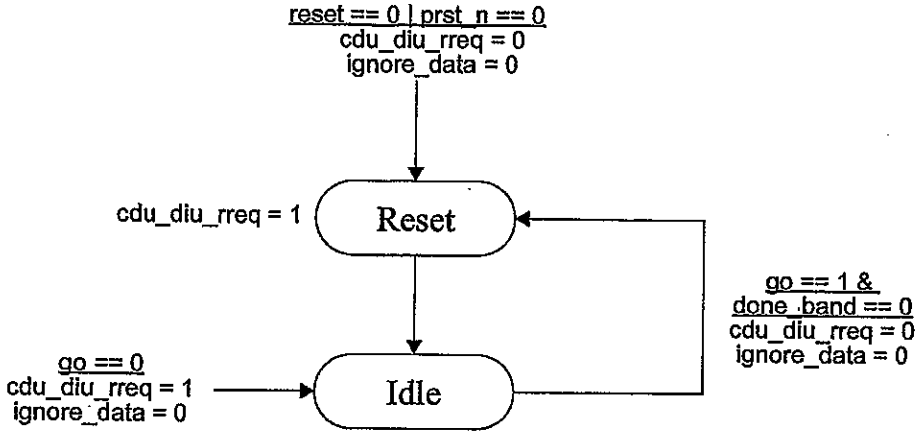

reset == 0 | prst_n == 0
cdu_diu_rreq = 0
ignore_data = 0
cdu_diu_rreq = 1
Reset
go == 1 &
done_band == 0
cdu_diu_rreq = 0
ignore_data = 0
go == 0
cdu_diu_rreq = 1
ignore_data = 0
Idle

FIG. 1

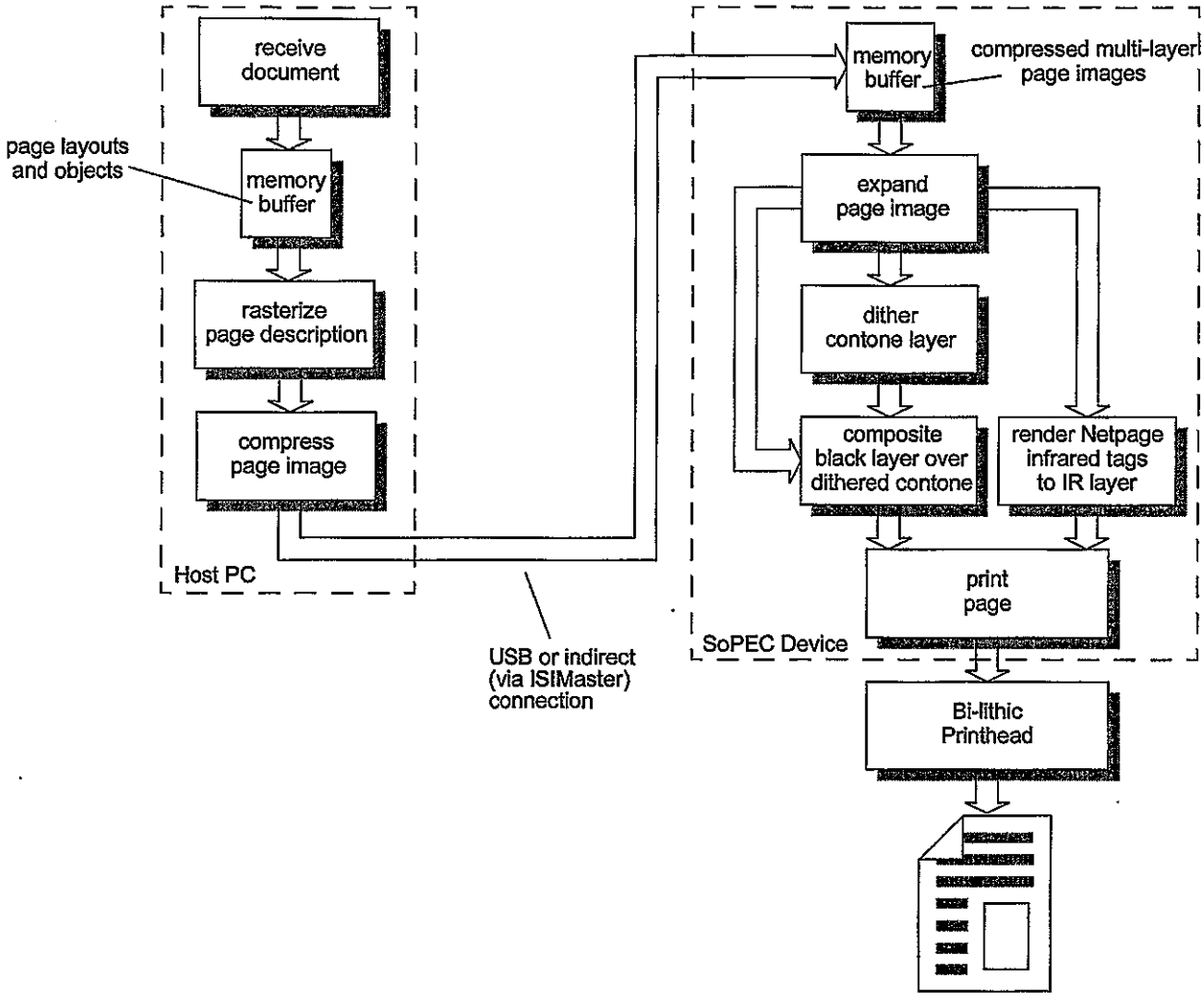

receive document
page layouts and objects
memory buffer
rasterize page description
compress page image
Host PC
USB or indirect (via ISIMaster) connection
memory buffer
compressed multi-layer page images
expand page image
dither contone layer
composite black layer over dithered contone
render Netpage infrared tags to IR layer
print page
SoPEC Device
Bi-lithic Printhead

FIG. 2

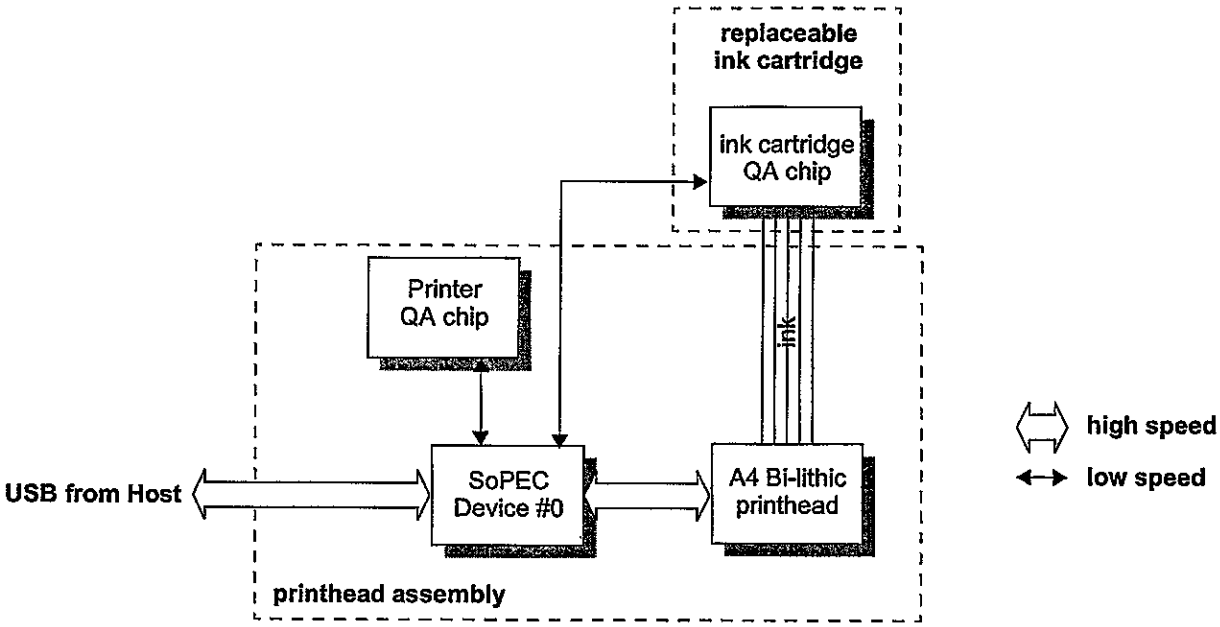

replaceable
ink cartridge
ink cartridge
QA chip
Printer
QA chip
ink
USB from Host
SoPEC
Device #0
A4 Bi-lithic
printhead
printhead assembly
high speed
low speed

FIG. 3

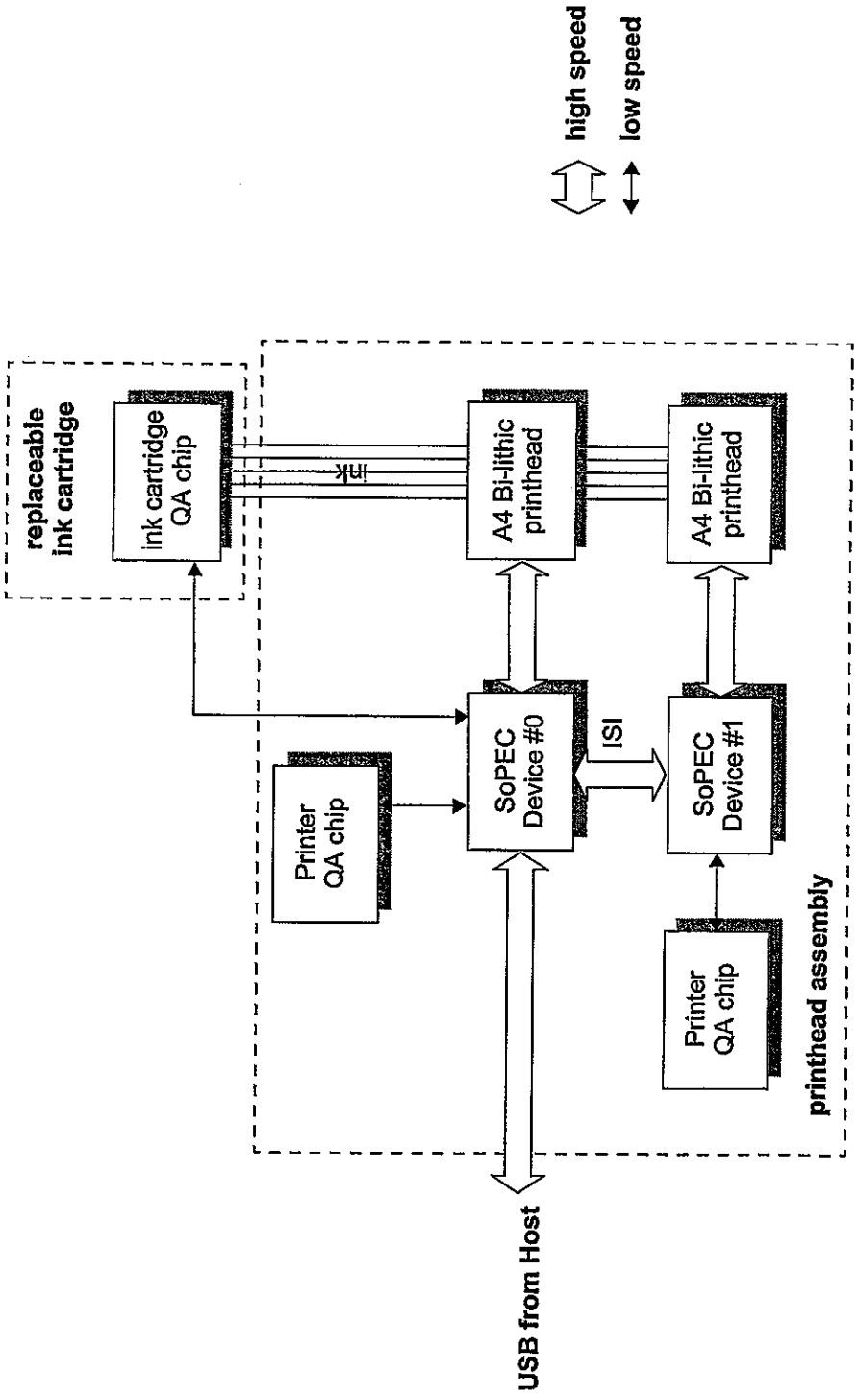

high speed
low speed
replaceable ink cartridge
ink cartridge QA chip
A4 Bi-lithic printhead
A4 Bi-lithic printhead
SoPEC Device #0
ISI
SoPEC Device #1
Printer QA chip
Printer QA chip
printhead assembly
USB from Host

FIG. 4

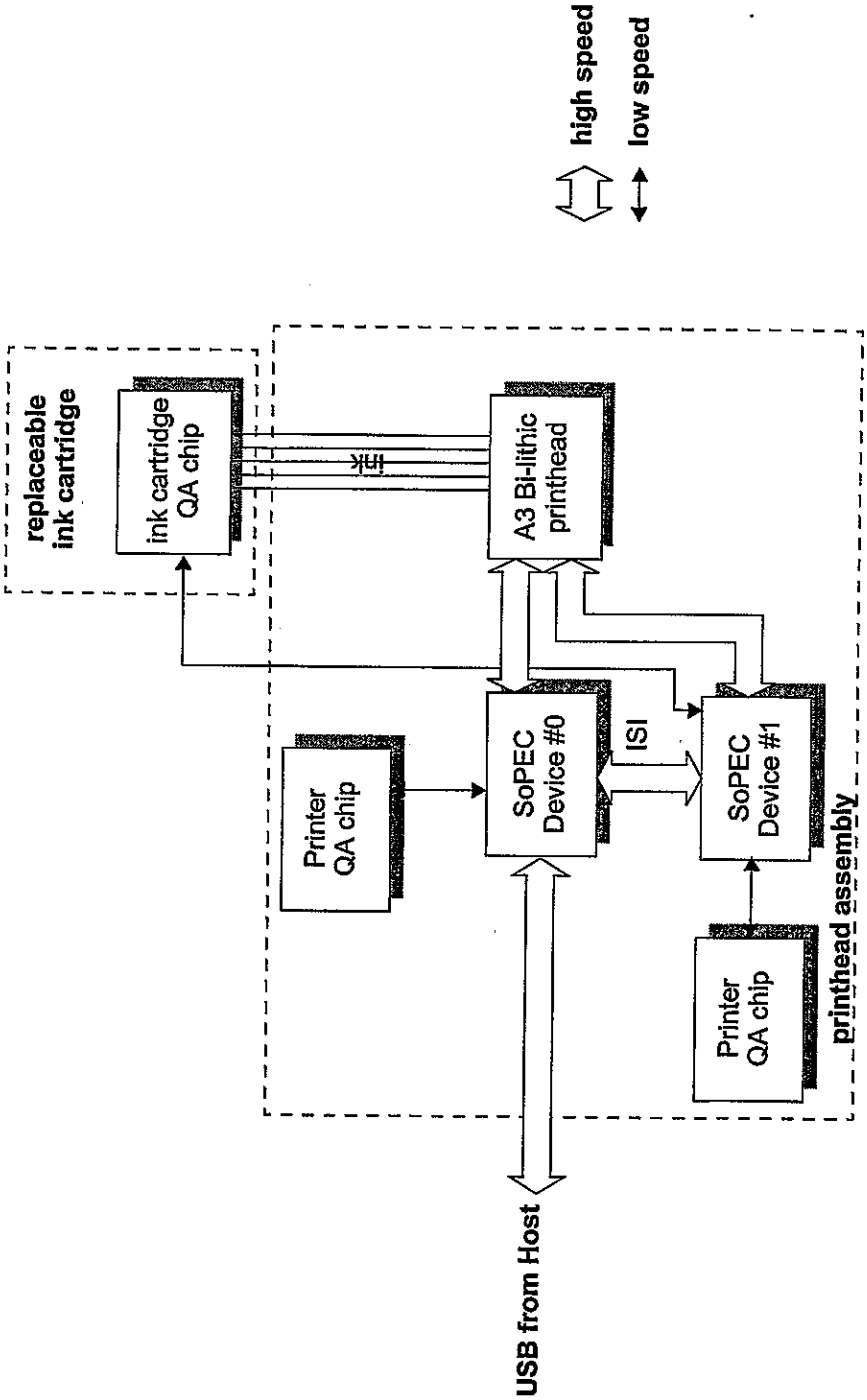

high speed
low speed
replaceable ink cartridge
ink cartridge QA chip
A3 Bi-lithic printhead
Printer QA chip
SoPEC Device #0
ISI
SoPEC Device #1
Printer QA chip
printhead assembly
USB from Host

