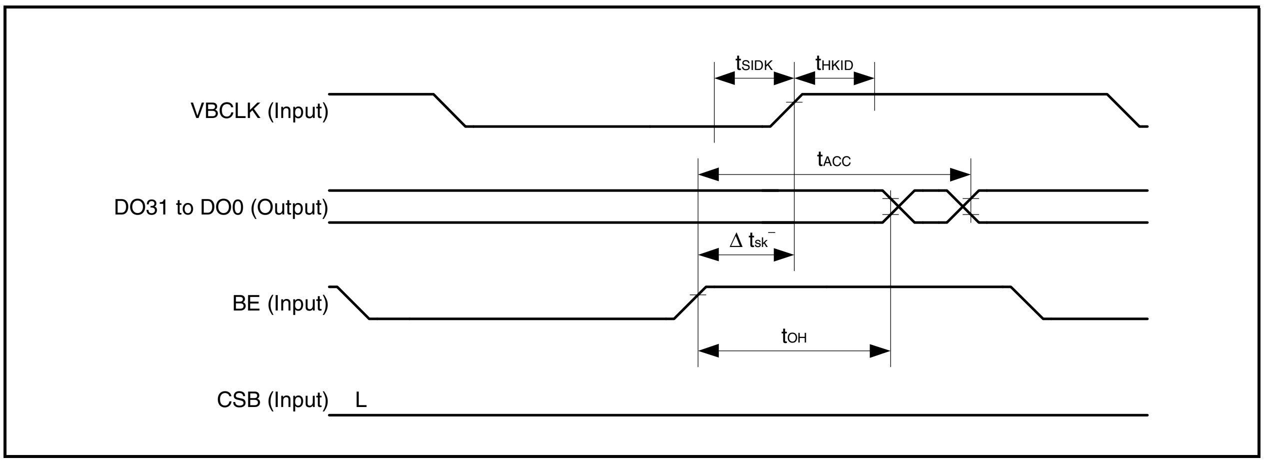
$\Delta t_{sk}^{+} < (1/f_{VBCLK} - t_{ACC}) - t_{SIDK}$: Data setup condition

(However, $t_{RC} < 1/f_{VBCLK}$)

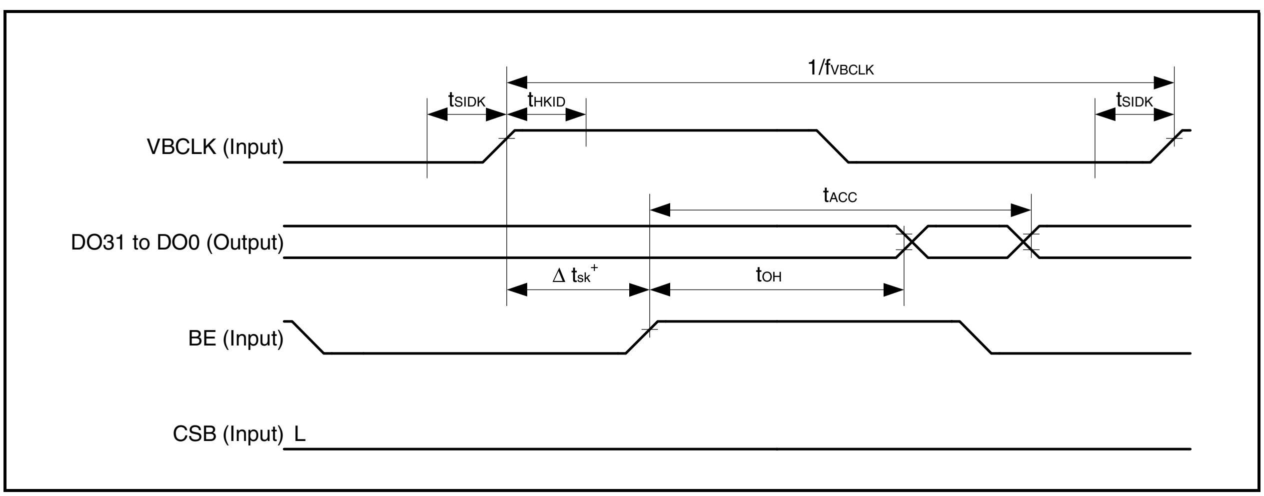
The meaning of each symbol is as follows.

t_{sk}^{-}: Value when BE skew against VBCLK is proceeding
t_{sk}^{+}: Value when BE skew against VBCLK is lagging
t_{OH}: ROM output hold time
t_{RC}: ROM cycle time
t_{ACC}: ROM access time
t_{SIDK}: CPU data input setup time
t_{HKID}: CPU data input hold time
f_{VBCLK}: CPU clock frequency (Hz)

(a) Data hold condition ($\Delta t_{sk}^{-} < t_{OH} - t_{HKID}$: however, $t_{RC} < 1/f_{VBCLK}$)



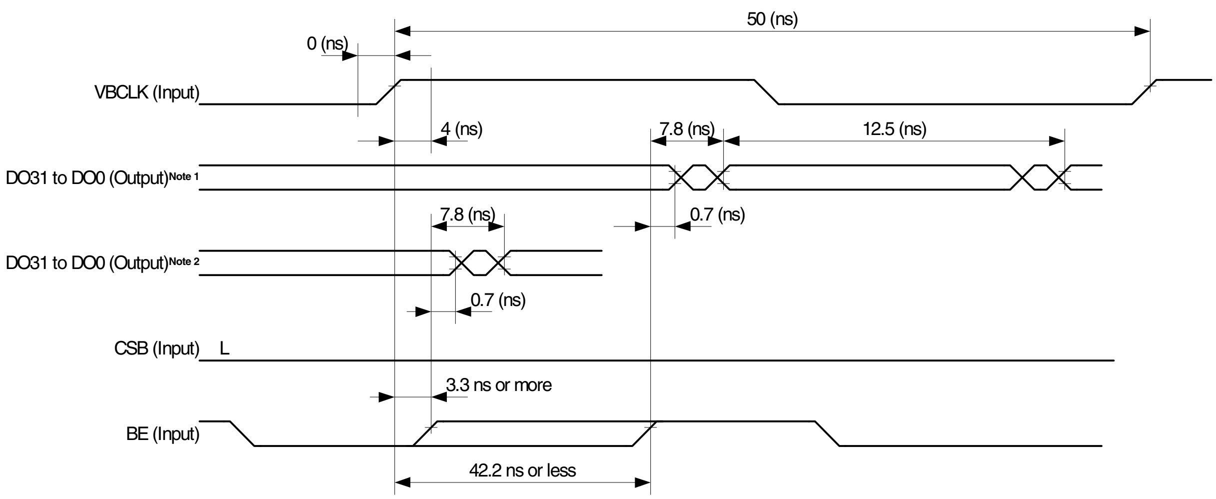
(b) Data setup condition ($\Delta t_{sk}^{+} < (1/f_{VBCLK} - t_{ACC}) - t_{SIDK}$: however, $t_{RC} < 1/f_{VBCLK}$)



(2) Calculation example of permissible value of clock skew (skew of BE signal against VBCLK signal)

For example, for a CB-9 Family VX Type (V_{DD} = 3.3 V) 32-bit × 4 Kword fast synchronous ROM, if t_{OH} = 0.7 ns, t_{ACC} = 7.8 ns, t_{RC} = 12.5 ns, VBCLK frequency (f_{VBCLK}) = 20 MHz (period: 50 ns), t_{SIDK} = 0 ns, t_{HKID} = 4 ns and the wiring delay is made small enough to ignore, the calculation is as follows.