FIG. 5

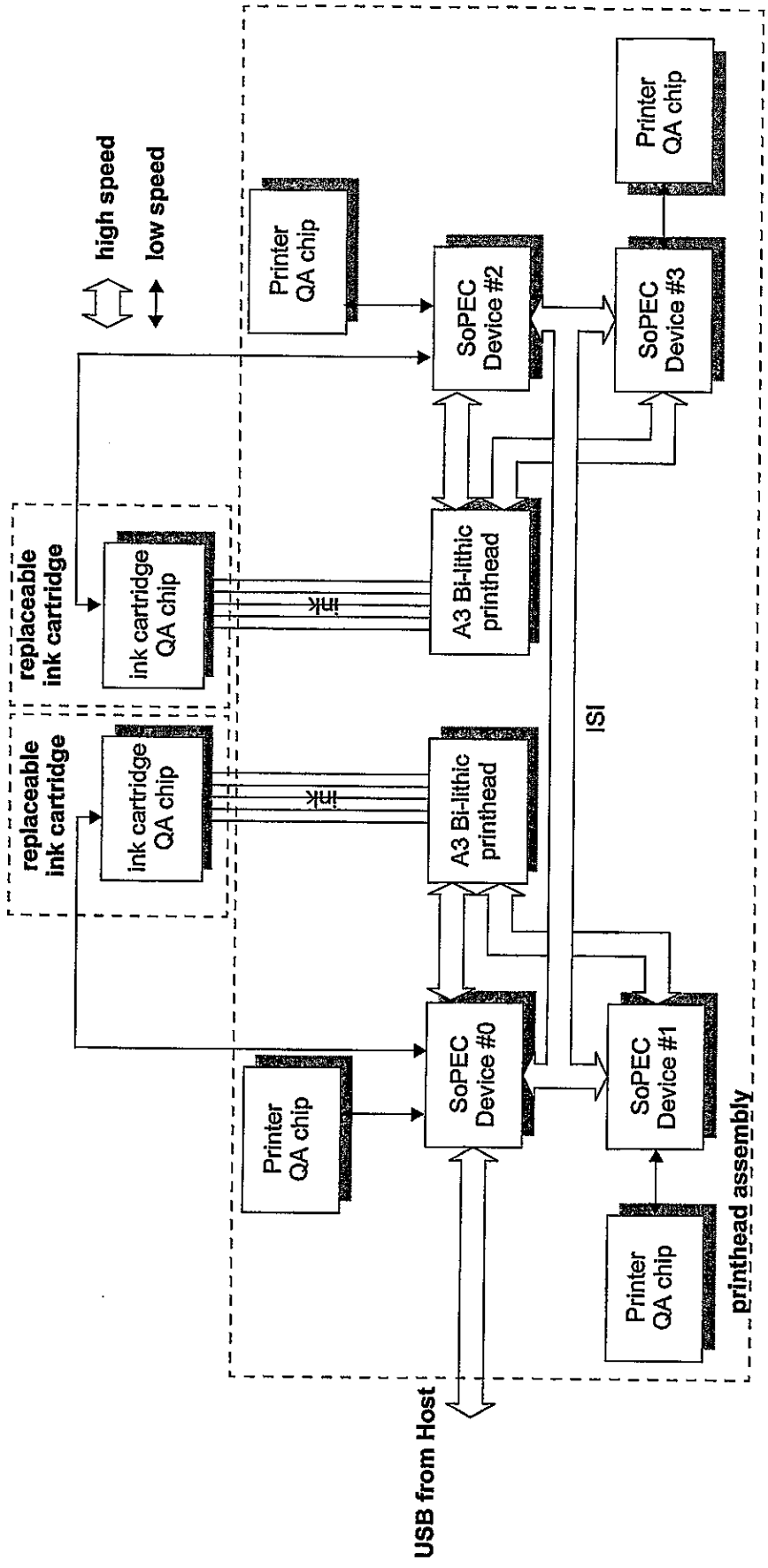

high speed
low speed
Printer QA chip
SoPEC Device #2
Printer QA chip
SoPEC Device #3
replaceable ink cartridge
ink cartridge QA chip
A3 Bi-lithic printhead
ink
replaceable ink cartridge
ink cartridge QA chip
A3 Bi-lithic printhead
ink
ISI
Printer QA chip
SoPEC Device #0
SoPEC Device #1
Printer QA chip
printhead assembly
USB from Host

FIG. 6

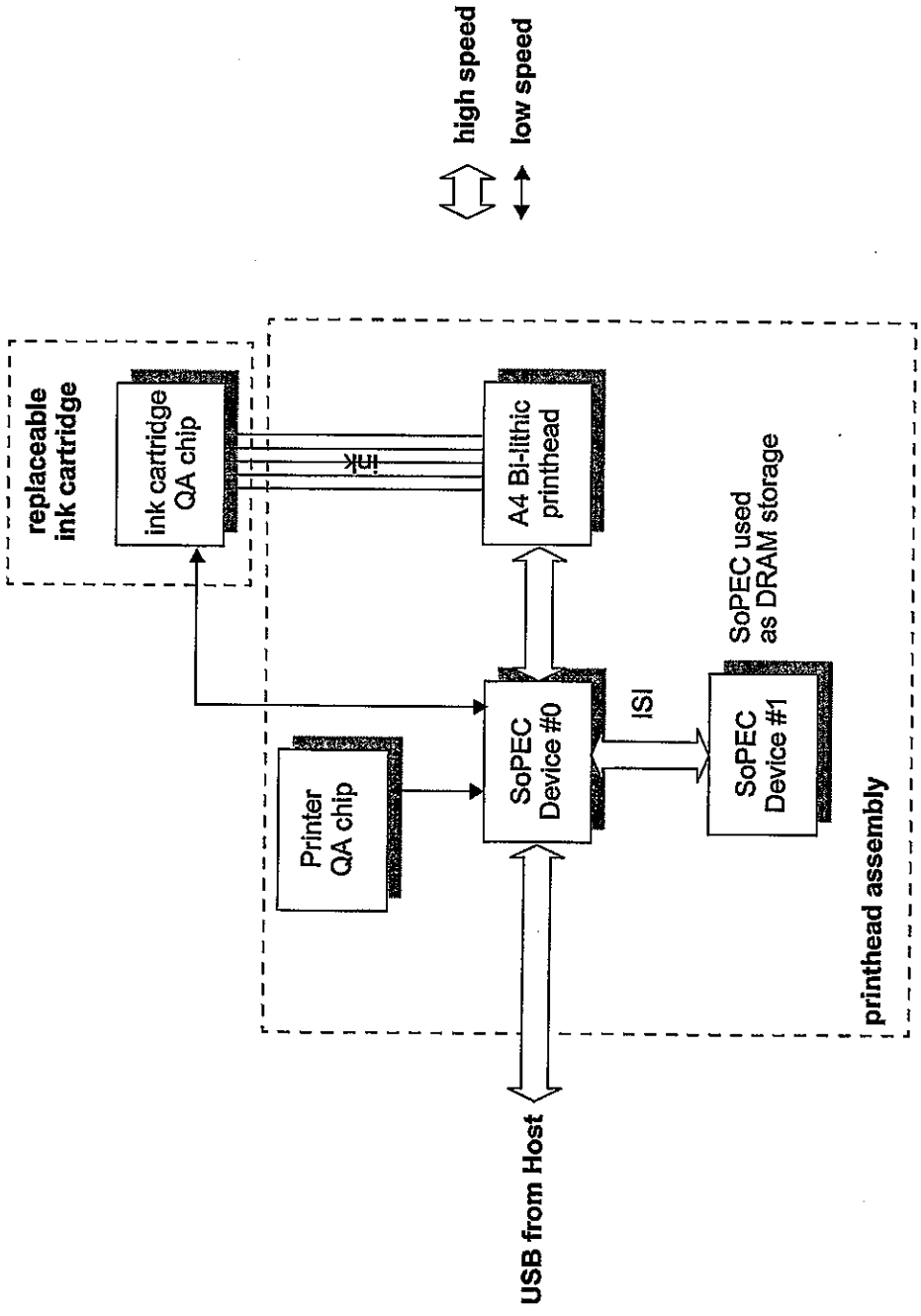

high speed
low speed
replaceable ink cartridge
ink cartridge QA chip
ink
A4 Bi-lithic printhead
SoPEC used as DRAM storage
Printer QA chip
SoPEC Device #0
ISI
SoPEC Device #1
printhead assembly
USB from Host

FIG. 7

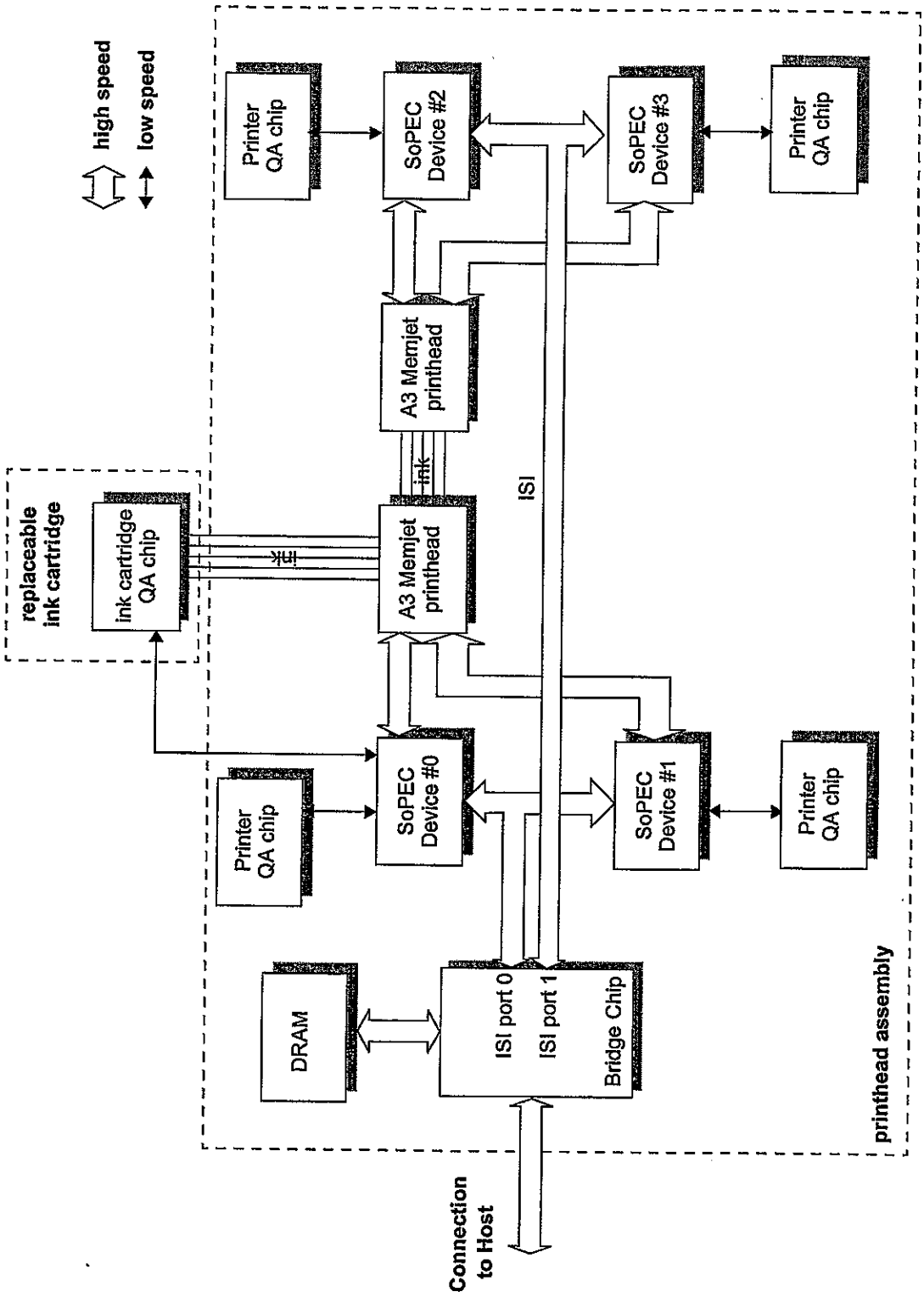

high speed
low speed
Printer QA chip
SoPEC Device #2
SoPEC Device #3
Printer QA chip
A3 Memjet printhead
ink
ISI
replaceable ink cartridge
ink cartridge QA chip
A3 Memjet printhead
Printer QA chip
SoPEC Device #0
SoPEC Device #1
Printer QA chip
DRAM
ISI port 0
ISI port 1
Bridge Chip
printhead assembly
Connection to Host

FIG. 8

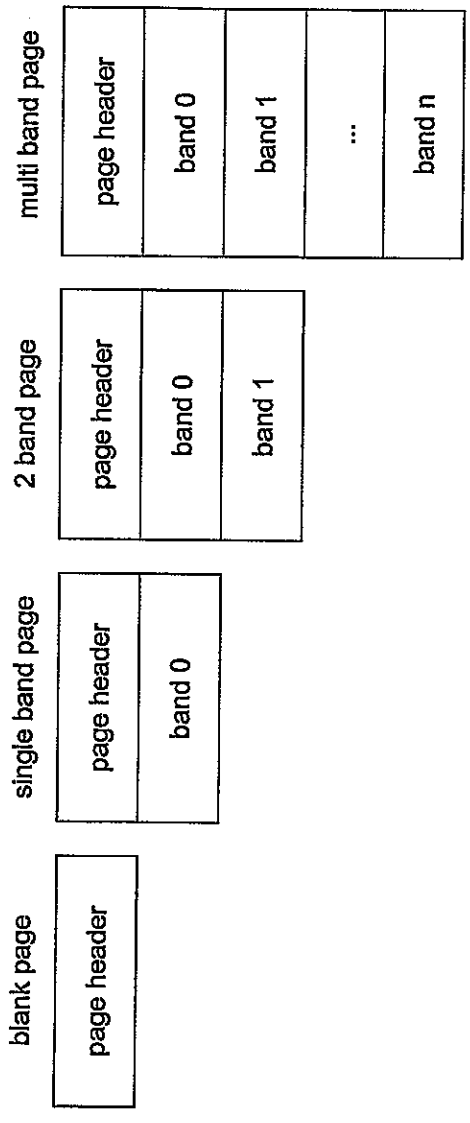

blank page
page header
single band page
page header
band 0
2 band page
page header
band 0
band 1
multi band page
page header
band 0
band 1
...
band n

FIG. 9

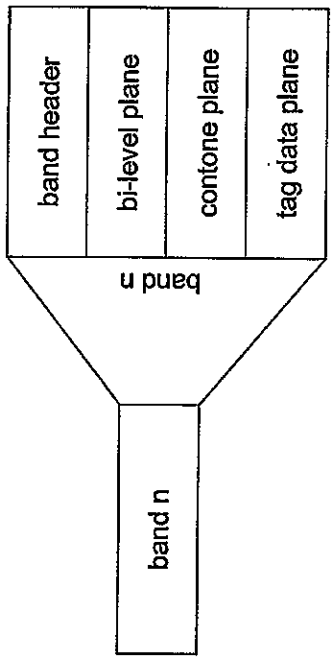

band header
bi-level plane
contone plane
tag data plane
band n
band n

FIG. 10

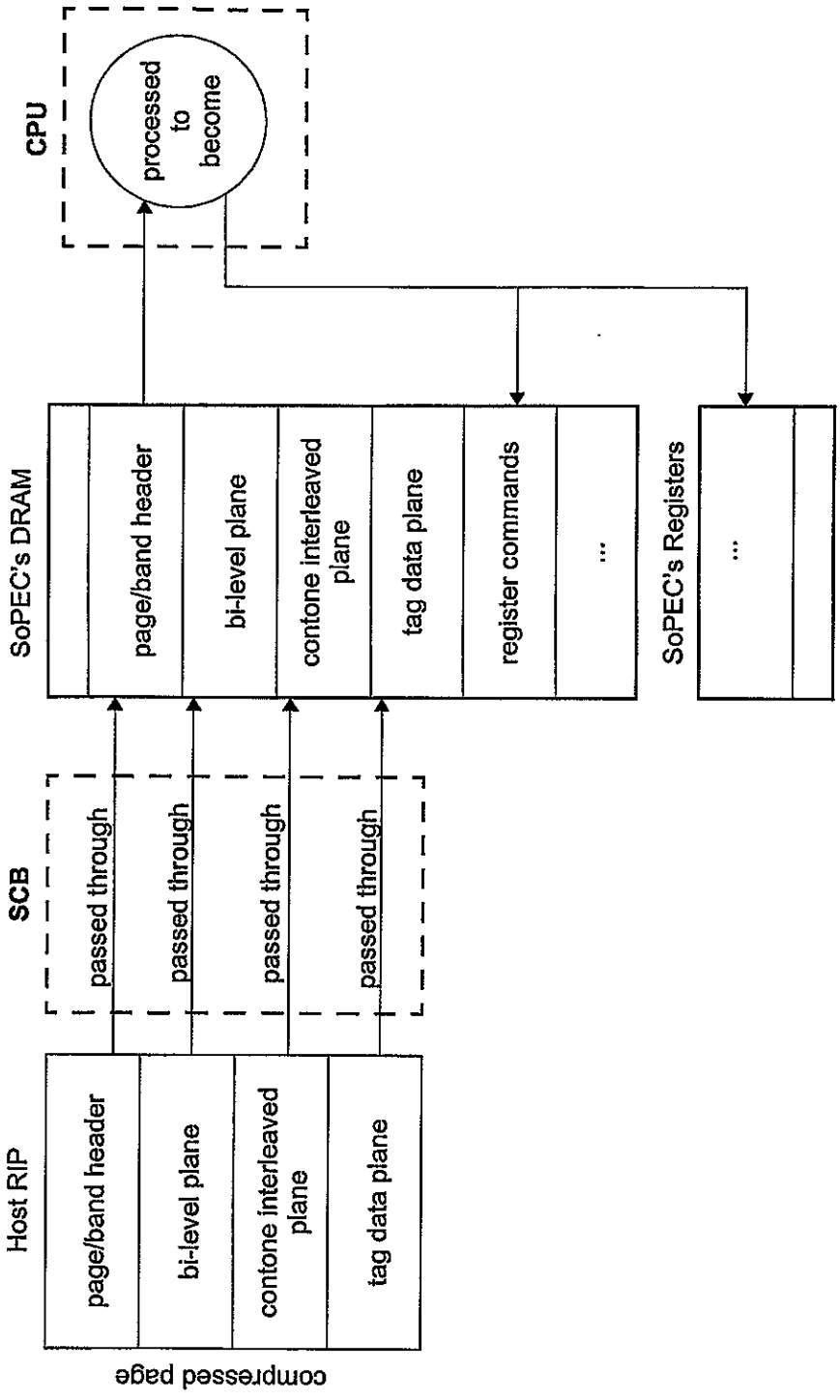

CPU
processed to become
SoPEC's DRAM
page/band header
bi-level plane
contone interleaved plane
tag data plane
register commands
...
SoPEC's Registers
...
SCB
passed through
passed through
passed through
passed through
Host RIP
page/band header
bi-level plane
contone interleaved plane
tag data plane
compressed page