$\Delta t_{sk}^{-} < t_{OH} - t_{HKID} = 0.7 - 4 = -3.3$ (ns)
$\Delta t_{sk}^{+} < (1/f_{VBCLK} - t_{ACC}) - t_{SIDK} = (50 - 7.8) - 0 = 42.2$ (ns)
(However, $t_{RC} < 50$ (ns))



Notes 1. DO31 to DO0 output example to satisfy data setup
2. DO31 to DO0 output example to satisfy data hold

From this result, it is known that the BE signal must be slowed at least 4 ns (against VBCLK).
The following measures can be taken if the skew timing is not suitable.

- Extend the output hold time by inserting a buffer for delay adjustment in memory data output (**<A>** in **Figure 9-2**).
- Insert a buffer for delay adjustment to slow BE signal input to memory against timing of VBCLK input to NB85E (**<B>** in **Figure 9-2**).

9.5 Device File

A device file is needed when using a C compiler/assembler (CA850), integrated debugger (ID850), or system simulator (SM850).

This device file is provided by NEC, but its contents differ depending on the development environment and type. Therefore, consult with NEC in advance concerning the following items.

(1) Development environment

Manufacturer and name of development tools (compiler, debugger, and in-circuit emulator) to be used.

★ **(2) Product specifications**

(a) CPU core (NB85E, NB85E + NB85E901, or NB85ET)
(b) Memory controllers (NB85E500, NU85E500, or NU85E502)
(c) Size of RAM to connect to VDB
(d) Size of ROM to connect to VFB
(e) Settings of IFIROME, IFIROB2, IFIRA64, IFIRA32, IFIRA16, IFIMAEN, IFID256, IFINSZ1, IFINSZ0, IFIWRTH, IFIUNCH1, and IFIUNCH0 pins
(f) Address assigned to each register name of user logic

CHAPTER 10 TEST CIRCUIT DESIGN

Since the NB85E has an on-chip test interface control unit (TIC), tests of the NB85E itself and peripheral macros (such as instruction cache, data cache, and memory controller (MEMC)) are performed via the test bus (TBI39 to TBI0, TBO34 to TBO0).

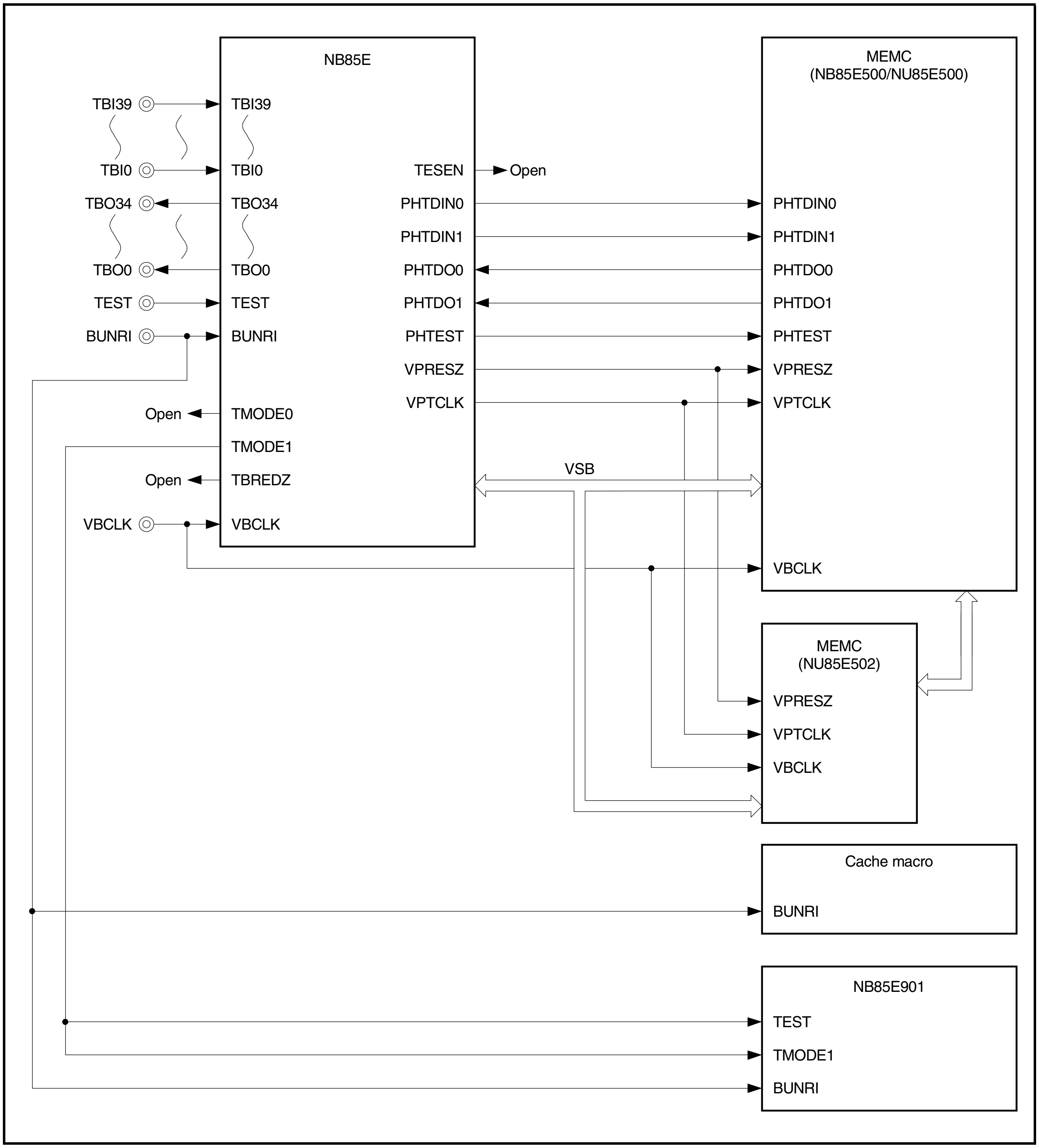
The test bus is activated when the TEST signal or BUNRI signal is active.

Tests for compiled memory connected to the VFB and VDB are performed using the test bus of that memory.

10.1 Peripheral Macro Connection Example in Test Mode

Figure 10-1 shows an example of connecting a peripheral macro to the NB85E.

★ **Figure 10-1. Example of Connecting Peripheral Macro to NB85E**



10.2 Processing Pins in Test Mode

10.2.1 Processing NB85E and NB85ET pins

(1) Other than test mode pins

(a) I/O pins

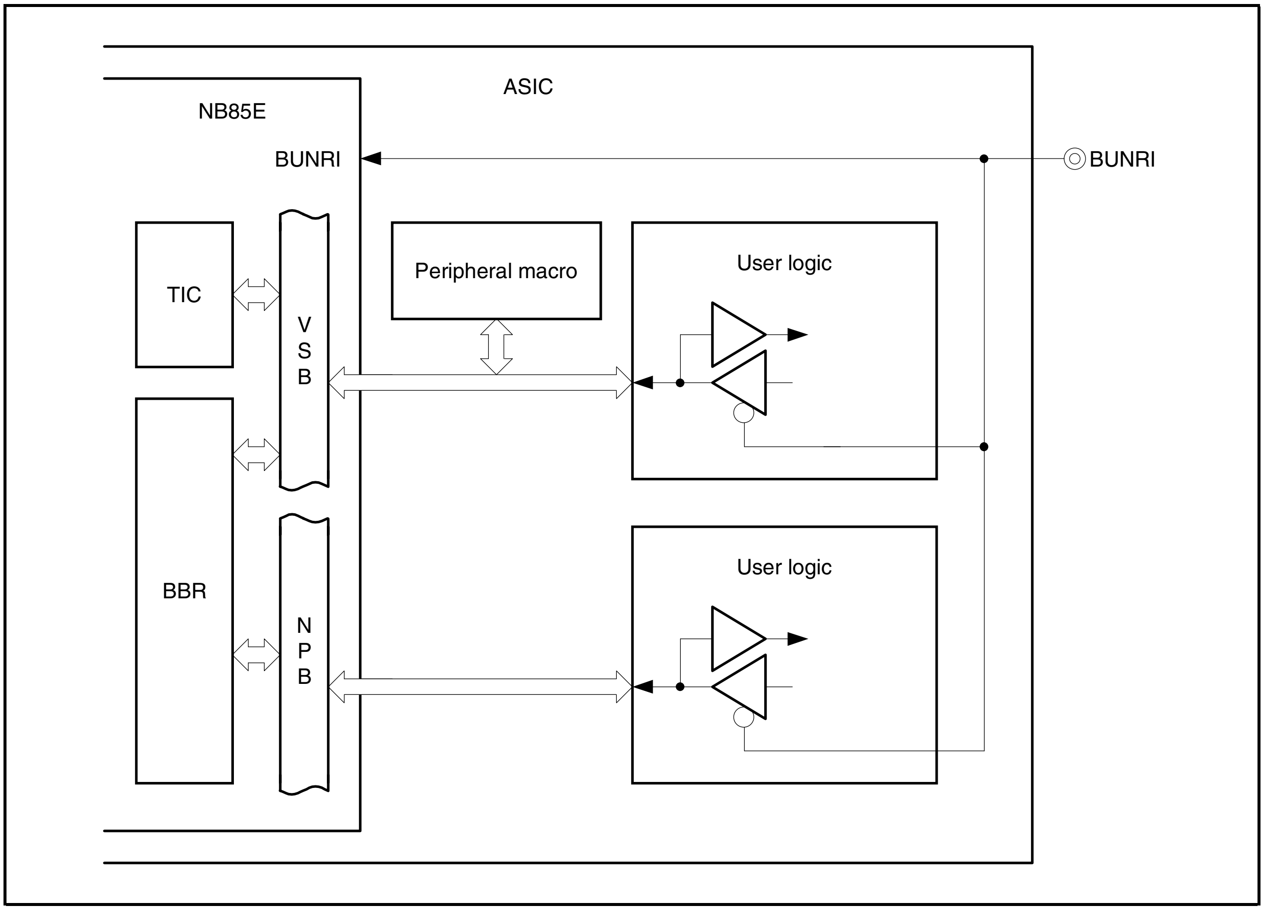
When testing a peripheral macro via test interface pins, if the peripheral macro and user logic are both connected to the VSB, a signal conflict may occur when in test mode. In order to prevent signal conflict, it is necessary to enable only the peripheral macro signal. Therefore, design the user logic so that the following I/O pins connected to user logic are high impedance in test mode (see **Figure 10-2**). If the design is already such that each signal does not conflict in test mode, this restriction dose not apply.

• VBWAIT	• VBLAST	• VBAHLD
• VBSEQ2 to VBSEQ0	• VBBSTR	• VDCSZ7 to VDCSZ0
• VDSELPZ	• VBTTYP1, VBTTYP0	• VBSIZE1, VBSIZE0
• VBBENZ3 to VBBENZ0	• VBWRITE	• VBLOCK
• VBCTYP2 to VBCTYP0	• VBA27 to VBA0	• VBSTZ
• VPD15 to VPD0	• VBD31 to VBD0	

Make I/O pins other than the above the same as in normal mode (leave open if unused).

Caution Although NEC supports test bus auto wiring tools, these are not supported in the NB85E. The user should perform test bus wiring.

Figure 10-2. User Logic Design Example



(b) Input pin processing

★ Input a low level to the VAREQ pin. No particular handling is required for pins other than VAREQ (handle them the same way as in normal mode).

(c) Output pin processing

★ No particular handling is required (handle them the same way as in normal mode).

(2) Pins for test mode

Perform pin processing for the pins for test mode as follows.

	Pin Name	I/O	Pin Processing		
			If MEMC is connected	If cache is connected	If MEMC and cache are not connected
★	PHTDOn	Input	Connect to PHTDOn pin of NB85E500/NU85E500.	–	Input low level.
★	PHTDINn	Output	Connect to PHTDINn pin of NB85E500/NU85E500.	–	Leave open.
★	VPRESZ	Output	Connect to VPRESZ pin of NB85E500/NU85E500 and NU85E502.	Connect to VPRESZ pin.	
★	VPTCLK	Output	Connect to VPTCLK pin of NB85E500/NU85E500 and NU85E502.	Connect to VPTCLK pin.	
★	TESEN	Output	–	–	
★	PHTEST	Output	Connect to PHTEST pin of NB85E500/NU85E500.	–	
	TMODEn, TBREDZ	Output	Leave open.		

Remark n = 1, 0

(3) Cautions when using NB85E901 (RCU)

★ If the NB85E901 (RCU) is connected to the NB85E, the following pins are used in the unit test mode. Output all these pins outside the chip as external pins.

- TBI39 to TBI0[Note 1]
- TBO34 to TBO0[Note 1]
- TEST[Note 1]
- BUNRI
- DCK[Note 2]
- DRSTZ[Note 2]
- DMS[Note 2]
- DDI[Note 2]
- DDO[Note 2]
- DBINT[Notes 1, 2]

Notes 1. Can also function as normal mode pin(s).

2. NB85E901 pin

(4) Cautions when using NB85ET

★ Thirteen N-wire type in-circuit emulator connection pins (DCK, DRSTZ, DMS, DDI, DDO, DBINT, EVTTRG, TRCCLK, TRCDATA3 to TRCDATA0, TRCEND) are also used in the unit test mode of the NB85ET in addition to test buses. Output all these pins outside the chip as external pins. Do not use these pins as alternate-function pins (however, the EVTTRG pin and the DBINT pin can be shared with non-test bus pins (other than TBI39 to TBI0, TBO34 to TBO0)).

10.2.2 Processing MEMC pins

(1) NB85E500

(a) Pins for connecting external memory

These operate the same in test mode as in normal mode.