FIG. 11

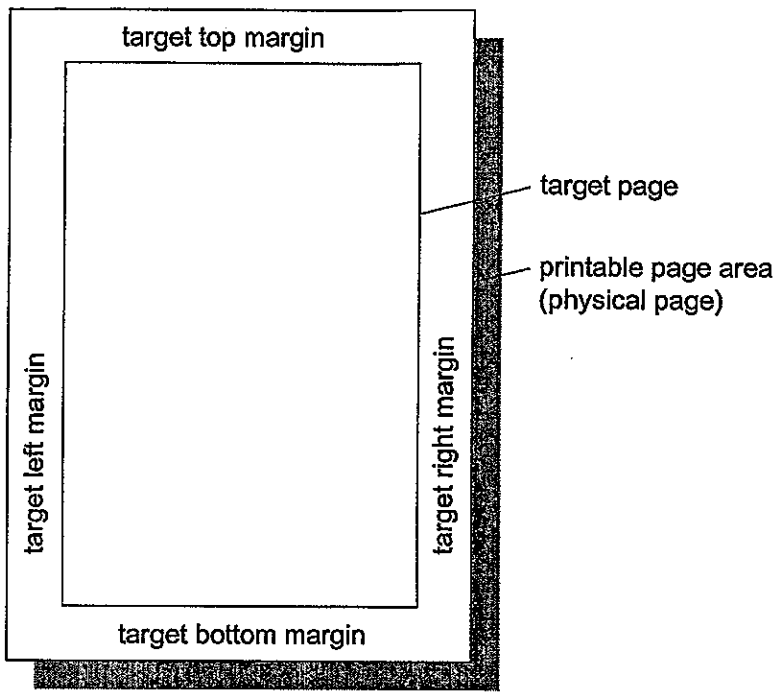

target top margin
target page
printable page area
(physical page)
target left margin
target right margin
target bottom margin

FIG. 12

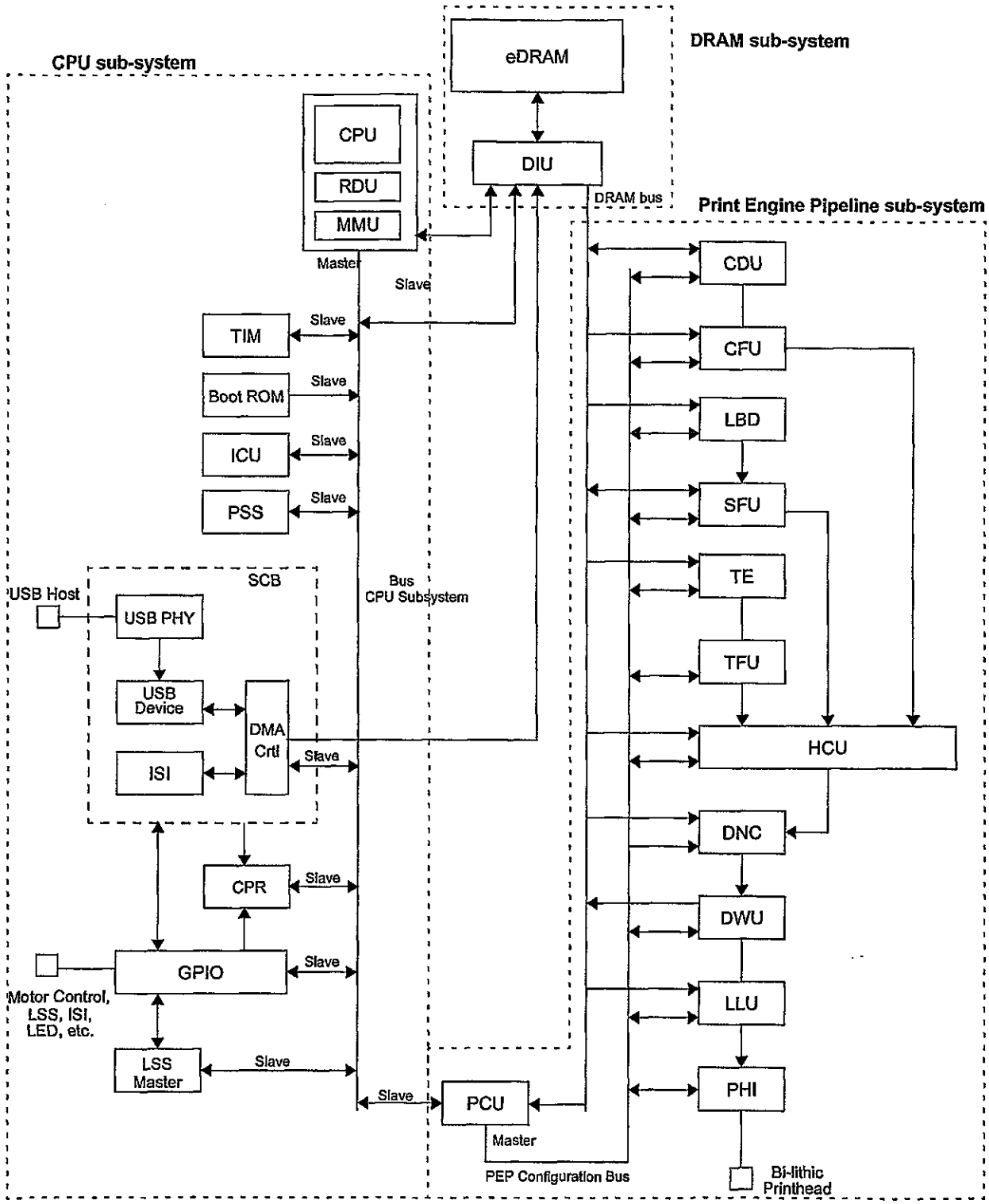

DRAM sub-system
eDRAM
DIU
DRAM bus
CPU sub-system
CPU
RDU
MMU
Master
Slave
Print Engine Pipeline sub-system
CDU
CFU
LBD
SFU
TE
TFU
HCU
DNC
DWU
LLU
PHI
TIM
Boot ROM
ICU
PSS
SCB
USB Host
USB PHY
USB Device
DMA Ctrl
ISI
Bus
CPU Subsystem
CPR
GPIO
Motor Control,
LSS, ISI,
LED, etc.
LSS Master
PCU
PEP Configuration Bus
Bi-lithic Printhead

FIG. 13

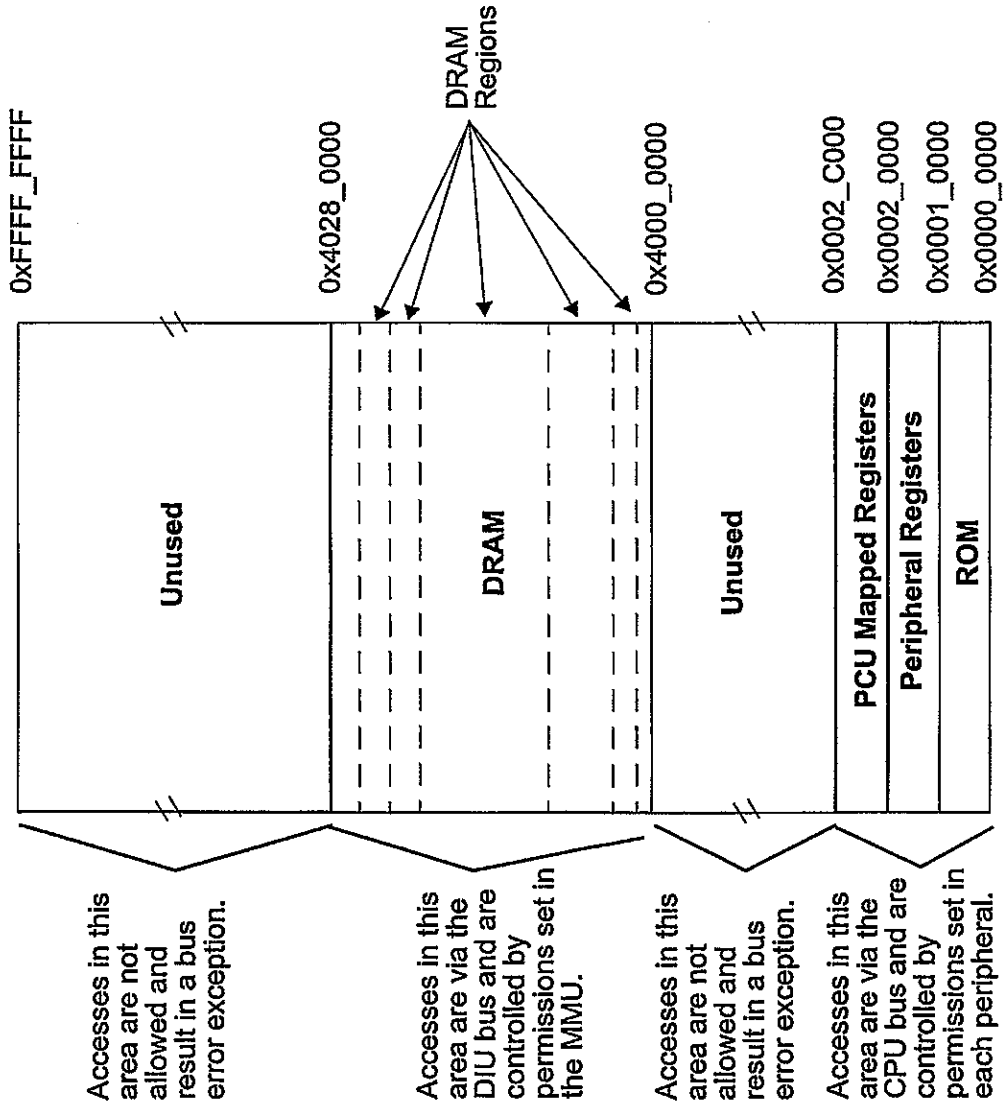

0xFFFF_FFFF
0x4028_0000
DRAM Regions
0x4000_0000
0x0002_C000
0x0002_0000
0x0001_0000
0x0000_0000
Unused
DRAM
Unused
PCU Mapped Registers
Peripheral Registers
ROM
Accesses in this area are not allowed and result in a bus error exception.
Accesses in this area are via the DIU bus and are controlled by permissions set in the MMU.
Accesses in this area are not allowed and result in a bus error exception.
Accesses in this area are via the CPU bus and are controlled by permissions set in each peripheral.

FIG. 14

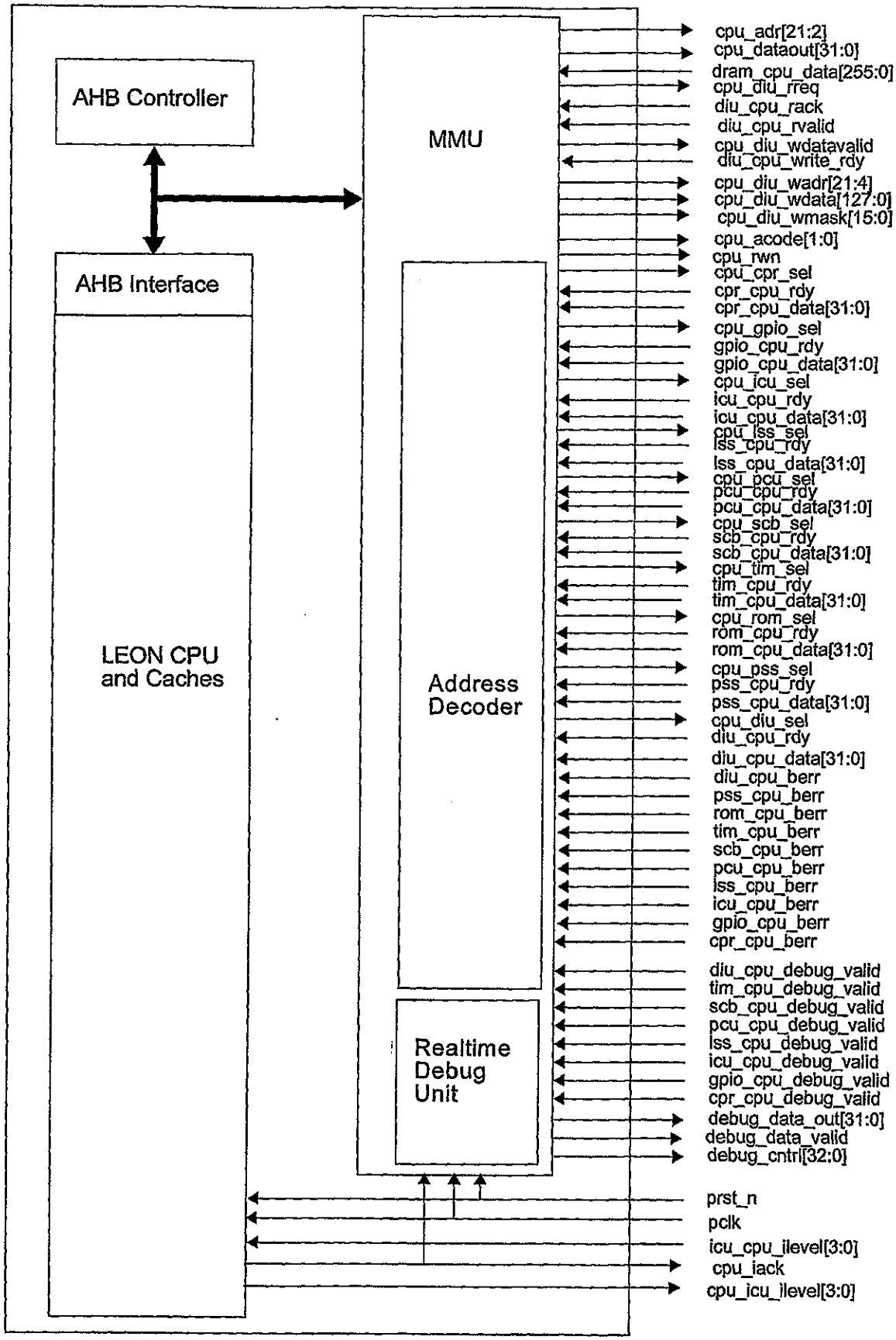

AHB Controller
AHB Interface
LEON CPU and Caches
MMU
Address Decoder
Realtime Debug Unit
cpu_adr[21:2]
cpu_dataout[31:0]
dram_cpu_data[255:0]
cpu_diu_rreq
diu_cpu_rack
diu_cpu_rvalid
cpu_diu_wdatavalid
diu_cpu_write_rdy
cpu_diu_wadr[21:4]
cpu_diu_wdata[127:0]
cpu_diu_wmask[15:0]
cpu_acode[1:0]
cpu_rwn
cpu_cpr_sel
cpr_cpu_rdy
cpr_cpu_data[31:0]
cpu_gpio_sel
gpio_cpu_rdy
gpio_cpu_data[31:0]
cpu_icu_sel
icu_cpu_rdy
icu_cpu_data[31:0]
cpu_lss_sel
lss_cpu_rdy
lss_cpu_data[31:0]
cpu_pcu_sel
pcu_cpu_rdy
pcu_cpu_data[31:0]
cpu_scb_sel
scb_cpu_rdy
scb_cpu_data[31:0]
cpu_tim_sel
tim_cpu_rdy
tim_cpu_data[31:0]
cpu_rom_sel
rom_cpu_rdy
rom_cpu_data[31:0]
cpu_pss_sel
pss_cpu_rdy
pss_cpu_data[31:0]
cpu_diu_sel
diu_cpu_rdy
diu_cpu_data[31:0]
diu_cpu_berr
pss_cpu_berr
rom_cpu_berr
tim_cpu_berr
scb_cpu_berr
pcu_cpu_berr
lss_cpu_berr
icu_cpu_berr
gpio_cpu_berr
cpr_cpu_berr
diu_cpu_debug_valid
tim_cpu_debug_valid
scb_cpu_debug_valid
pcu_cpu_debug_valid
lss_cpu_debug_valid
icu_cpu_debug_valid
gpio_cpu_debug_valid
cpr_cpu_debug_valid
debug_data_out[31:0]
debug_data_valid
debug_cntrl[32:0]
prst_n
pclk
icu_cpu_ilevel[3:0]
cpu_iack
cpu_icu_ilevel[3:0]

FIG. 15

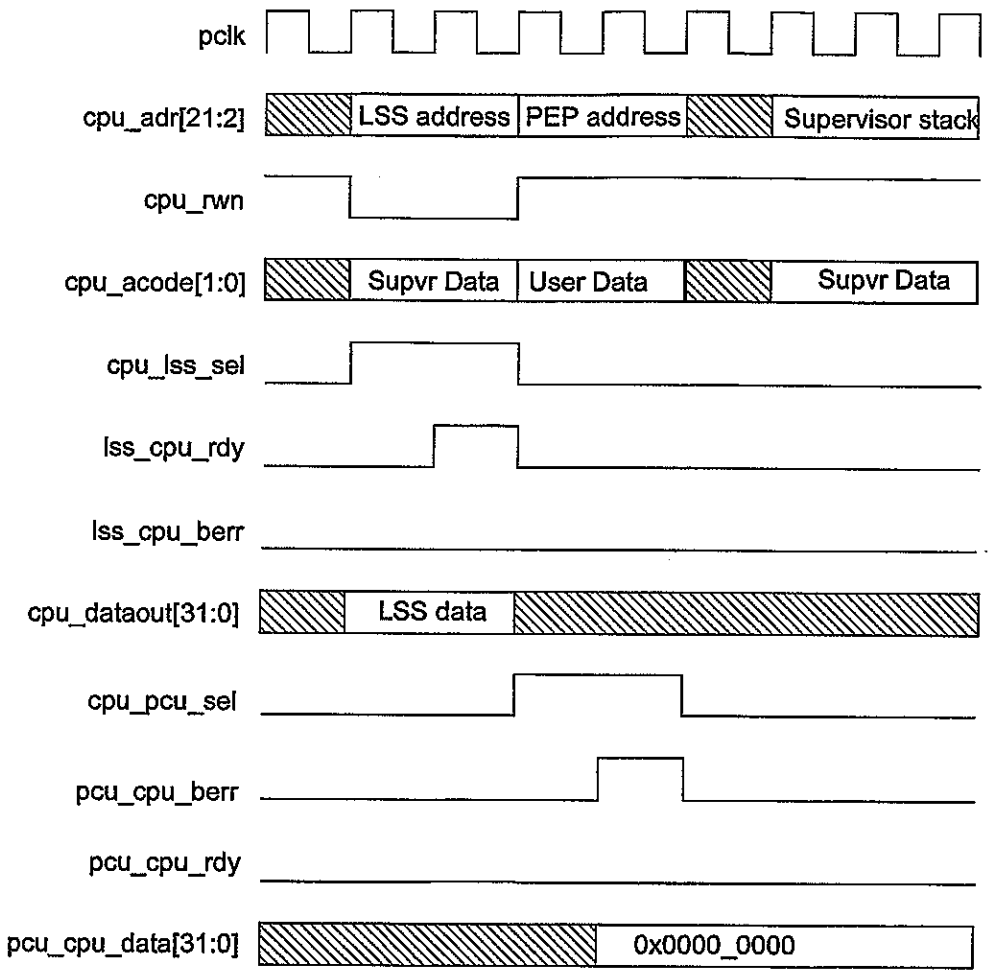

pclk
cpu_adr[21:2]
LSS address
PEP address
Supervisor stack
cpu_rwn
cpu_acode[1:0]
Supvr Data
User Data
Supvr Data
cpu_lss_sel
lss_cpu_rdy
lss_cpu_berr
cpu_dataout[31:0]
LSS data
cpu_pcu_sel
pcu_cpu_berr
pcu_cpu_rdy
pcu_cpu_data[31:0]
0x0000_0000