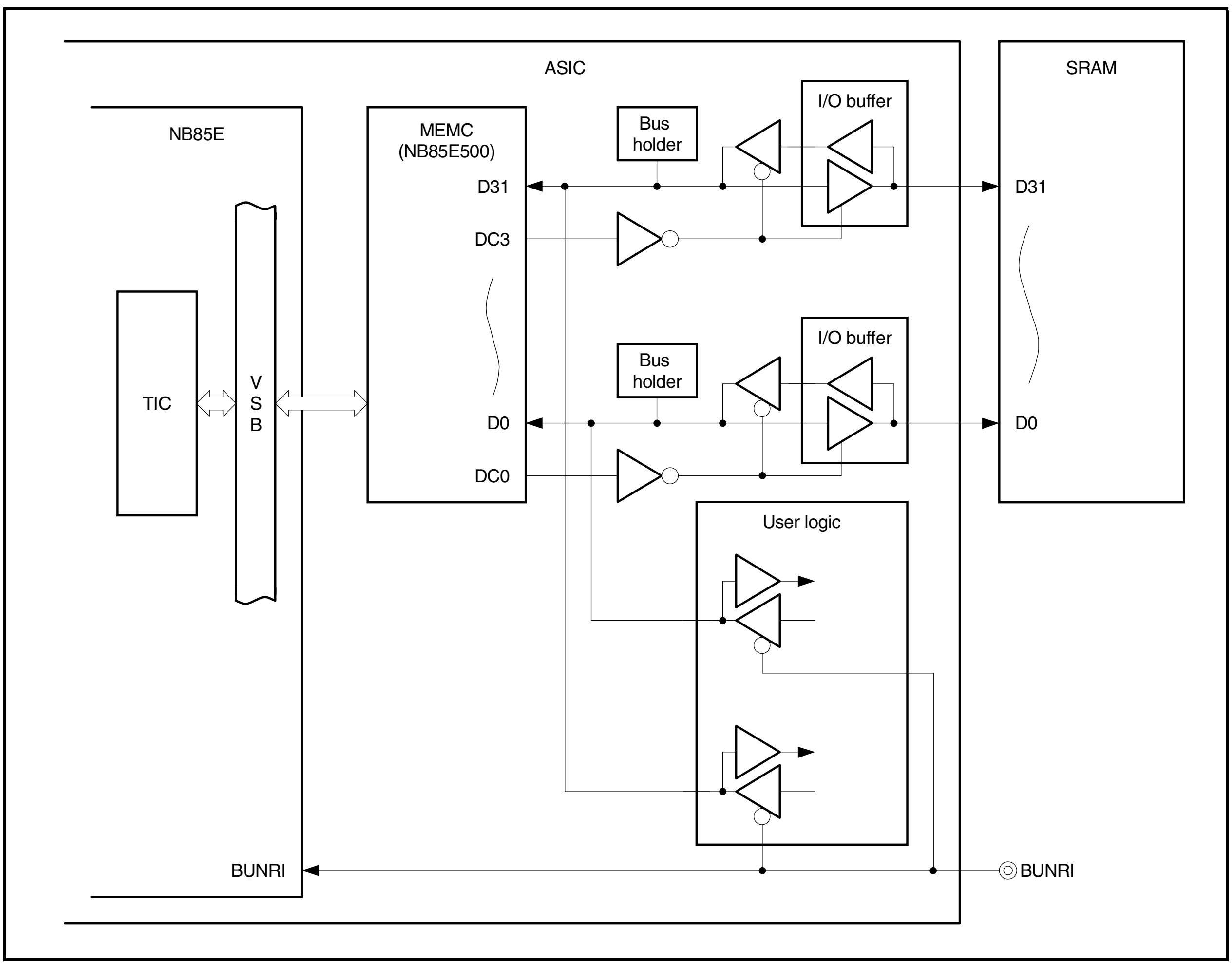
Therefore, if user logic, SRAM, or the like is connected to the data bus (D31 to D0), data bus signals may conflict when in test mode. In order to avoid this, design the user logic so that pins D31 to D0 are high impedance in test mode (see **Figure 10-3**).

Make pins other than D31 to D0 the same as in normal mode (if unused, process them according to the contents shown in the unused pin processing in the **Memory Controller NB85E, NB85ET User's Manual (A14206E)**).

Input signals at input pins (HLDRQZ, WAITZ, SELFREF) are ignored.

Caution Although NEC supports test bus auto wiring tools, these are not supported in the NB85E. The user should perform test bus wiring.

Figure 10-3. User Logic Design Example



(b) Pins for test mode

Connect to NB85E as shown in Figure 10-1.

(c) Other pins

Make the same as in normal mode (if unused, process them according to the contents shown in the unused pin processing in the **Memory Controller NB85E, NB85ET User's Manual (A14206E)**).

(2) NU85E502

(a) Pins for connecting external memory

These operate the same in test mode as in normal mode.
Input pins (D31 to D0) are ignored regardless of input values.

(b) Pins for test mode

Connect to NB85E as shown in Figure 10-1.

(c) Other pins

Make the same as in normal mode.

★ 10.3 Test Bus Auto Wiring

Since the NB85E and NB85E peripheral macros do not support test bus auto wiring tools, test bus wiring had to be performed by the user, but it is possible to apply an auto wiring tool by using the method described in **10.3.1 Test bus auto wiring method** (auto wiring using the auto wiring tool is possible only for the NB85E and NB85ET).

Connection prior to auto wiring or netlist editing following auto wiring is required for NB85E peripheral macros (memory controller, instruction/data cache, etc.) and user logic.

Also, test bus PINF file creation/editing is required regardless of whether test bus auto wiring is performed (for details, refer to **10.4 Test Bus PINF File Creation/Editing Methods**).

10.3.1 Test bus auto wiring method

(1) NB85E

Perform auto wiring using the normal method.

(2) NB85ET

Perform auto wiring using the normal method. However, specify the pins for N-wire type in-circuit emulator connection (DCK, DRSTZ, DMS, DDI, DDO, DBINT, EVTTRG, TRCCLK, TRCDATA3 to TRCDATA0, TRCEND) during auto wiring so that they are not shared with test pins (however, the DBINT and EVTTRG pins can be shared with pins other than test bus pins).

(3) Memory controller (NB85E500, NU85E500, NU85E502)

Memory controllers are not eligible for auto wiring because they do not have test mode pins.
Perform connection to the NB85E beforehand. The NB85E test mode pins are used for unit testing of the NB85E500, NU85E500, and NU85E502.

(4) Instruction cache (NB85E212, NB85E213), data cache (NB85E252, NB85E263)

Although they have a BUNRI pin, these caches are not eligible for auto wiring because they do not have TBIx and TBOx pins.
Since connecting the BUNRI pin beforehand makes auto wiring impossible, connect the TMODE1 pin of the NB85E to the BUNRI pin of the NB85E2xx (during auto wiring, the TMODE1 signal of the NB85E replaces the BUNRI signal). The NB85E2xx unit test is performed using the test mode pins of the NB85E.

(5) NB85E901

Although the NB85E901 has BUNRI and TEST pins, it is not eligible for auto wiring because it does not have the TBIx and TBOx pins.
Since connecting the BUNRI and TEST pins beforehand makes auto wiring impossible, clamp the BUNRI pin to low level, connect the TEST pin to the TMODE1 pin of the NB85E, and perform auto wiring of the test bus.
Convert the PWC format netlist output following execution to a Verilog netlist, and connect the signal input to the BUNRI pin of the NB85E to the BUNRI pin of the NB85E901. Then convert the Verilog netlist to the PWC format again.
The NB85E901 uses the test mode pins of the NB85E and the N-wire type in-circuit emulator connection pins of the NB85E901 (DCK, DRSTZ, DMS, DDI, DDO, DBINT) during unit tests. Be sure to output the N-wire type in-circuit emulator connection pins outside the chip as external pins. Also, specify them so that they are not shared with test pins during auto wiring (the DBINT pin can be shared with pins that are used in normal mode).