FIG. 16

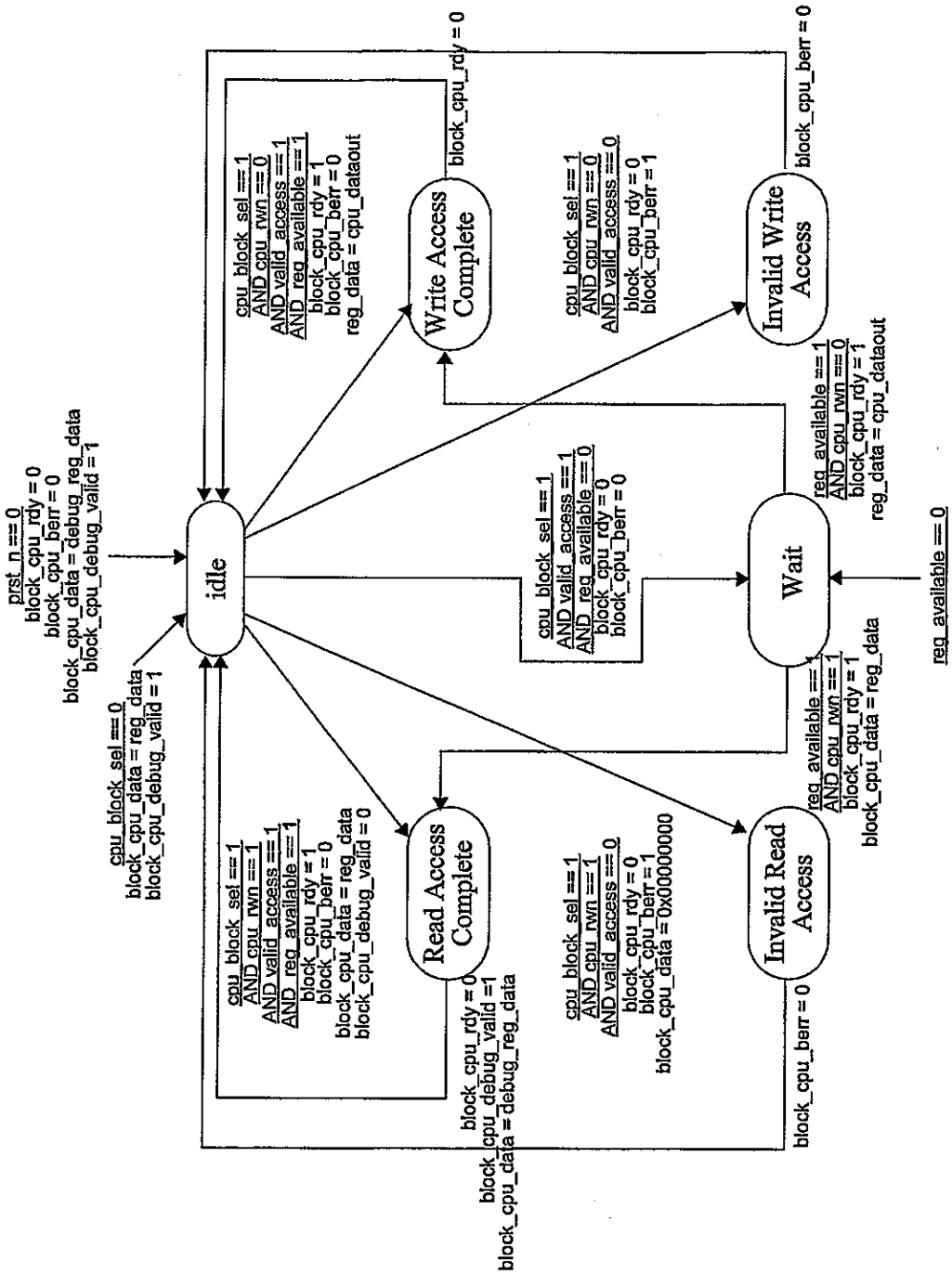

idle
Write Access Complete
Invalid Write Access
Wait
Read Access Complete
Invalid Read Access
prst_n == 0
block_cpu_rdy = 0
block_cpu_berr = 0
block_cpu_data = debug_reg_data
block_cpu_debug_valid = 1
cpu_block_sel == 0
block_cpu_data = reg_data
block_cpu_debug_valid = 1
cpu_block_sel == 1
AND cpu_rwn == 1
AND valid_access == 1
AND reg_available == 1
block_cpu_rdy = 1
block_cpu_berr = 0
block_cpu_data = reg_data
block_cpu_debug_valid = 0
block_cpu_rdy = 0
block_cpu_debug_valid =1
block_cpu_data = debug_reg_data
cpu_block_sel == 1
AND cpu_rwn == 1
AND valid_access == 1
AND reg_available == 1
block_cpu_rdy = 1
block_cpu_berr = 0
reg_data = cpu_dataout
block_cpu_rdy = 0
cpu_block_sel == 1
AND valid_access == 1
AND reg_available == 0
block_cpu_rdy = 0
block_cpu_berr = 0
cpu_block_sel == 1
AND valid_access == 0
block_cpu_rdy = 0
block_cpu_berr = 1
block_cpu_data = 0x00000000
block_cpu_berr = 0
cpu_block_sel == 1
AND valid_access == 0
block_cpu_rdy = 0
block_cpu_berr = 0
reg_available == 1
AND cpu_rwn == 0
block_cpu_rdy = 1
reg_data = cpu_dataout
reg_available == 1
AND cpu_rwn == 1
block_cpu_rdy = 1
block_cpu_data = reg_data
reg_available == 0

FIG. 17

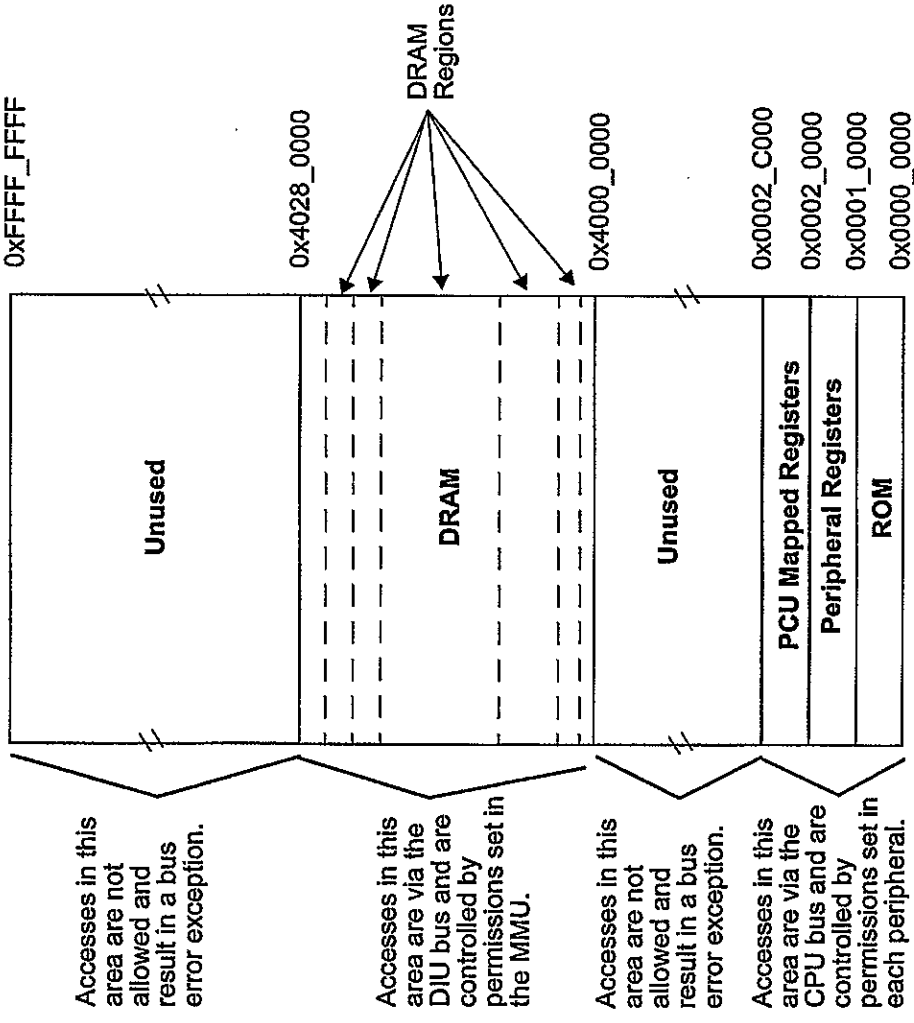

0xFFFF_FFFF
0x4028_0000
DRAM Regions
0x4000_0000
0x0002_C000
0x0002_0000
0x0001_0000
0x0000_0000
Unused
DRAM
Unused
PCU Mapped Registers
Peripheral Registers
ROM
Accesses in this area are not allowed and result in a bus error exception.
Accesses in this area are via the DIU bus and are controlled by permissions set in the MMU.
Accesses in this area are not allowed and result in a bus error exception.
Accesses in this area are via the CPU bus and are controlled by permissions set in each peripheral.

FIG. 18

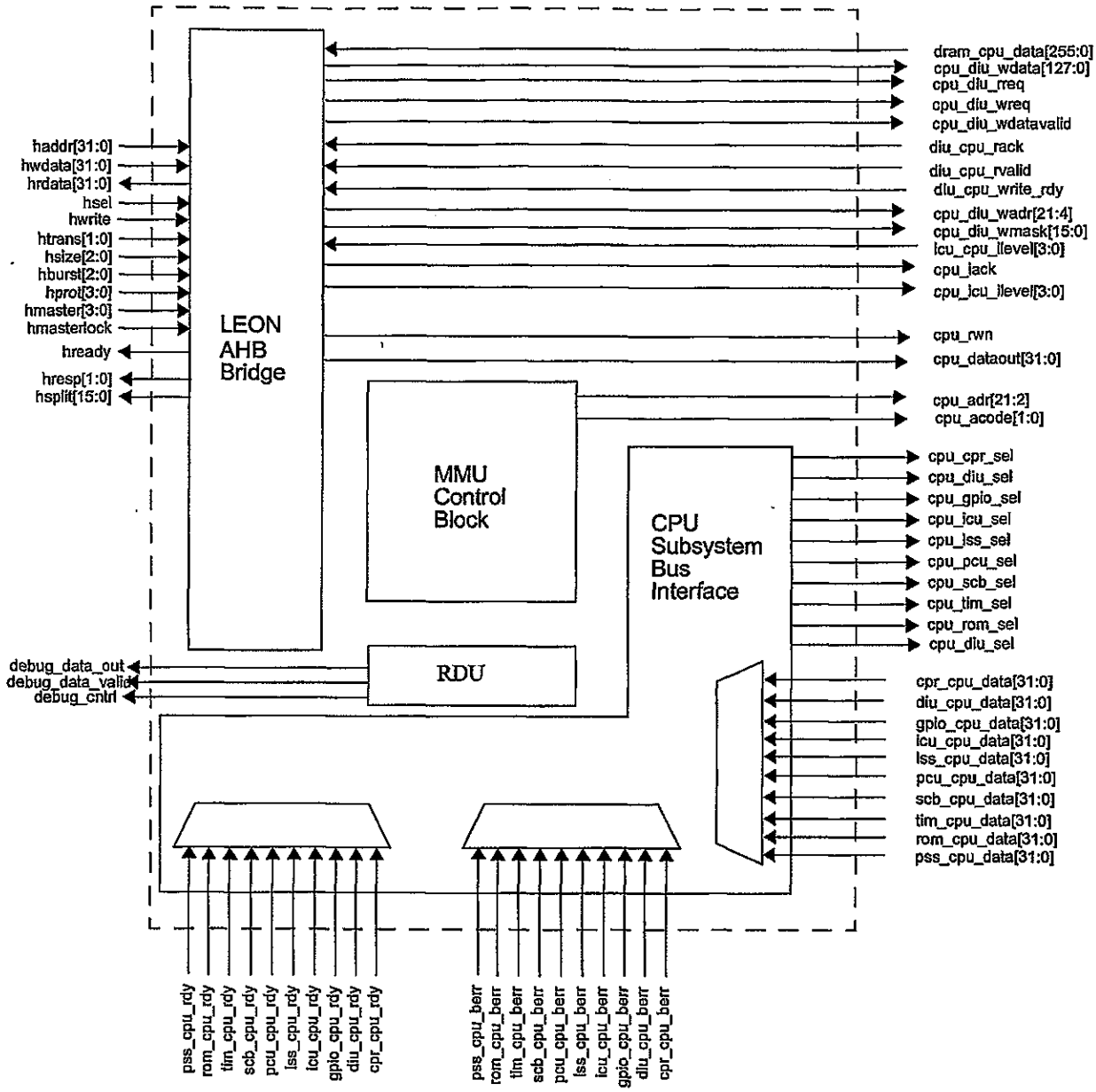

haddr[31:0]
hwdata[31:0]
hrdata[31:0]
hsel
hwrite
htrans[1:0]
hsize[2:0]
hburst[2:0]
hprot[3:0]
hmaster[3:0]
hmasterlock
hready
hresp[1:0]
hsplit[15:0]
LEON AHB Bridge
MMU Control Block
CPU Subsystem Bus Interface
RDU
debug_data_out
debug_data_valid
debug_cntrl
dram_cpu_data[255:0]
cpu_diu_wdata[127:0]
cpu_diu_rreq
cpu_diu_wreq
cpu_diu_wdatavalid
diu_cpu_rack
diu_cpu_rvalid
diu_cpu_write_rdy
cpu_diu_wadr[21:4]
cpu_diu_wmask[15:0]
icu_cpu_ilevel[3:0]
cpu_iack
cpu_icu_ilevel[3:0]
cpu_rwn
cpu_dataout[31:0]
cpu_adr[21:2]
cpu_acode[1:0]
cpu_cpr_sel
cpu_diu_sel
cpu_gpio_sel
cpu_icu_sel
cpu_lss_sel
cpu_pcu_sel
cpu_scb_sel
cpu_tim_sel
cpu_rom_sel
cpu_dlu_sel
cpr_cpu_data[31:0]
diu_cpu_data[31:0]
gpio_cpu_data[31:0]
icu_cpu_data[31:0]
lss_cpu_data[31:0]
pcu_cpu_data[31:0]
scb_cpu_data[31:0]
tim_cpu_data[31:0]
rom_cpu_data[31:0]
pss_cpu_data[31:0]
pss_cpu_rdy
rom_cpu_rdy
tim_cpu_rdy
scb_cpu_rdy
pcu_cpu_rdy
lss_cpu_rdy
icu_cpu_rdy
gpio_cpu_rdy
diu_cpu_rdy
cpr_cpu_rdy
pss_cpu_berr
rom_cpu_berr
tim_cpu_berr
scb_cpu_berr
pcu_cpu_berr
lss_cpu_berr
icu_cpu_berr
gpio_cpu_berr
diu_cpu_berr
cpr_cpu_berr

FIG. 19

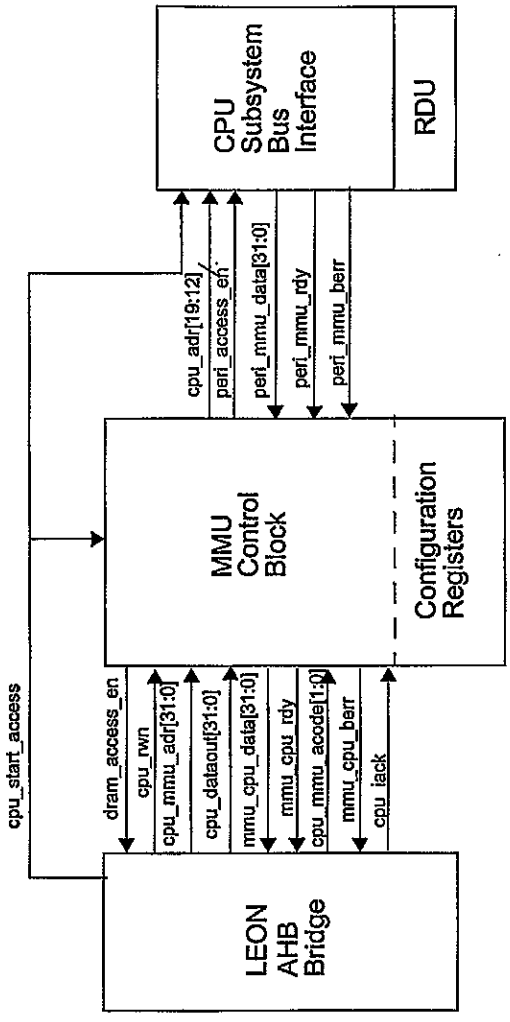

CPU Subsystem Bus Interface
RDU
MMU Control Block
Configuration Registers
LEON AHB Bridge
cpu_start_access
dram_access_en
cpu_rwn
cpu_mmu_adr[31:0]
cpu_dataout[31:0]
mmu_cpu_data[31:0]
mmu_cpu_rdy
cpu_mmu_acode[1:0]
mmu_cpu_berr
cpu_lack
cpu_adr[19:12]
peri_access_en
peri_mmu_data[31:0]
peri_mmu_rdy
peri_mmu_berr

FIG. 20

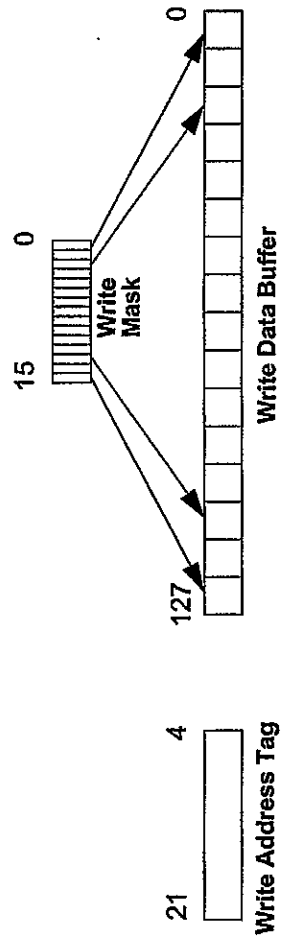

15
0
Write Mask
0
127
Write Data Buffer
21
4
Write Address Tag

FIG. 21

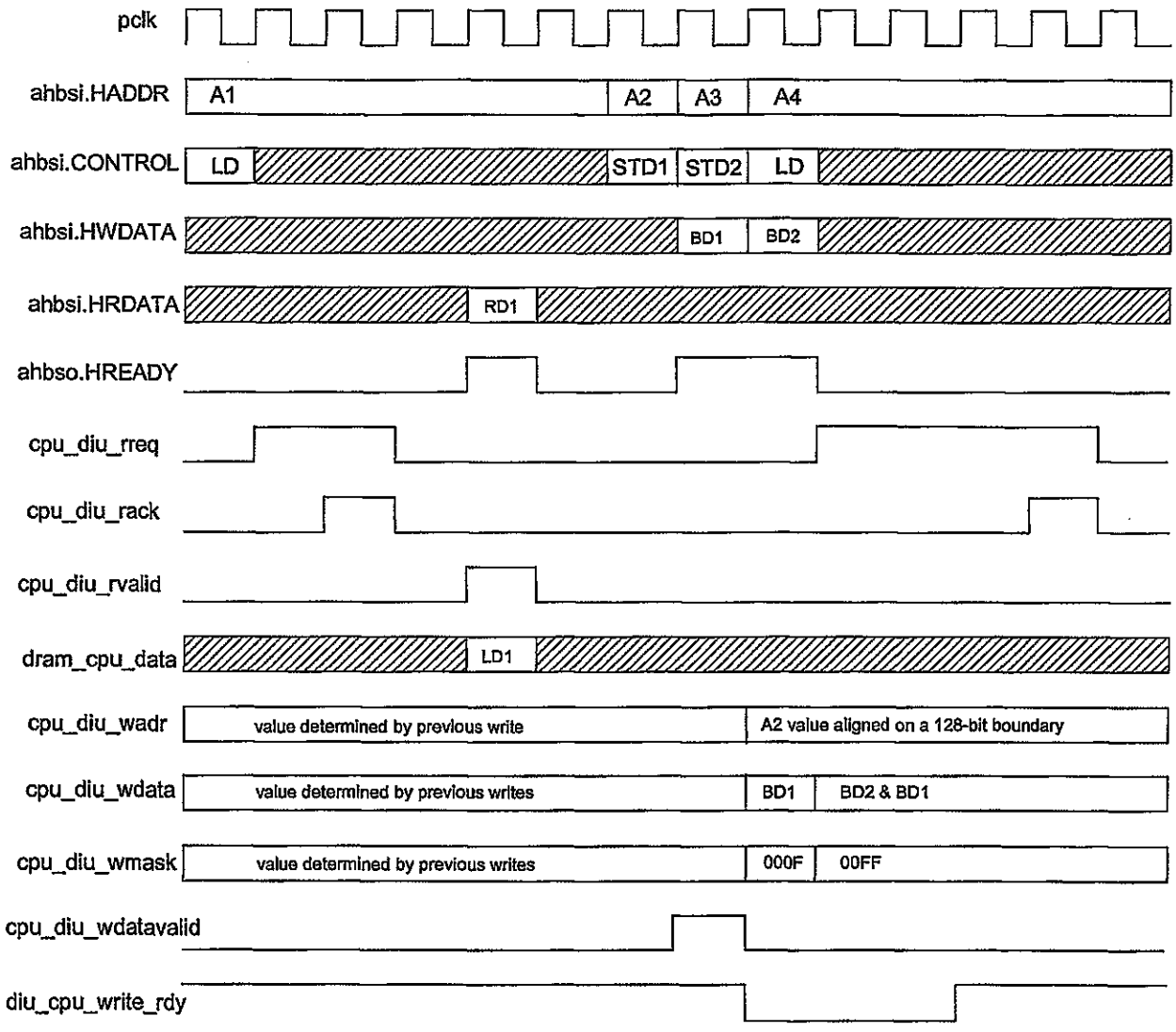

pclk
ahbsi.HADDR
A1
A2
A3
A4
ahbsi.CONTROL
LD
STD1
STD2
LD
ahbsi.HWDATA
BD1
BD2
ahbsi.HRDATA
RD1
ahbso.HREADY
cpu_diu_rreq
cpu_diu_rack
cpu_diu_rvalid
dram_cpu_data
LD1
cpu_diu_wadr
value determined by previous write
A2 value aligned on a 128-bit boundary
cpu_diu_wdata
value determined by previous writes
BD1
BD2 & BD1
cpu_diu_wmask
value determined by previous writes
000F
00FF
cpu_diu_wdatavalid
diu_cpu_write_rdy

FIG. 22

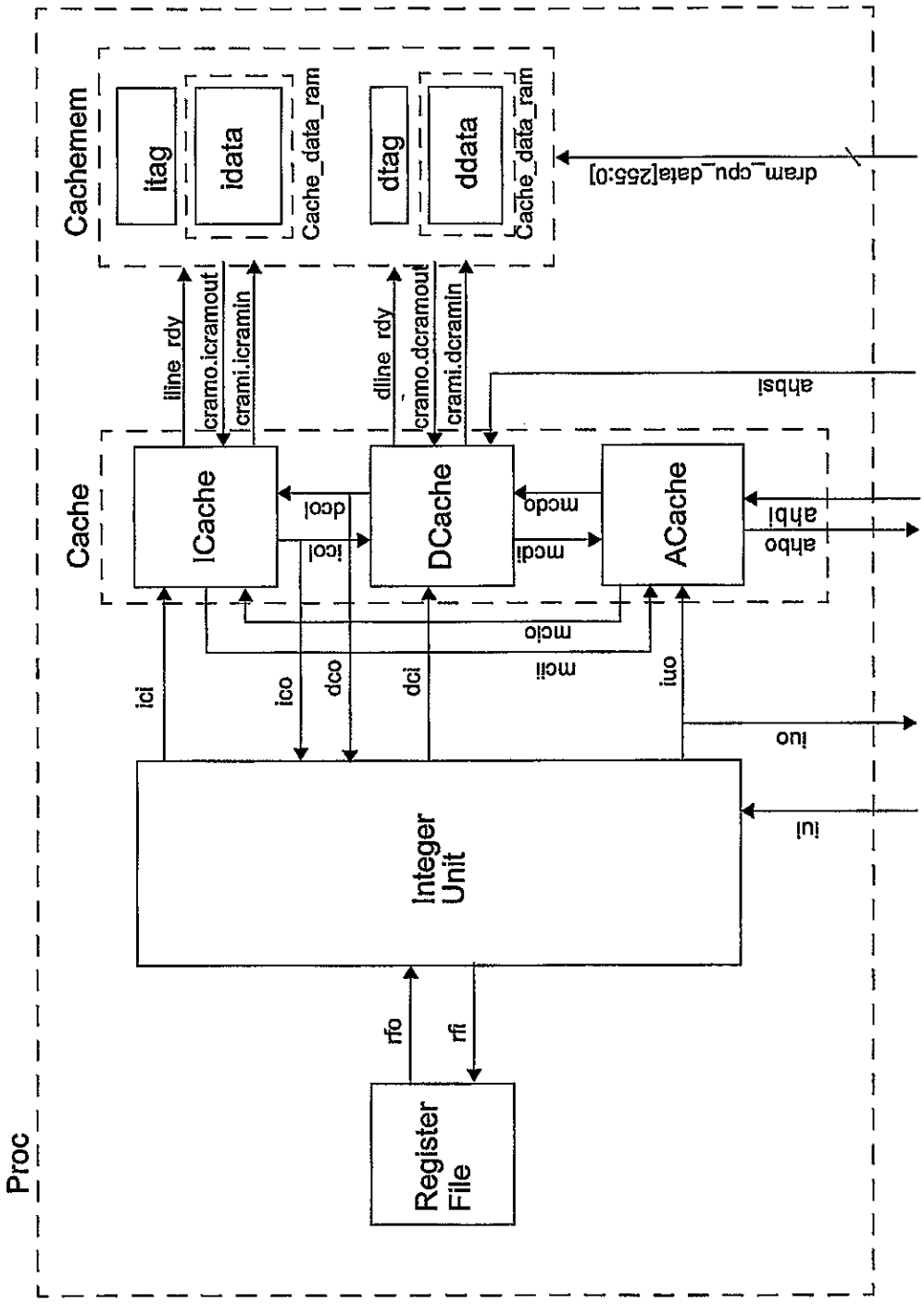

Proc
Cachemem
itag
idata
Cache_data_ram
dtag
ddata
Cache_data_ram
dram_cpu_data[255:0]
iline_rdy
cramo.icramout
crami.icramin
dline_rdy
cramo.dcramout
crami.dcramin
ahbsi
Cache
ICache
DCache
ACache
dcol
icol
mcdo
mcdi
ahbi
ahbo
mcio
mcii
ici
ico
dco
dci
oni
iuo
iui
Integer Unit
rfo
rfi
Register File

FIG. 23

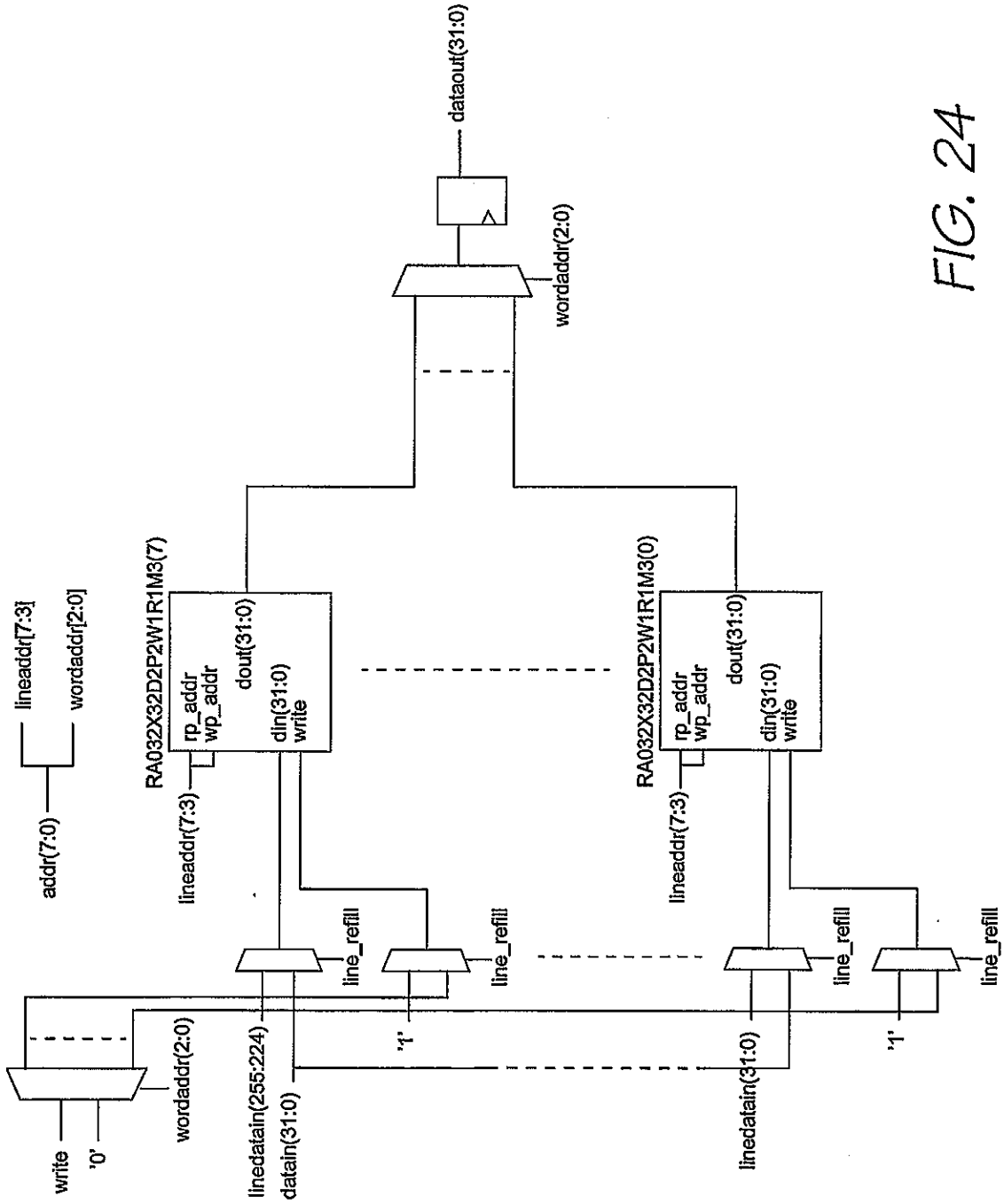

dataout(31:0)
wordaddr(2:0)
RA032X32D2P2W1R1M3(7)
rp_addr
wp_addr
dout(31:0)
din(31:0)
write
RA032X32D2P2W1R1M3(0)
lineaddr[7:3]
wordaddr[2:0]
addr(7:0)
lineaddr(7:3)
line_refill
'1'
write
'0'
linedatain(255:224)
datain(31:0)
linedatain(31:0)

FIG. 24

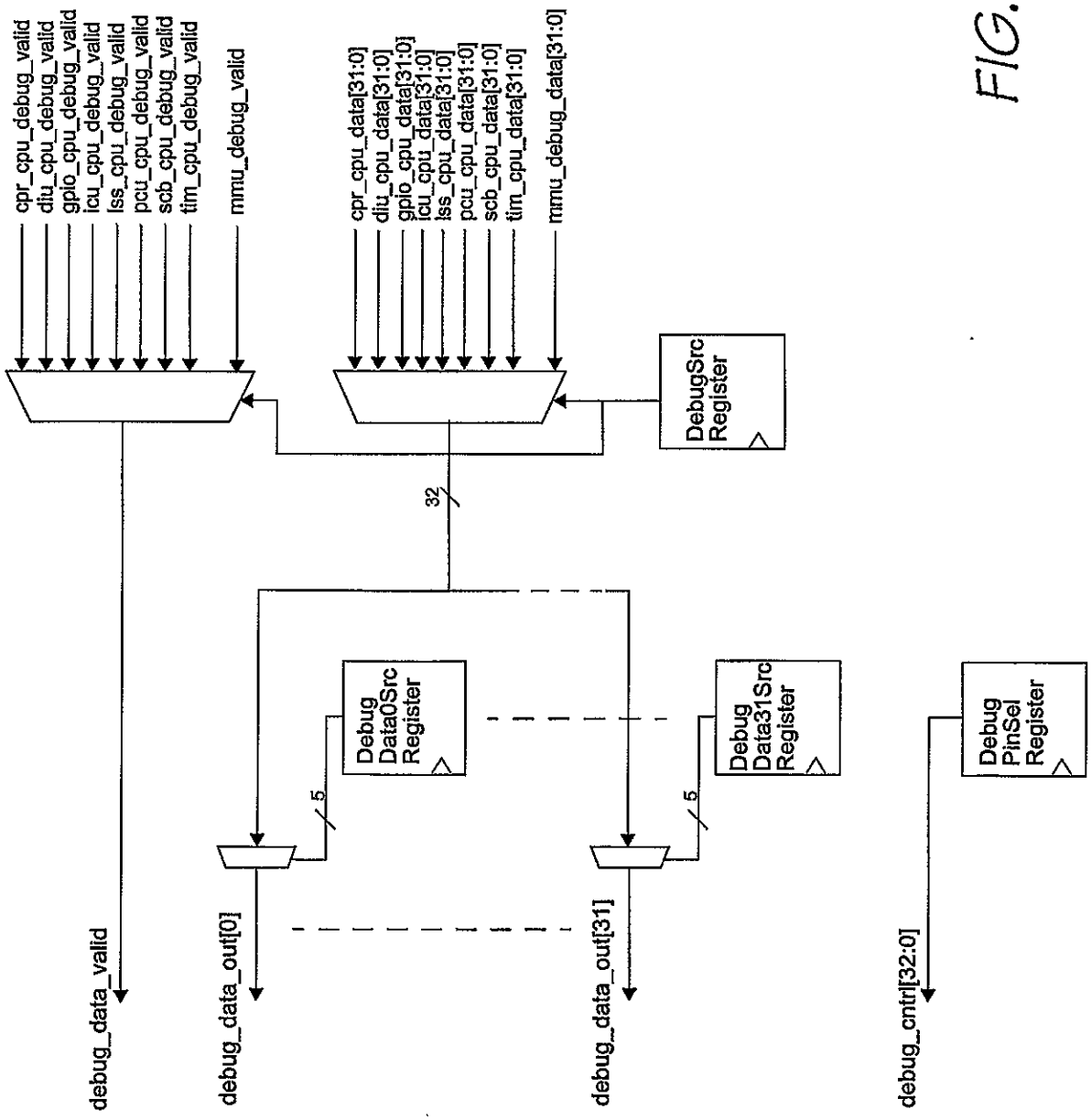

cpr_cpu_debug_valid
diu_cpu_debug_valid
gpio_cpu_debug_valid
icu_cpu_debug_valid
lss_cpu_debug_valid
pcu_cpu_debug_valid
scb_cpu_debug_valid
tim_cpu_debug_valid
mmu_debug_valid
cpr_cpu_data[31:0]
diu_cpu_data[31:0]
gpio_cpu_data[31:0]
icu_cpu_data[31:0]
lss_cpu_data[31:0]
pcu_cpu_data[31:0]
scb_cpu_data[31:0]
tim_cpu_data[31:0]
mmu_debug_data[31:0]
DebugSrc Register
32
Debug Data0Src Register
Debug Data31Src Register
5
5
Debug PinSel Register
debug_data_valid
debug_data_out[0]
debug_data_out[31]
debug_cntrl[32:0]