(6) User logic

In the user logic design examples shown in Figures 10-2 and 10-3, an external BUNRI pin is set beforehand, and the user logic is designed using that BUNRI signal. However, if the BUNRI pin is connected beforehand, auto wiring becomes impossible. Therefore, clamp the BUNRI input signal to the user logic to low level similarly to the NB85E901, edit the netlist following auto wiring, and then reconnect the BUNRI signal.

10.3.2 Verification of test bus by dummy model

Following test bus auto wiring, verification of the test bus connection can be done through simulation using a dummy model for the NB85E and NB85ET.

However, test bus connection verification via this dummy model consists only of verifying whether the NB85E and NB85ET test busses are normally connected.

Connection with the CPU core and NB85E peripheral macros, and connection on the normal pin side such as N-wire in-circuit emulator connection pins is not possible. Nor is it possible to verify whether unit testing of NB85E peripheral macros that use the test bus of the NB85E and NB85ET has been performed normally. Therefore, if the NB85E core and an NB85E peripheral macro are connected, or if the user circuit is connected to the VSB or memory controller, perform total chip simulation using a full-function model, as required (refer to **CHAPTER 11 TOTAL CHIP SIMULATION**).

★ 10.4 Test Bus PINF File Creation/Editing Methods

Since the memory controller, instruction/data cache, and NB85E901, which are peripheral macros of the NB85E, are not eligible for auto wiring, that macro information is not reflected in the PINF file created during auto wiring. Neither is the information of the N-wire type in-circuit emulator connection pins used during separation tests for the NB85E901 and NB85ET.

The method used to edit the PINF file created automatically is described below. Even if auto wiring is not performed, this PINF file is required to create the test patterns used to perform unit tests for mega macros.

A PINF file creation example is shown below.

10.4.1 PINF file creation example

(1) For NB85E + NU85E500 + NU85E502 + NB85E901 + NB85E212

BUNRI external pin name:	BUNRI
Test mode setting external pin names:	TMC1, TMC2
	TMC1 = TMC2 = 0 (NB85E and NB85E peripheral macro unit test)

Remarks 1. In the case of the unit test mode of a mega macro that does not use the NB85E test bus, it is not necessary to set N-wire type in-circuit emulator connection pins.

2. The N-wire type in-circuit emulator connection pins of the NB85E901 are used as external pins without changing their names.

3. *DECODER block: Describes the pin settings required for macro testing.

4. *TESTBUS block: Describes the test signal I/O pins.

Figure 10-4. PINF File Creation Example (NB85E + NU85E500 + NU85E502 + NB85E901 + NB85E212) (1/2)

MACRO NB85E (MACRO1)	Specification of pin during NB85E unit test
*DECODER	
TMC1 : 0 TMC2 : 0 BUNRI : 1	Specify the NB85E unit test mode
DCK : 1 DMS : 1 DDI : 1 DRSTZ : 0 DBINT : 0	Set the N-wire pin of NB85E901 to inactive level (However, reset pin is active) (Even if the DBINT pin is not set as a dedicated pin, it is required for macro separation testing, so set is as an alternate-function pin)
*TESTBUS	
xxxx : TBI0 : : xxxx : TBI39 xxxx : TBO0 : : xxxx : TBO34	Describe the test bus pins (xxxx: alternate-function external pin name)
*END	
MACRO NU85E500 (MACRO2)	Pin specification during NU85E500 unit test
*DECODER	
TMC1 : 0 TMC2 : 0 BUNRI : 1	Specify the NB85E unit test mode
DCK : 1 DMS : 1 DDI : 1 DRSTZ : 0 DBINT : 0	Set the N-wire pin of the NB85E901 to the inactive level (However, reset pin is active) (Even if the DBINT pin is not set as a dedicated pin, it is required for macro separation testing, so set it as an alternate-function pin)
*TESTBUS	
xxxx : TBI0 : : xxxx : TBI39 xxxx : TBO0 : : xxxx : TBO34	Describe the test bus pins (xxxx: alternate-function external pin name)
*END	
MACRO NU85E502Cn (MACRO3)	Pin specification during NU85E502 unit test (n = 7 to 0 (set the chip select area corresponding to NU85E500))
*DECODER	
TMC1 : 0 TMC2 : 0 BUNRI : 1	Specify the NB85E unit test mode
DCK : 1 DMS : 1 DDI : 1 DRSTZ : 0 DBINT : 0	Set the N-wire pin of the NB85E901 to the inactive level (However, reset pin is active) (Even if the DBINT pin is not set as a dedicated pin, it is required for macro separation testing, so set it as an alternate-function pin)
*TESTBUS	
xxxx : TBI0 : : xxxx : TBI39 xxxx : TBO0 : : xxxx : TBO34	Describe the test bus pins (xxxx: alternate-function external pin name)
*END	
MACRO NB85E212 (MACRO4)	Pin specification during NB85E212 unit test
*DECODER	
TMC1 : 0 TMC2 : 0 BUNRI : 1	Specify the NB85E unit test mode
DCK : 1 DMS : 1 DDI : 1 DRSTZ : 0 DBINT : 0	Set the N-wire pin of the NB85E901 to the inactive level (However, reset pin is active) (Even if the DBINT pin is not set as a dedicated pin, it is required for macro separation testing, so set it as an alternate-function pin)

Figure 10-4. PINF File Creation Example (NB85E + NU85E500 + NU85E502 + NB85E901 + NB85E212) (2/2)

```
          *TESTBUS
               xxxx : TBI0
                 :    :
               xxxx : TBI39
               xxxx : TBO0        }  Describe the test bus pin (xxxx: alternate-function external pin name)
                 :    :
               xxxx : TBO34
          *END
MACRO NB85E901 (MACRO5)           }  Pin specification during NB85E901 unit test mode
          *DECODER
               TMC1 : 0
               TMC2 : 0           }  Specify the NB85E unit test mode
               BUNRI : 1
          *TESTBUS
               xxxx : TBI0
                 :    :
               xxxx : TBI39
               xxxx : TBO0        }  Describe the test bus pins (xxxx: alternate-function external pin name)
                 :    :
               xxxx : TBO34
               DCK : DCK
               DMS : DMS
               DDI : DDI
               DRSTZ : DRSTZ      }  During NB85E901 unit test, enter all the N-wire pins (both input and output) to the TESTBUS block
               DBINT : DBINT
               DBO : DBO
          *END
MACRO Nxxxxxxx (MACRO6)           }  Pin specification during macro unit test other than Nx85Exxx
          *DECODER
               TMC1 : 0
               TMC2 : 1           }  Specify the Nxxxxxxx unit test mode
               BUNRI : 1
          *TESTBUS
               xxxx : TBI0
                 :    :
               xxxx : TBI20
               xxxx : TBO0        }  Describe the Nxxxxxxx test bus pins (xxxx: alternate-function external pin name)
                 :    :
               xxxx : TBO20
          *END
```

(2) For NB85ET + NU85E500 + NU85E502 + NB85E212

BUNRI external pin name: BUNRI
Test mode setting external pin names: TMC1, TMC2
TMC1 = TMC2 = 0 (NB85ET and NB85ET peripheral macro unit test)

Remarks 1. In the case of a separation test mode that does not use the NB85E test bus, it is not necessary to set N-wire type in-circuit emulator connection pins.