FIG. 25

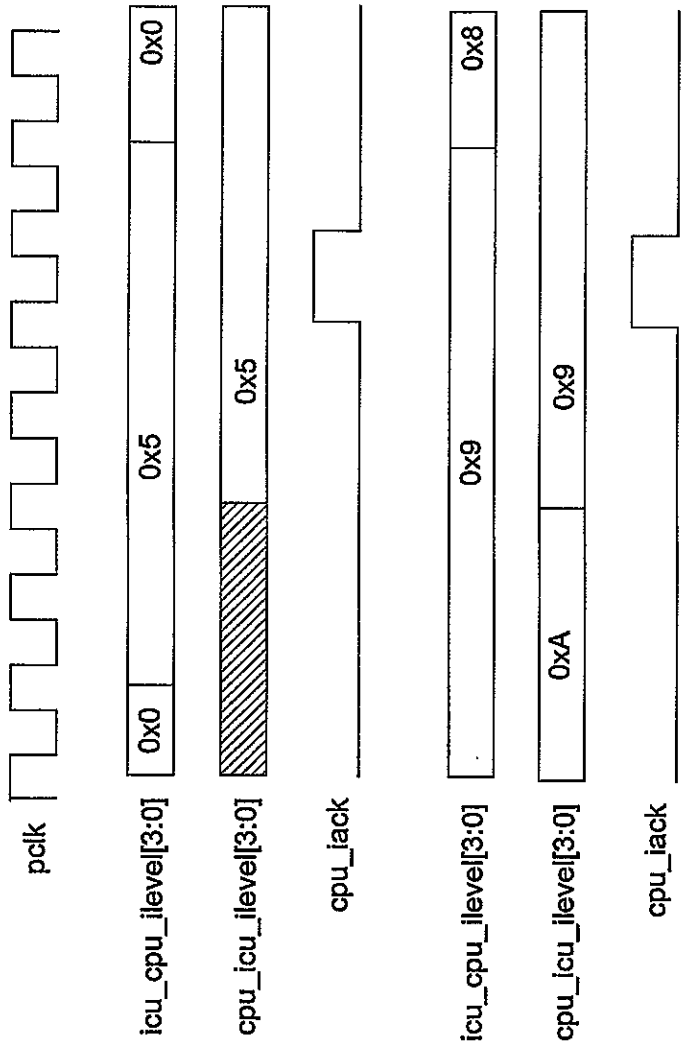

pclk
icu_cpu_ilevel[3:0]
0x0
0x5
0x0
cpu_icu_ilevel[3:0]
0x5
cpu_iack
icu_cpu_ilevel[3:0]
0x9
0x8
cpu_icu_ilevel[3:0]
0xA
0x9
cpu_iack

FIG. 26

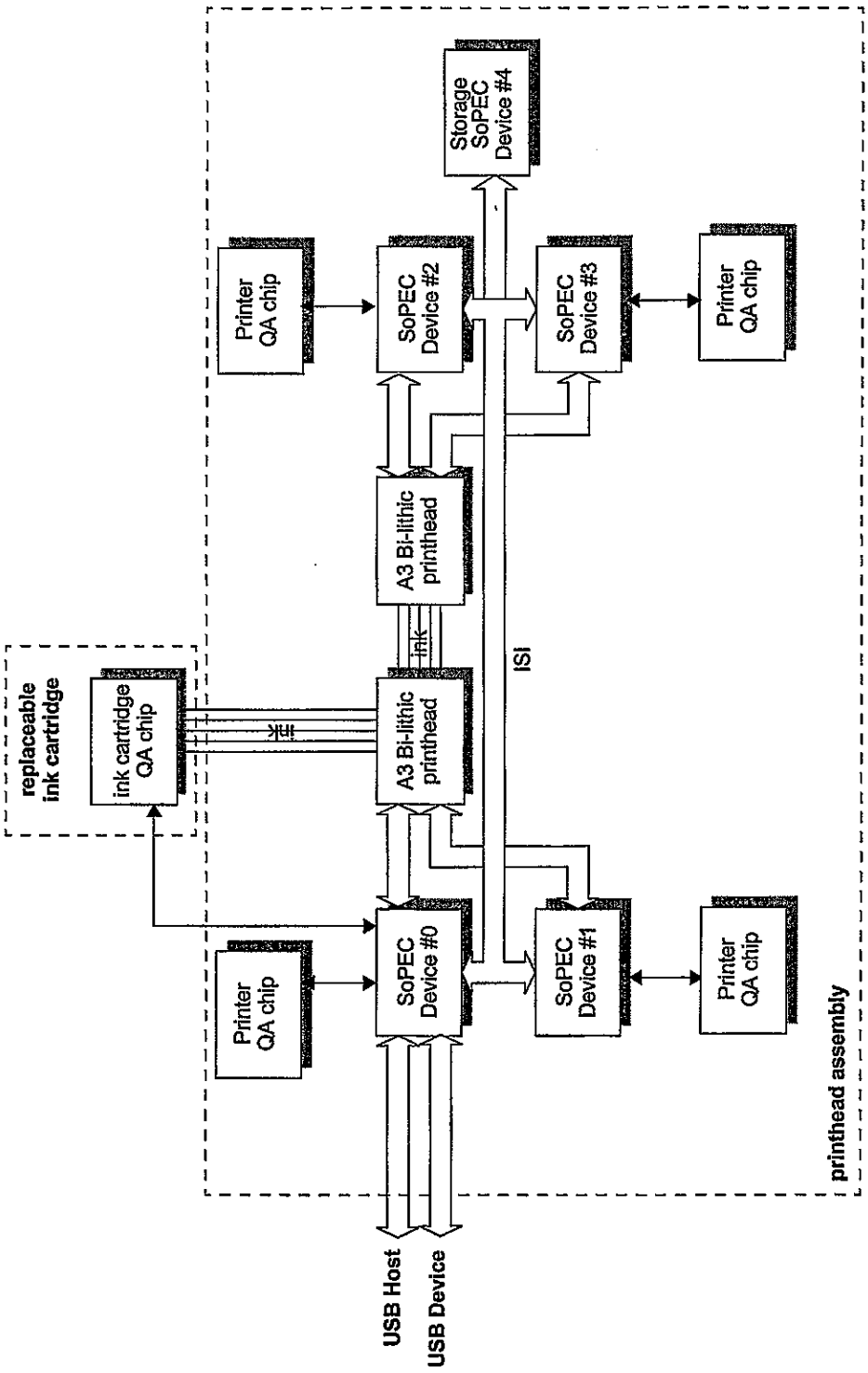

printhead assembly
replaceable ink cartridge
ink cartridge QA chip
ink
A3 Bi-lithic printhead
A3 Bi-lithic printhead
ISI
Storage SoPEC Device #4
Printer QA chip
SoPEC Device #2
SoPEC Device #3
Printer QA chip
Printer QA chip
SoPEC Device #0
SoPEC Device #1
Printer QA chip
USB Host
USB Device

FIG. 27

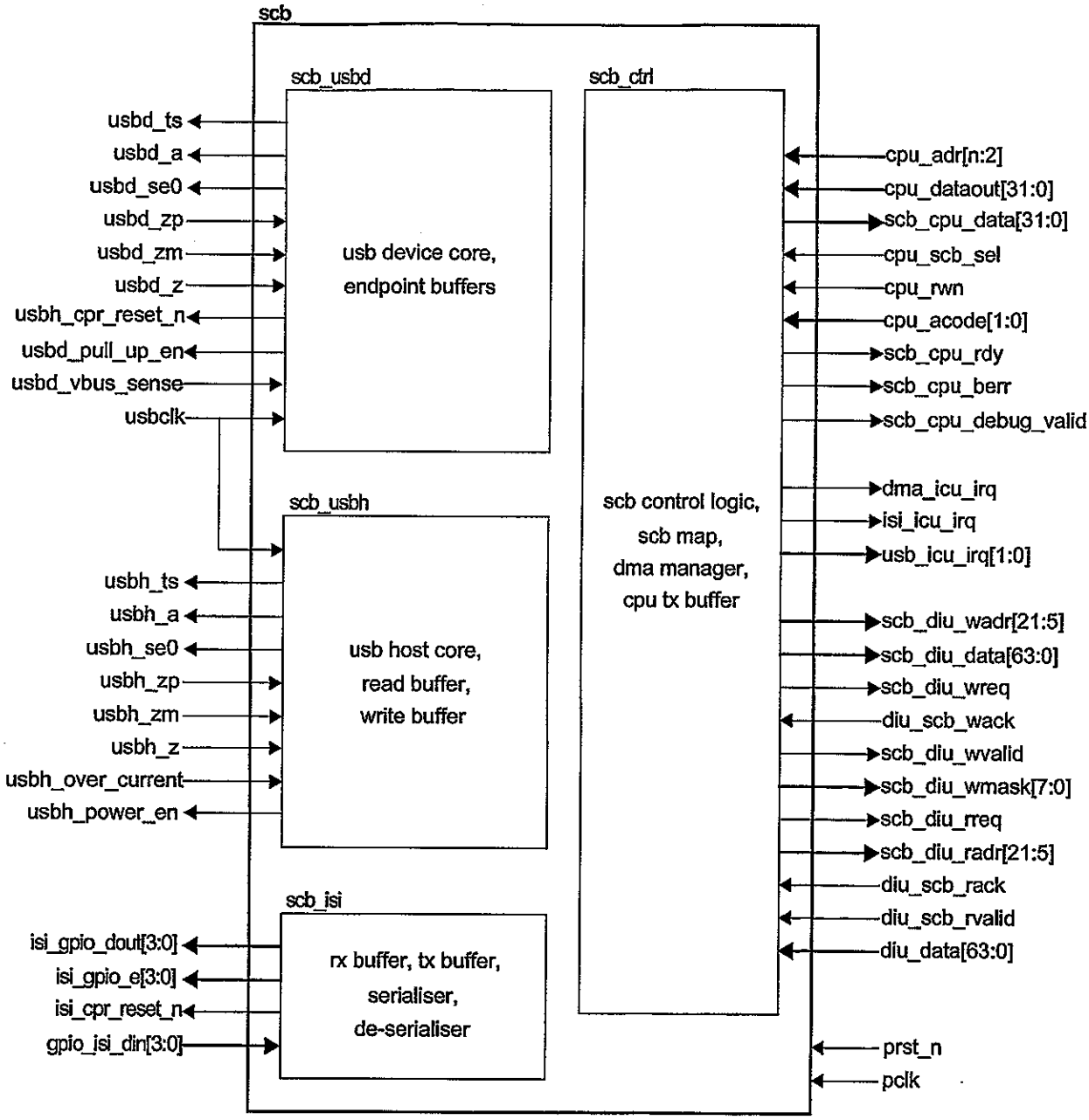

scb
scb_usbd
usb device core,
endpoint buffers
scb_ctrl
scb control logic,
scb map,
dma manager,
cpu tx buffer
scb_usbh
usb host core,
read buffer,
write buffer
scb_isi
rx buffer, tx buffer,
serialiser,
de-serialiser
usbd_ts
usbd_a
usbd_se0
usbd_zp
usbd_zm
usbd_z
usbh_cpr_reset_n
usbd_pull_up_en
usbd_vbus_sense
usbclk
usbh_ts
usbh_a
usbh_se0
usbh_zp
usbh_zm
usbh_z
usbh_over_current
usbh_power_en
isi_gpio_dout[3:0]
isi_gpio_e[3:0]
isi_cpr_reset_n
gpio_isi_din[3:0]
cpu_adr[n:2]
cpu_dataout[31:0]
scb_cpu_data[31:0]
cpu_scb_sel
cpu_rwn
cpu_acode[1:0]
scb_cpu_rdy
scb_cpu_berr
scb_cpu_debug_valid
dma_icu_irq
isi_icu_irq
usb_icu_irq[1:0]
scb_diu_wadr[21:5]
scb_diu_data[63:0]
scb_diu_wreq
diu_scb_wack
scb_diu_wvalid
scb_diu_wmask[7:0]
scb_diu_rreq
scb_diu_radr[21:5]
diu_scb_rack
diu_scb_rvalid
diu_data[63:0]
prst_n
pclk

FIG. 28

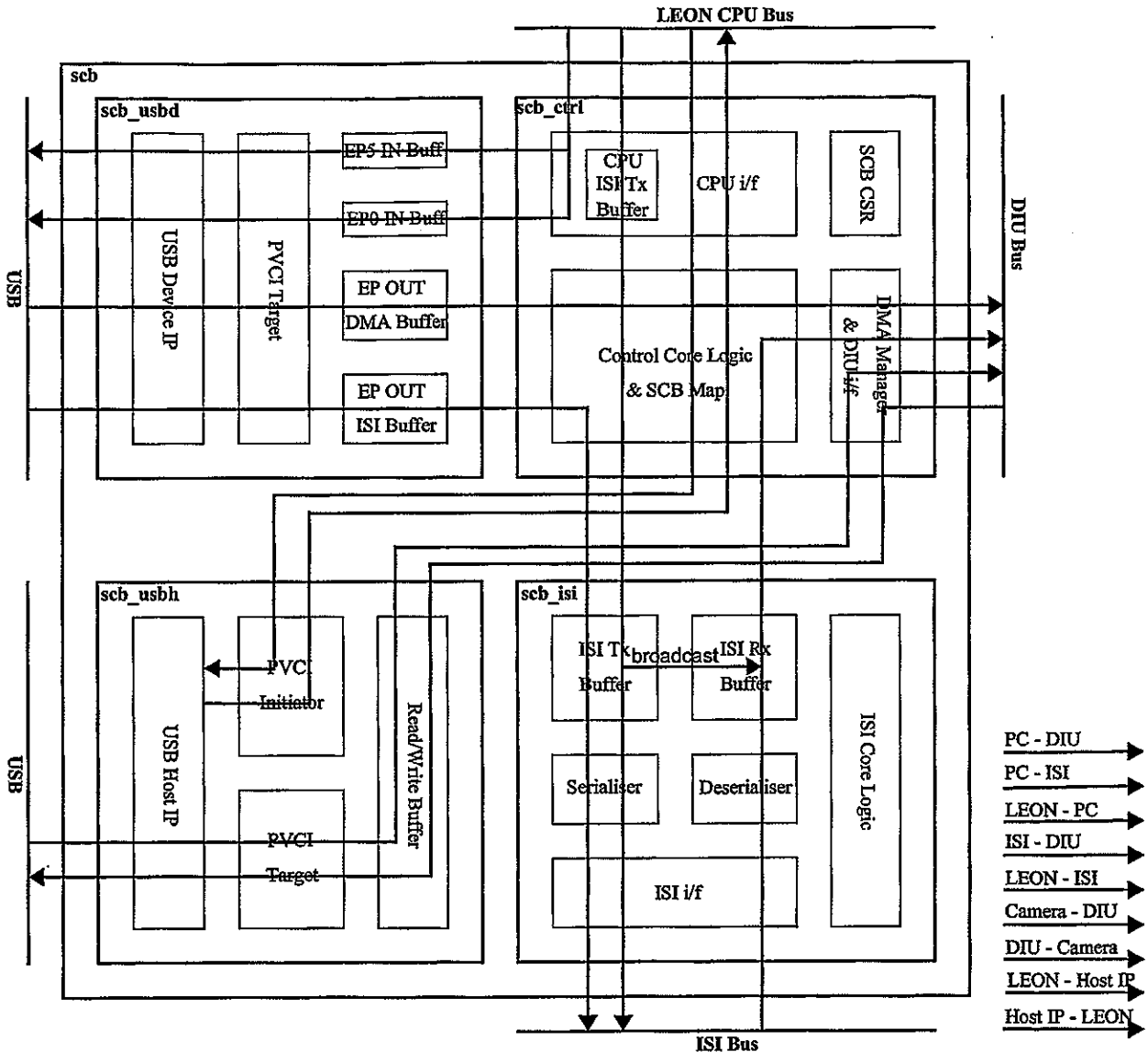

LEON CPU Bus
scb
scb_usbd
scb_ctrl
EP5 IN Buff
EP0 IN Buff
EP OUT
DMA Buffer
EP OUT
ISI Buffer
USB Device IP
PVCI Target
CPU
ISI Tx
Buffer
CPU i/f
SCB CSR
DMA Manager
& DIU i/f
Control Core Logic
& SCB Map
DIU Bus
USB
scb_usbh
scb_isi
PVC
Initiator
PVCI
Target
USB Host IP
Read/Write Buffer
ISI Tx
Buffer
broadcast
ISI Rx
Buffer
Serialiser
Deserialiser
ISI i/f
ISI Core Logic
ISI Bus
PC - DIU
PC - ISI
LEON - PC
ISI - DIU
LEON - ISI
Camera - DIU
DIU - Camera
LEON - Host IP
Host IP - LEON