2. The N-wire type in-circuit emulator connection pins of the NB85E901 are used as external pins without changing their names.

3. *DECODER block: Describes the pin settings required for macro testing.

4. *TESTBUS block: Describes the test signal I/O pins.

Figure 10-5. PINF File Creation Example (NB85ET + NU85E500 + NU85E502 + NB85E212) (1/2)

MACRO NB85ET (MACRO1)	Pin specification during NB85ET unit test
∗DECODER	
TMC1 : 0	Specify the NB85ET unit test mode
TMC2 : 0	
BUNRI : 1	
∗TESTBUS	
xxxx : TBI0	Specify the test bus pins (xxxx: alternate-function external pin name)
: :	
xxxx : TBI39	
xxxx : TBO0	
: :	
xxxx : TBO34	
DCK : DCK	During NB85ET unit test, enter all the N-wire pins (both input and output) to the TESTBUS block (Even if the DBINT and EVTTRG pins are not set as dedicated pins, they are required during macro separation testing, so set them as alternate-function pins even if not setting them as dedicated pins)
DMS : DMS	
DDI : DDI	
DDO : DDO	
DRSTZ : DRSTZ	
DBINT : DBINT	
TRCCLK : TRCCLK	
TRCDATA0 : TRCDATA0	
TRCDATA1 : TRCDATA1	
TRCDATA2 : TRCDATA2	
TRCDATA3 : TRCDATA3	
TRCEND : TRCEND	
EVTTRG : EVTTRG	
∗END	
MACRO NU85E500 (MACRO2)	Pin specification during NU85E500 unit test
∗DECODER	
TMC1 : 0	Specify the NB85ET unit test mode
TMC2 : 0	
BUNRI : 1	
DCK : 1	Set N-wire pin of NB85ET to inactive level (However, reset pin is active) (Even if the DBINT pin is not set as a dedicated pin, it is required for macro separation testing, so set it as an alternate-function pin)
DMS : 1	
DDI : 1	
DRSTZ : 0	
DBINT : 0	
∗TESTBUS	
xxxx : TBI0	Describe test bus pins (xxxx: alternate-function external pin name)
: :	
xxxx : TBI39	
xxxx : TBO0	
: :	
xxxx : TBO34	
∗END	
MACRO NU85E502Cn (MACRO3)	Pin specification during NU85E502 unit test (n = 7 to 0 (set the chip select area corresponding to NU85E500))
∗DECODER	
TMC1 : 0	Specify the NB85ET unit test mode
TMC2 : 0	
BUNRI : 1	
DCK : 1	Set the N-wire pin of the NB85ET to the inactive level (However, reset pin is active) (Even if the DBINT pin is not set as a dedicated pin, it is required for macro separation testing, so set it as an alternate-function pin)
DMS : 1	
DDI : 1	
DRSTZ : 0	
DBINT : 0	
∗TESTBUS	
xxxx : TBI0	Describe the test bus pins (xxxx: alternate-function external pin name)
: :	
xxxx : TBI39	
xxxx : TBO0	
: :	
xxxx : TBO34	
∗END	

Figure 10-5. PINF File Creation Example (NB85ET + NU85E500 + NU85E502 + NB85E212) (2/2)

MACRO NB85E212 (MACRO4)	Pin specification during NB85E212 unit test
∗DECODER	
TMC1 : 0 TMC2 : 0 BUNRI : 1	Specify the NB85ET unit test mode
DCK : 1 DMS : 1 DDI : 1 DRSTZ : 0 DBINT : 0	Set N-wire pin of NB85ET to inactive level (However, reset pin is active) (Even if the DBINT pin is not set as a dedicated pin, it is required for macro separation testing, so set it as an alternate-function pin)
∗TESTBUS	
xxxx : TBI0 : : xxxx : TBI39 xxxx : TBO0 : : xxxx : TBO34	Describe the test bus pin names (xxxx: alternate-function external pin name)
∗END	
MACRO Nxxxxxxx (MACRO6)	Pin specification during macro unit test other than Nx85Exxx
∗DECODER	
TMC1 : 0 TMC2 : 1 BUNRI : 1	Specify the Nxxxxxxx unit test mode
∗TESTBUS	
xxxx : TBI0 : : xxxx : TBI20 xxxx : TBO0 : : xxxx : TBO20	Describe the Nxxxxxxx test bus pins (xxxx: alternate-function external pin name)
∗END	

CHAPTER 11 TOTAL CHIP SIMULATION

Total chip simulation is simulation performed for the entire CBIC by operating the NB85E.
The following are the two goals of total chip simulation.

(1) Creation of test patterns for checking connections

Creates test patterns (test patterns for selection by tester) that perform checks of connections between macros, checks of connections between user logic and macros, and checks of connections between external pins and macros or user logic.

★ It is difficult to perform all the connection checks between the NB85E and peripheral macros (memory controller, cache, NB85E901) and the memory controller (NB85E500/NU85E500 and NU85E502) on the user's side. Therefore, connection checks can be performed by executing Verilog simulation with a full function model using unit test patterns via a test bus for all the macros prepared by NEC.

Consult NEC on how to obtain unit test patterns.

(2) Verification of timing between macros

Checks the timing between macros, between a macro and user logic, or between a macro or user logic and external pins.

11.1 Creation of Test Patterns for Checking Connections

It is extremely difficult to input the necessary instruction pattern for the CPU read timing while referencing a program in order to create test patterns for checking connections. Therefore, ROM is placed virtually outside the CBIC (virtual ROM) and the program for testing[Note] is allocated there.

The CPU executes a simulation by reading the program for testing in the virtual ROM and extracts the results of dumping the pattern of all pins of the CBIC as the test pattern for checking connections.

This section describes the test pattern creation method for the following three cases.

- When ROM is not connected to VFB
- When ROM is connected to VFB and creating test pattern by connecting virtual ROM to VSB
- When ROM is connected to VFB and writing program for testing in that ROM