FIG. 29

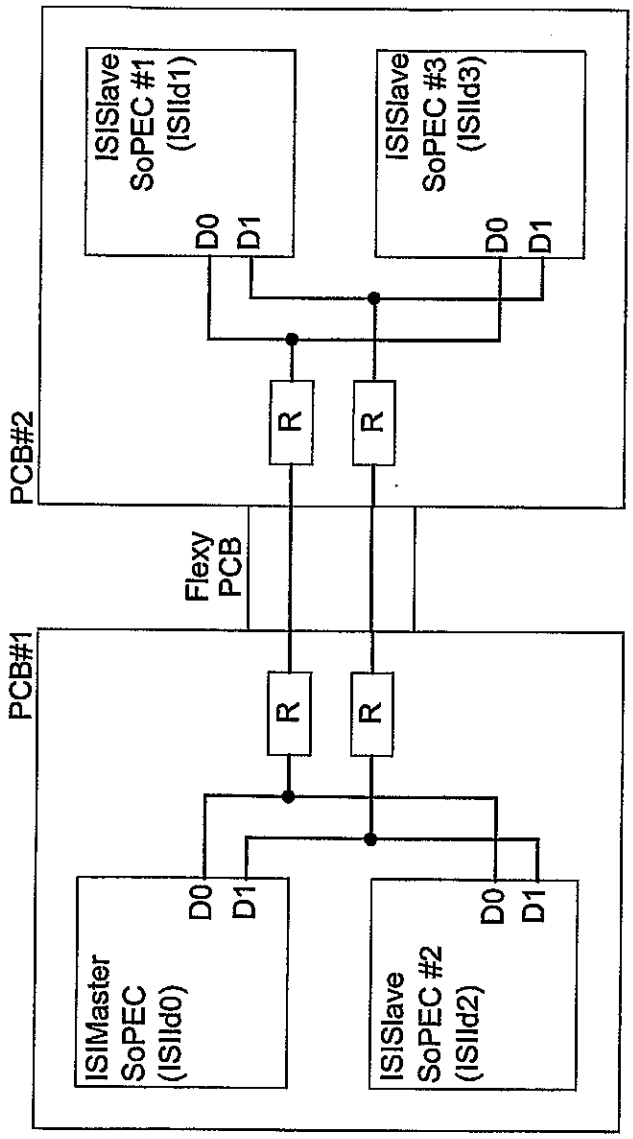

PCB#2
ISISlave SoPEC #1 (ISIId1)
D0
D1
ISISlave SoPEC #3 (ISIId3)
D0
D1
R
R
Flexy PCB
PCB#1
R
R
ISIMaster SoPEC (ISIId0)
D0
D1
ISISlave SoPEC #2 (ISIId2)
D0
D1

FIG. 30

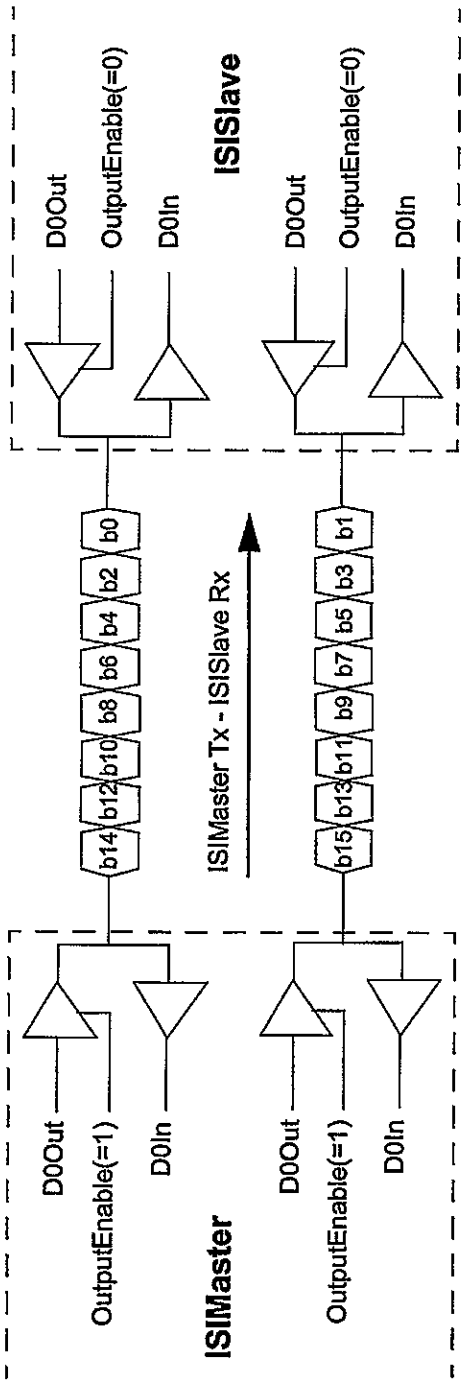
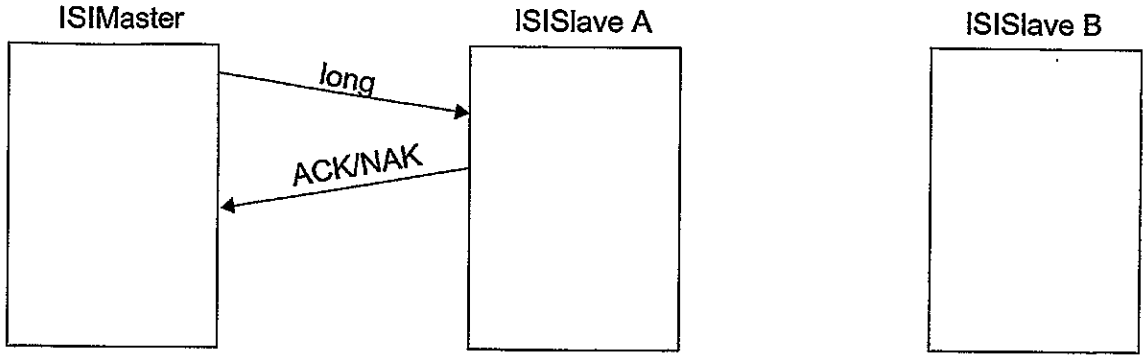
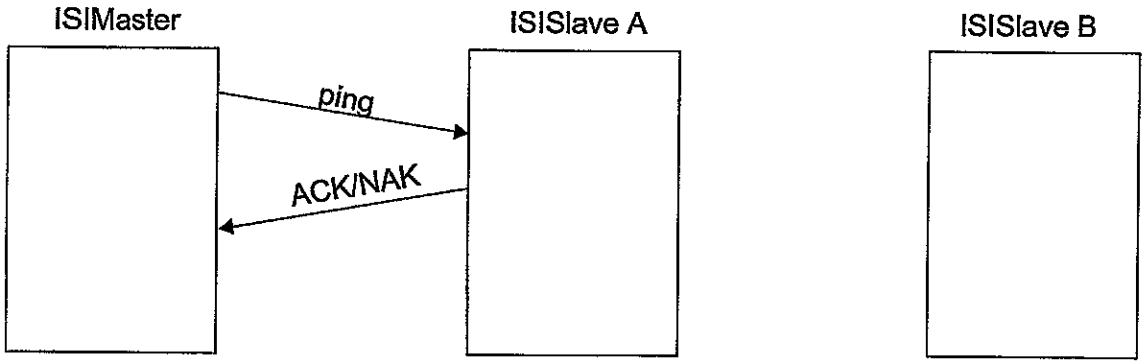

ISISlave
D0Out
OutputEnable(=0)
D0In
D0Out
OutputEnable(=0)
D0In
b0
b2
b4
b6
b8
b10
b12
b14
b1
b3
b5
b7
b9
b11
b13
b15
ISIMaster Tx - ISISlave Rx
ISIMaster
D0Out
OutputEnable(=1)
D0In
D0Out
OutputEnable(=1)
D0In

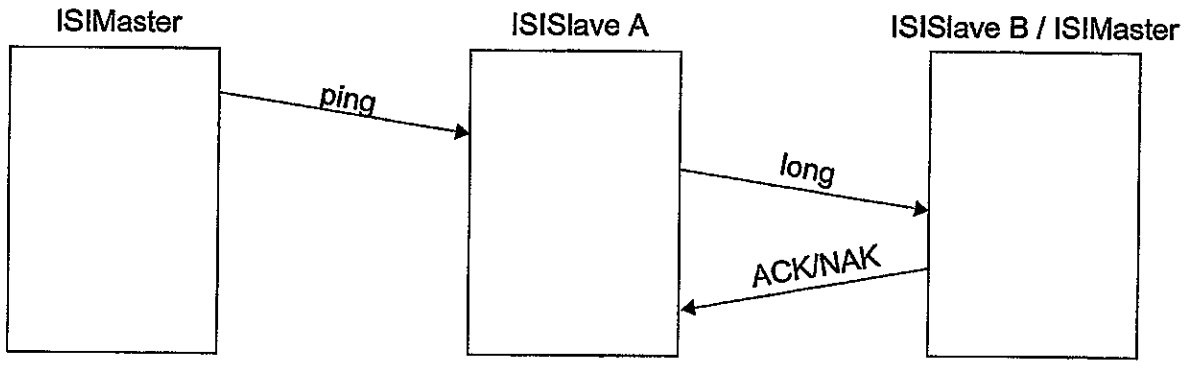
FIG. 31



Transaction 1: Long packet to an addressed ISISlave



Transaction 2: Ping packet to an addressed ISISlave. ISISlave has nothing to send



Transaction 3: Ping packet to an addressed ISISlave. ISISlaveA responds with a long packet to ISISlaveB (or the ISIMaster) and ISISlaveB (or the ISIMaster) responds with an ACK or NAK.

FIG. 32

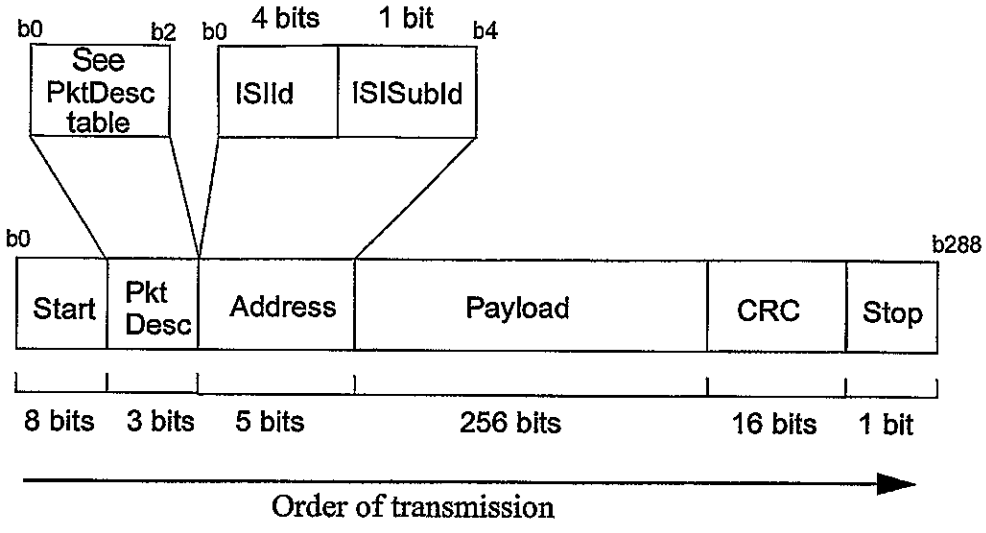

b0
b2
See PktDesc table
b0
4 bits
1 bit
b4
ISIId
ISISubId
b0
b288
Start
Pkt Desc
Address
Payload
CRC
Stop
8 bits
3 bits
5 bits
256 bits
16 bits
1 bit
Order of transmission

FIG. 33

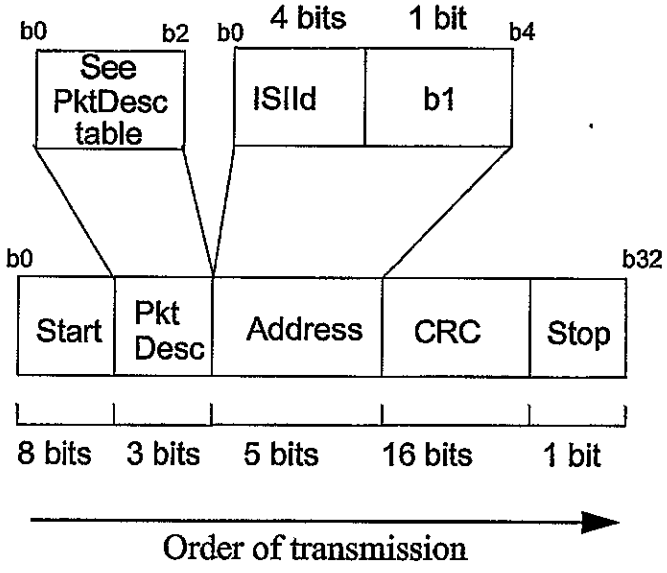

b0
b2
See PktDesc table
b0
4 bits
1 bit
b4
ISIId
b1
b0
b32
Start
Pkt Desc
Address
CRC
Stop
8 bits
3 bits
5 bits
16 bits
1 bit
Order of transmission

FIG. 34

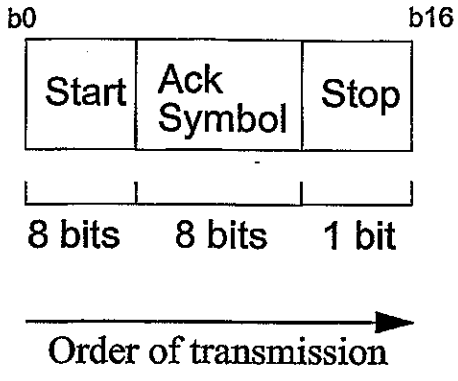

b0
b16
Start
Ack Symbol
Stop
8 bits
8 bits
1 bit
Order of transmission

FIG. 35

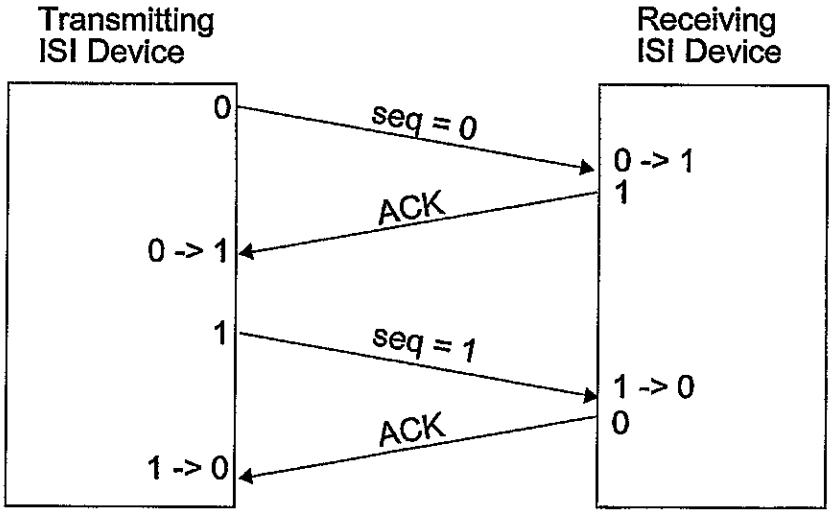

Transmitting ISI Device
Receiving ISI Device
0
seq = 0
0 -> 1
1
ACK
0 -> 1
1
seq = 1
1 -> 0
0
ACK
1 -> 0

FIG. 36

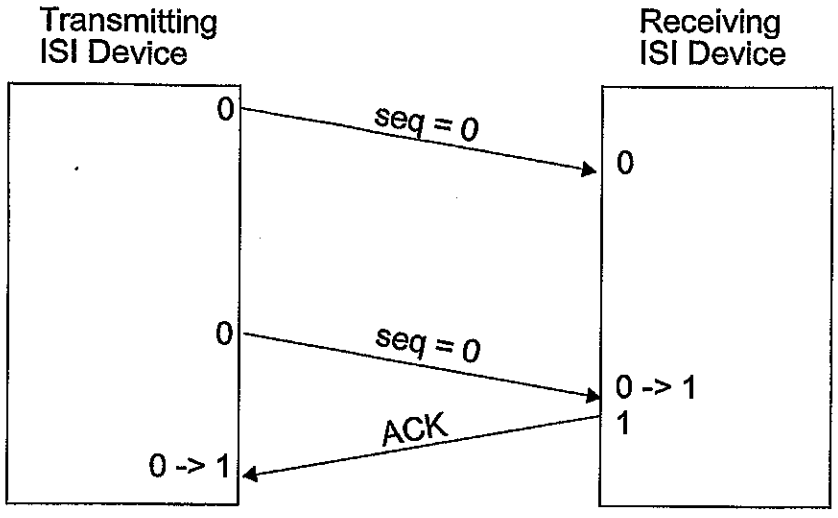

Transmitting
ISI Device
Receiving
ISI Device
0
seq = 0
0
0
seq = 0
0 -> 1
1
ACK
0 -> 1