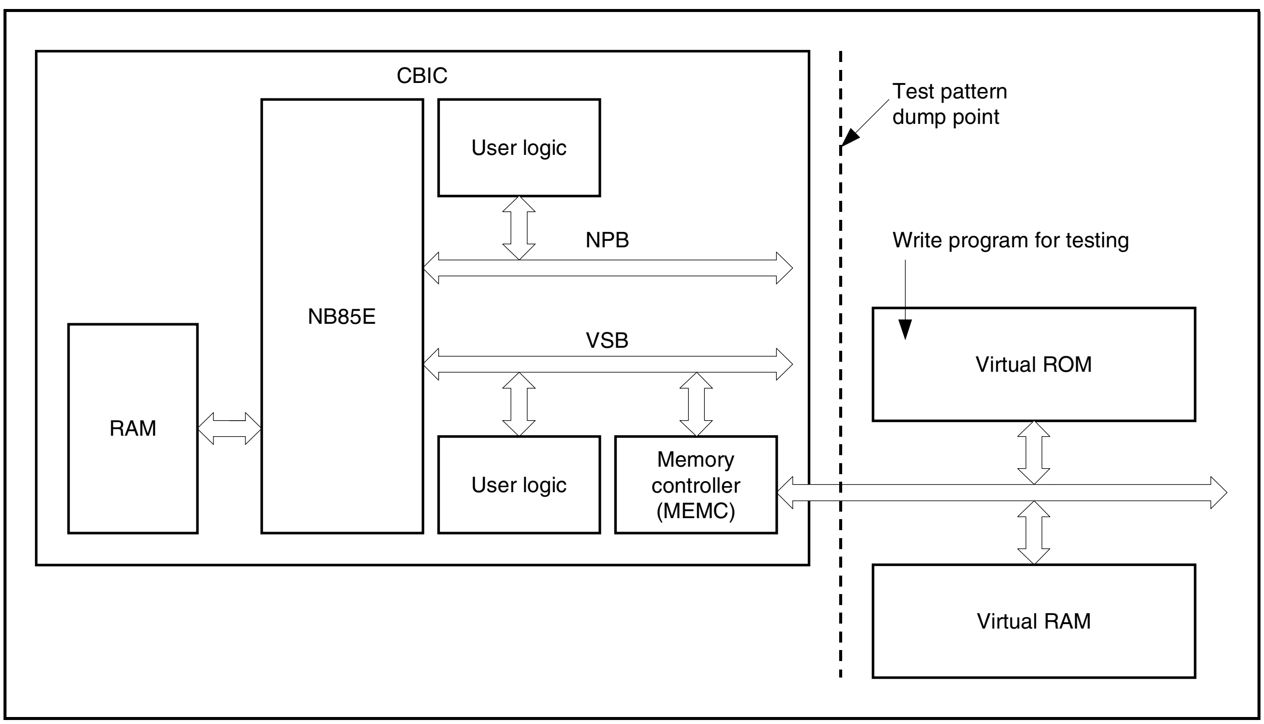
Note In this chapter, "program for testing" is defined as a program for performing checks of connections between macros, checks of connections between macros and user logic, and checks of connections between macro or user logic and external pins.

(1) When ROM is not connected to VFB

Connect a virtual ROM in which the program for testing is written outside the CBIC.

Dump all pins of the CBIC and extract them as the test pattern.

★ **Figure 11-1. Block Diagram for Total Chip Simulation (When There Is No ROM in VFB)**



(2) When ROM is connected to VFB

Creation of test patterns for checking connections when program ROM is connected to the VFB is described for when not writing a program for testing in VFB ROM and when writing one.

(a) When not writing program for testing in ROM of VFB
(Create test pattern by connecting virtual ROM)

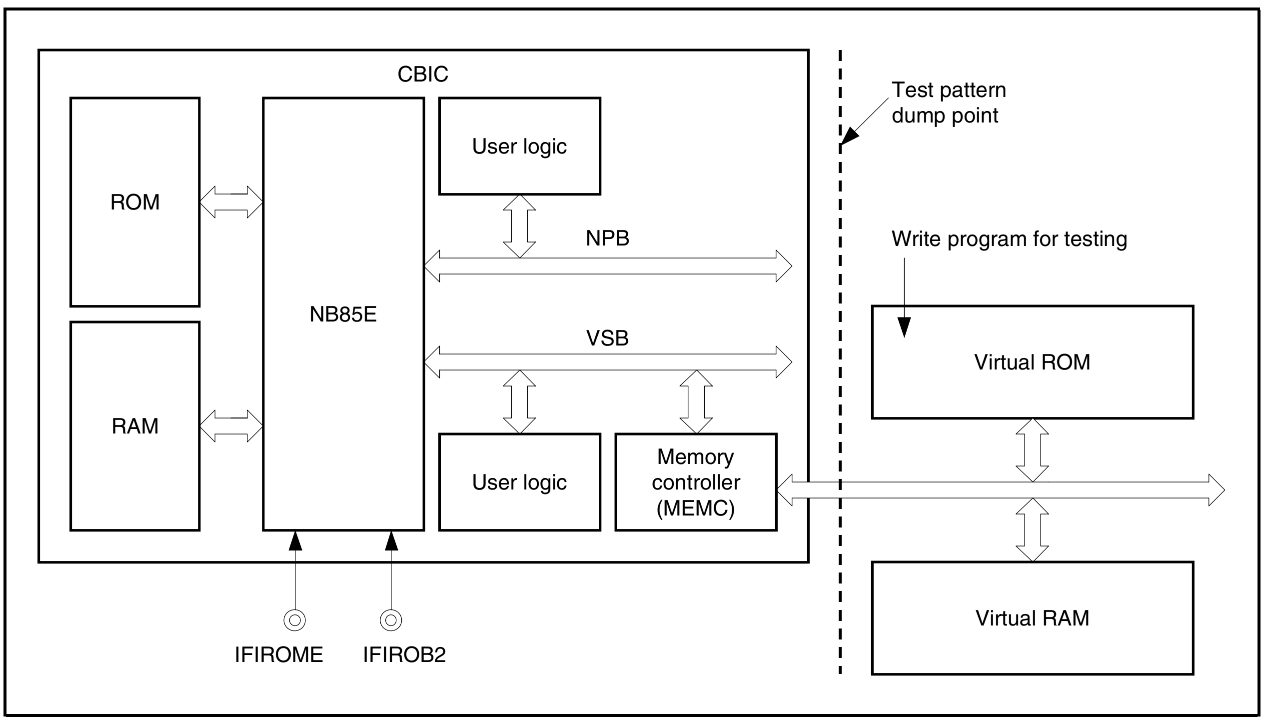
As shown in Figure 11-2, place virtual ROM outside the CBIC the same as when there is no ROM in the VFB, and dump all pins of the CBIC and extract them as the test pattern.

Perform test pattern creation by setting IFIROME = 0 and IFIROB2 = 1 and assigning the program for testing to the address 100000H.

Virtual RAM is not required.

Caution When load testing, the NB85E must be switched to ROMless mode. Therefore, establish pins IFIROME and IFIROB2 as the CBIC external pins.

★ **Figure 11-2. Block Diagram for Total Chip Simulation**
(When There Is ROM in VFB and Creating Test Pattern by Connecting Virtual ROM to VSB)



(b) When writing program for testing in ROM connected to VFB

Write the program for testing in the ROM of the VFB so that connections between macros or connections between macros and user logic can be verified and make settings so that this program for testing is executed only when testing.

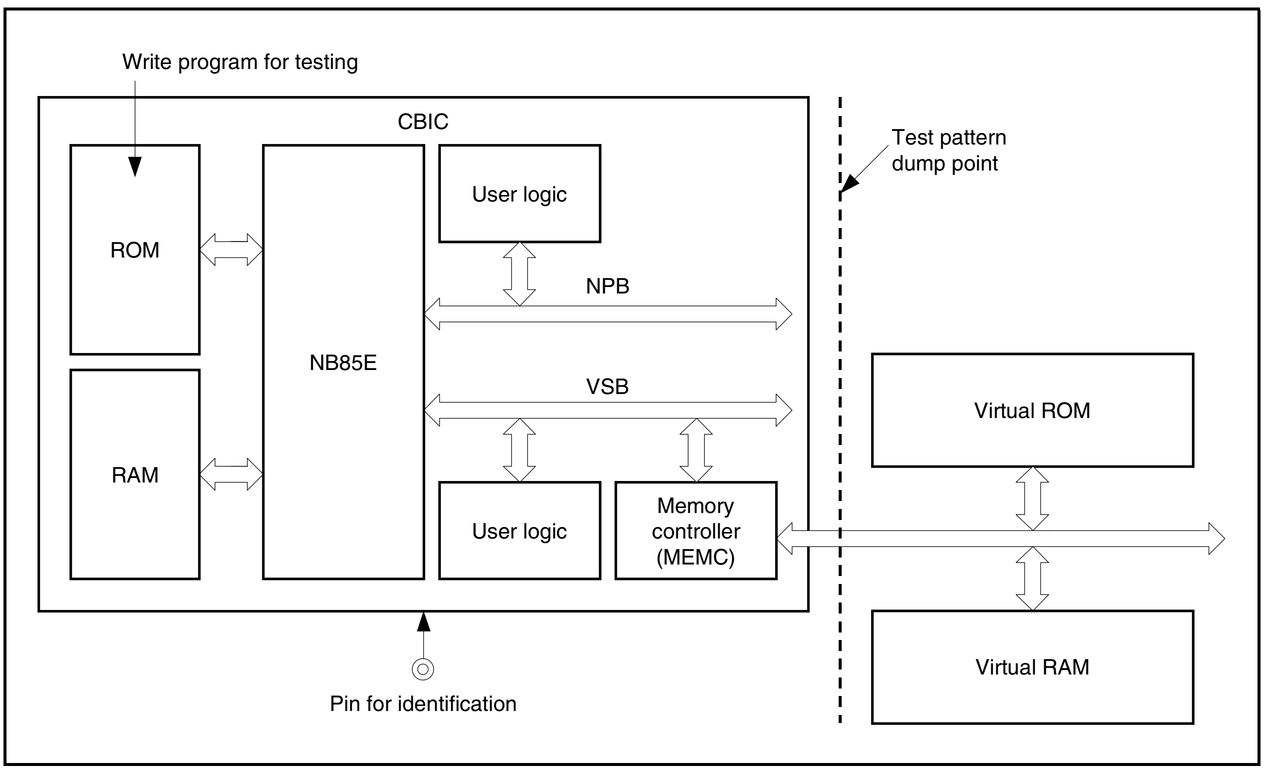
In order to write the program for testing in the ROM of the VFB, actual operating programs[Note] must be decreased by the amount of the test program.

The main advantages in this case are the following.

- When not assigning the external bus to CBIC pins, decrease the number of pins for testing (when reading virtual ROM, the address bus and data bus must be output externally)
- Decrease in the number of test patterns according to difference in number of clocks per access and bus width

Note In this chapter, "actual operating program" is defined as a program for making the target system operate.

★ **Figure 11-3. Block Diagram for Total Chip Simulation (When There Is ROM in VFB and Writing Program for Testing in That ROM)**



As a method of identifying execution of the program for testing and execution of the actual operating program, for example, set the pin for identification and ensure that the pin for identification is checked after NB85E initialization is performed. Depending on the level of the pin for identification, jump to the program for testing or the actual operating program and execute the program.

Virtual ROM and virtual RAM are not needed in this case.

11.2 Verifying Timing Between Macros

Besides checking functions such as connections between macros, simulation performs real-time operation verification (real-time simulation). It verifies that there are no timing problems between the NB85E and other macros, the NB85E and user logic, other macros and user logic, or other circuits.

In order to operate the NB85E, a test program for timing verification is needed. The test program for timing verification is allocated in the ROM if there is ROM in the CBIC and in virtual ROM if there is no ROM, and simulation is performed (the test program for timing verification is used only in simulation; it need not actually be placed in ROM in the CBIC).

Real-time simulation predicts when an inconsistency occurs due to a delay difference in MIN. and MAX. simulation. In this case, confirm that there is no problem in inconsistent portions by checking that there are no timing or functional errors.

CHAPTER 12 ROM CODE CREATION

A compiled type ROM (compiled ROM) is used as the ROM connected to the NB85E.
Be aware of the following points when using compiled ROM.

(1) ROM code format at sign-off

NEC can receive orders in NINCF format.

For details about NINCF format, refer to the **CB-9 Family VX/VM Type Memory Macro (Compiled Type) Design Manual (A12982E)**.

(2) Assigning ROM code to multiple compiled ROMs

Consult NEC when assigning ROM code to multiple compiled ROMs.

(3) Simulation with ROM code

For details, refer to **NEC SYSTEM LSI DESIGN OPENCAD V5.4 Verilog-XL Interface User's Manual (A15052E).**

★

APPENDIX REVISION HISTORY

Revisions up to the previous edition are shown below. The “Pages” column indicates pages in the earlier edition to which the revision was applied.

(1) 1st edition → 2nd edition

Pages	Description
Throughout	• Change of name of NPB peripheral macros from “NANPxxx” to “QLNPBxxx” • Deletion of description regarding NB85E501 of MEMC • Change of name of MEMC from “NB85E502” to “NU85E502”
pp.19 to 22	Modification of **CHAPTER 2 CONNECTION OF CLOCK CONTROL CIRCUIT**
p.29	Modification of **Figure 3-6 Connection Example of Compiled ROM to VFB**
p.48	Modification of **Figure 5-1 Connection Example of NB85E, MEMC, and External Memory (SRAM, SDRAM)**
pp.57, 58	Deletion of description regarding a data cache connection in **5.3 Connection to SDRAM**
p.72	Modification of **Figure 6-4 Connection Example of Address Decoder**
p.75	Modification of **Figure 6-7 Connection Example of User Logic**
p.76	Modification of **Figure 6-9 HDL Creation Example of User Logic**
p.78	Modification of **Figure 6-11 HDL Creation Example of User Logic with Retry Function**
p.92	Modification of **Remark 2** in **Figure 8-4 Recommended Circuit Example for IE Connection (NB85E + RCU (NB85E901))**
p.93	Modification of **Figure 8-5 Recommended Circuit Example for IE Connection (NB85ET)** Modification of **Remarks 2** and **3**
p.103	Modification of **9.5 Device File**
p.109	Modification of **10.2.1 (3) Precautions when using NB85E901 (RCU)** and **(4) Precautions when using NB85ET**

Facsimile Message

From:

Name

Company

Tel. FAX

Address

Although NEC has taken all possible steps to ensure that the documentation supplied to our customers is complete, bug free and up-to-date, we readily accept that errors may occur. Despite all the care and precautions we've taken, you may encounter problems in the documentation. Please complete this form whenever you'd like to report errors or suggest improvements to us.

Thank you for your kind support.

North America
NEC Electronics Inc.
Corporate Communications Dept.
Fax: +1-800-729-9288
+1-408-588-6130

Europe
NEC Electronics (Europe) GmbH
Market Communication Dept.
Fax: +49-211-6503-274

South America
NEC do Brasil S.A.
Fax: +55-11-6462-6829

Hong Kong, Philippines, Oceania
NEC Electronics Hong Kong Ltd.
Fax: +852-2886-9022/9044

Korea
NEC Electronics Hong Kong Ltd.
Seoul Branch
Fax: +82-2-528-4411

Taiwan
NEC Electronics Taiwan Ltd.
Fax: +886-2-2719-5951

Asian Nations except Philippines
NEC Electronics Singapore Pte. Ltd.
Fax: +65-250-3583

Japan
NEC Semiconductor Technical Hotline
Fax: +81- 44-435-9608

I would like to report the following error/make the following suggestion:

Document title: ____________________

Document number: ____________________ Page number: __________

If possible, please fax the referenced page or drawing.

Document Rating	Excellent	Good	Acceptable	Poor
Clarity	❏	❏	❏	❏
Technical Accuracy	❏	❏	❏	❏
Organization	❏	❏	❏	❏

CS 01.11