FIG. 37

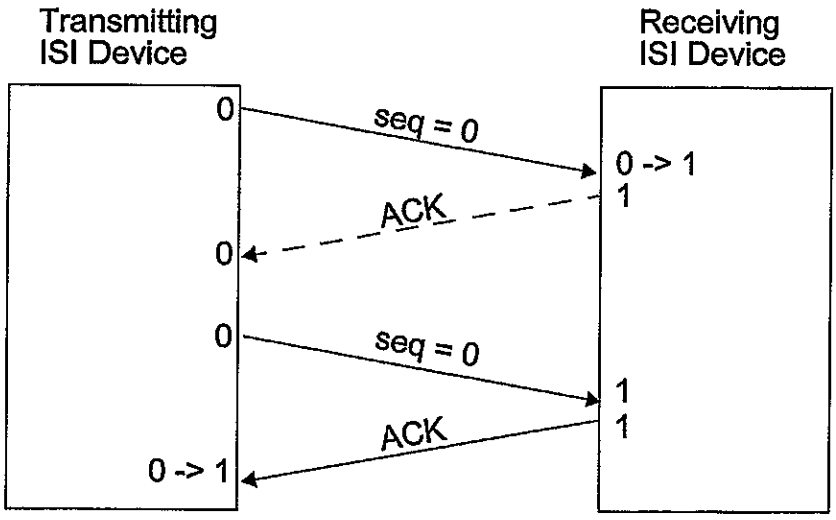

Transmitting
ISI Device
Receiving
ISI Device
0
seq = 0
0 -> 1
1
ACK
0
0
seq = 0
1
1
ACK
0 -> 1

FIG. 38

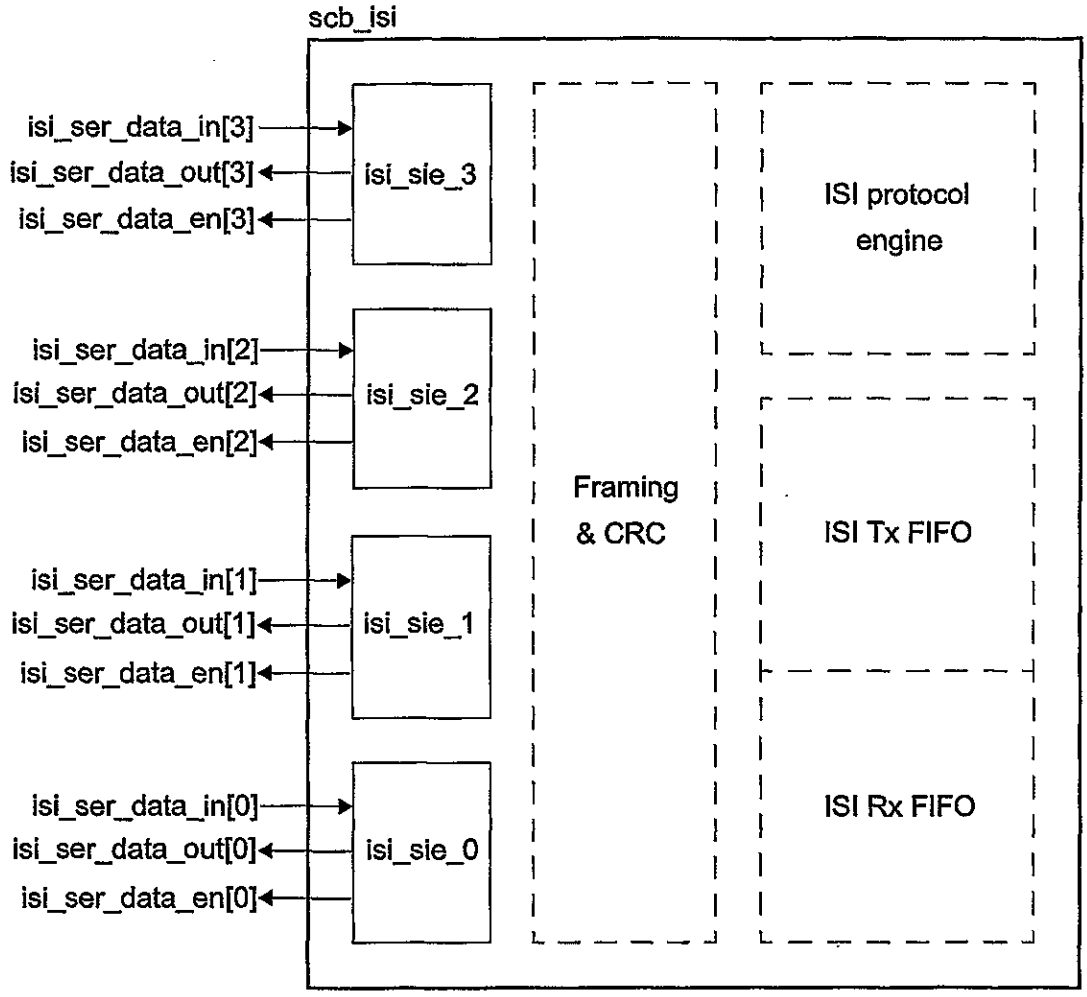

scb_isi
isi_ser_data_in[3]
isi_ser_data_out[3]
isi_ser_data_en[3]
isi_sie_3
isi_ser_data_in[2]
isi_ser_data_out[2]
isi_ser_data_en[2]
isi_sie_2
isi_ser_data_in[1]
isi_ser_data_out[1]
isi_ser_data_en[1]
isi_sie_1
isi_ser_data_in[0]
isi_ser_data_out[0]
isi_ser_data_en[0]
isi_sie_0
Framing & CRC
ISI protocol engine
ISI Tx FIFO
ISI Rx FIFO

FIG. 39

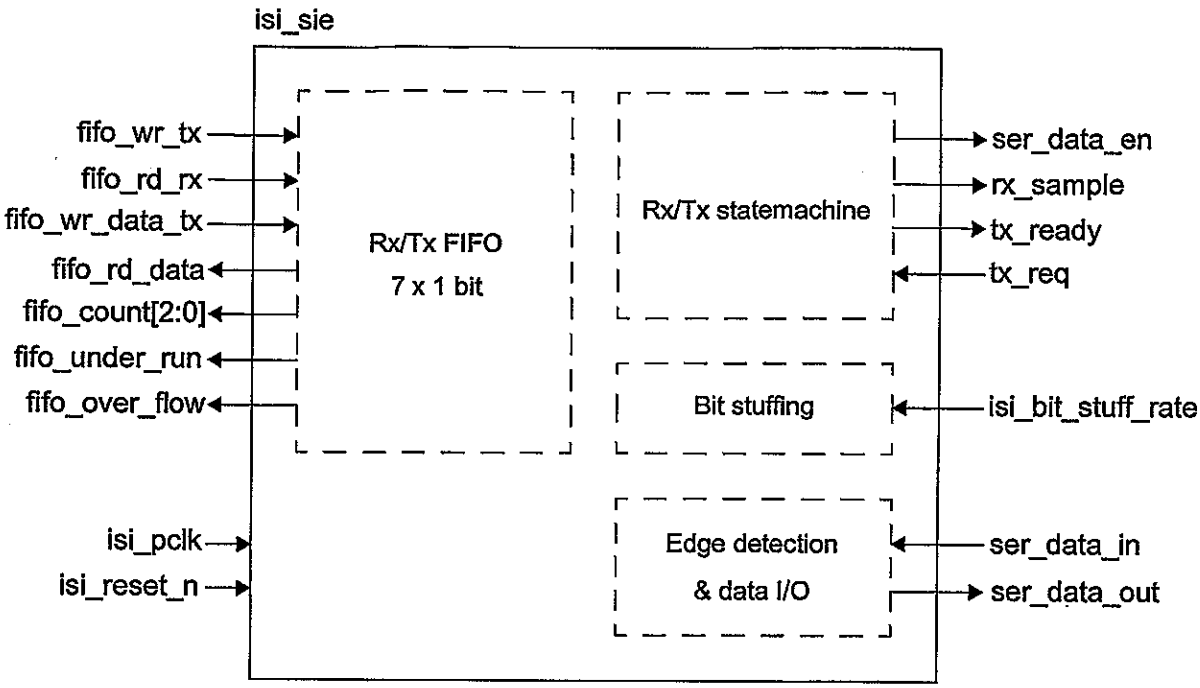

isi_sie
fifo_wr_tx
fifo_rd_rx
fifo_wr_data_tx
fifo_rd_data
fifo_count[2:0]
fifo_under_run
fifo_over_flow
Rx/Tx FIFO
7 x 1 bit
Rx/Tx statemachine
Bit stuffing
Edge detection
& data I/O
ser_data_en
rx_sample
tx_ready
tx_req
isi_bit_stuff_rate
ser_data_in
ser_data_out
isi_pclk
isi_reset_n

FIG. 40

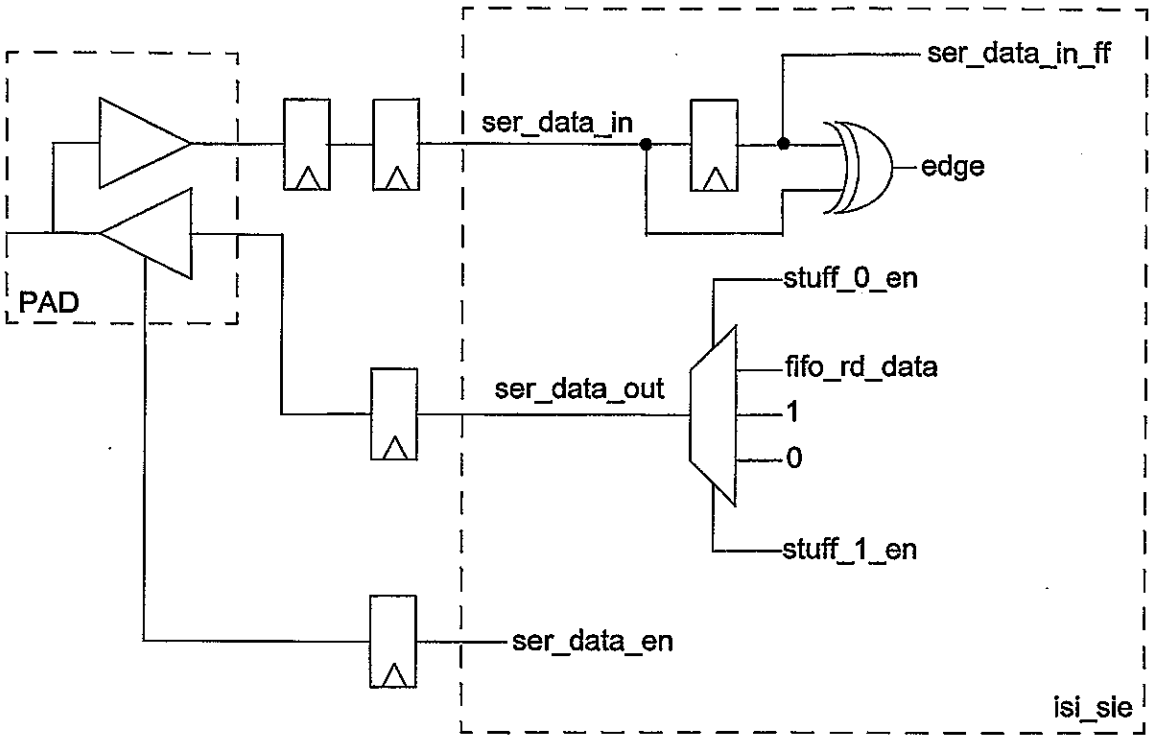

ser_data_in_ff
ser_data_in
edge
PAD
stuff_0_en
fifo_rd_data
ser_data_out
1
0
stuff_1_en
ser_data_en
isi_sie

FIG. 41

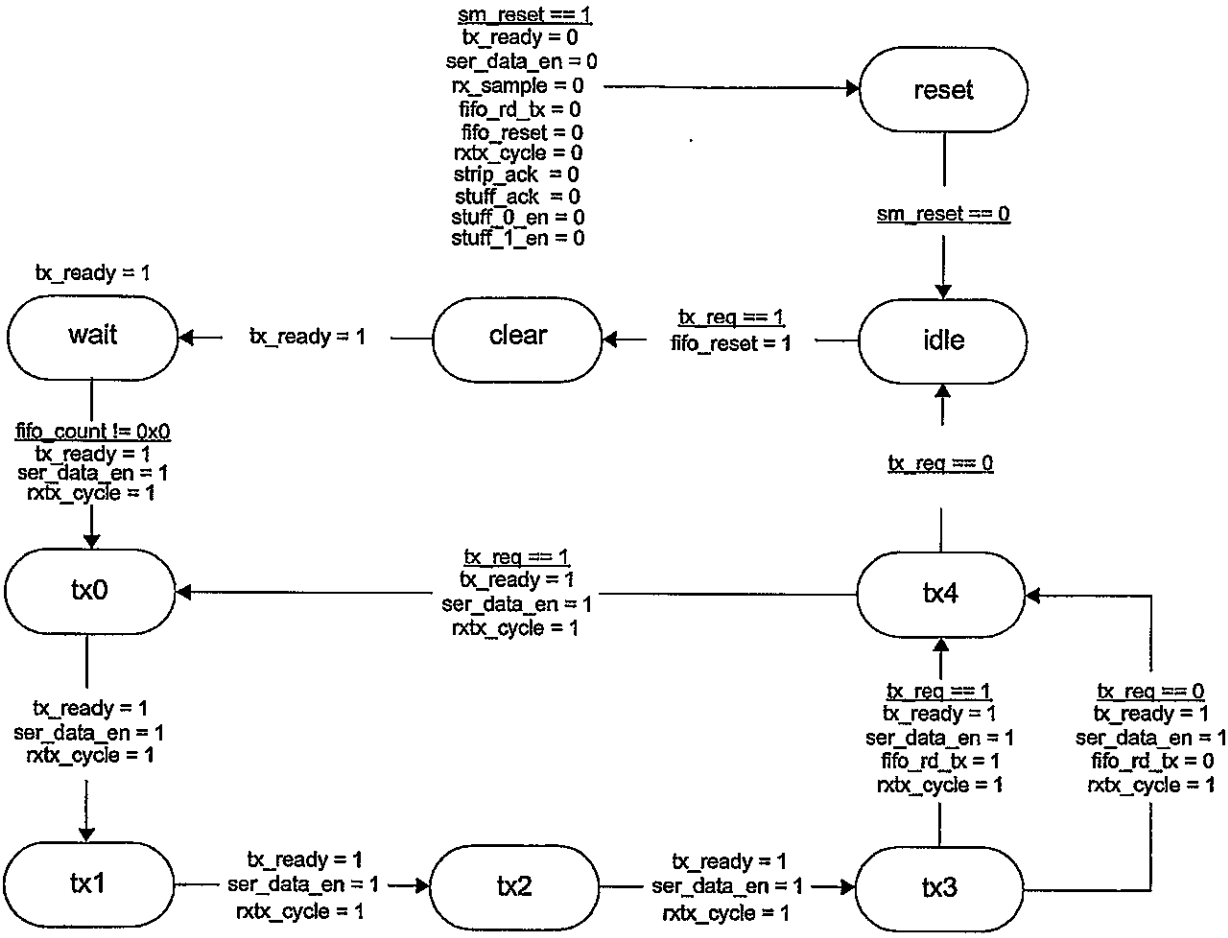

sm_reset == 1
tx_ready = 0
ser_data_en = 0
rx_sample = 0
fifo_rd_tx = 0
fifo_reset = 0
rxtx_cycle = 0
strip_ack = 0
stuff_ack = 0
stuff_0_en = 0
stuff_1_en = 0
reset
sm_reset == 0
idle
tx_req == 1
fifo_reset = 1
clear
tx_ready = 1
wait
tx_ready = 1
fifo_count != 0x0
tx_ready = 1
ser_data_en = 1
rxtx_cycle = 1
tx_req == 0
tx4
tx_req == 1
tx_ready = 1
ser_data_en = 1
rxtx_cycle = 1
tx0
tx_ready = 1
ser_data_en = 1
rxtx_cycle = 1
tx_req == 1
tx_ready = 1
ser_data_en = 1
fifo_rd_tx = 1
rxtx_cycle = 1
tx_req == 0
tx_ready = 1
ser_data_en = 1
fifo_rd_tx = 0
rxtx_cycle = 1
tx1
tx_ready = 1
ser_data_en = 1
rxtx_cycle = 1
tx2
tx_ready = 1
ser_data_en = 1
rxtx_cycle = 1
tx3

FIG. 42

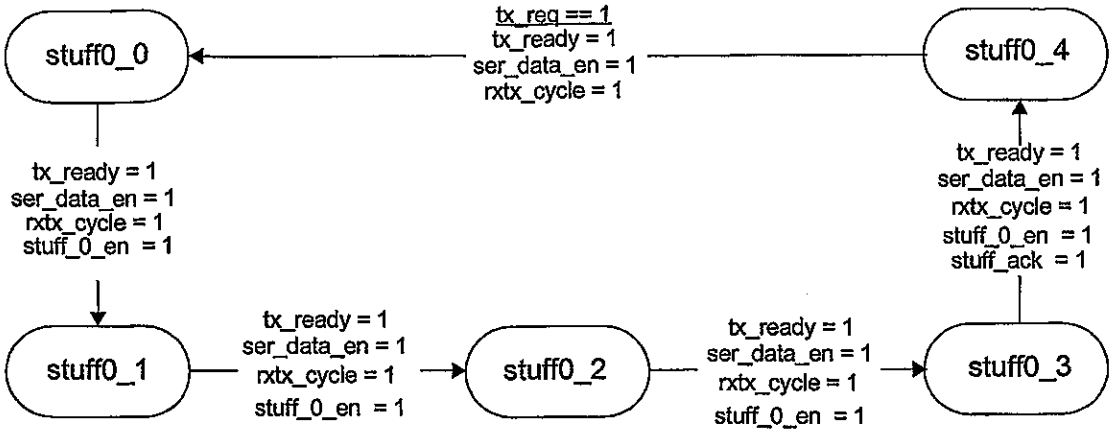

stuff0_0
tx_req == 1
tx_ready = 1
ser_data_en = 1
rxtx_cycle = 1
stuff0_4
tx_ready = 1
ser_data_en = 1
rxtx_cycle = 1
stuff_0_en = 1
tx_ready = 1
ser_data_en = 1
rxtx_cycle = 1
stuff_0_en = 1
stuff_ack = 1
stuff0_1
tx_ready = 1
ser_data_en = 1
rxtx_cycle = 1
stuff_0_en = 1
stuff0_2
tx_ready = 1
ser_data_en = 1
rxtx_cycle = 1
stuff_0_en = 1
stuff0_3

FIG. 43

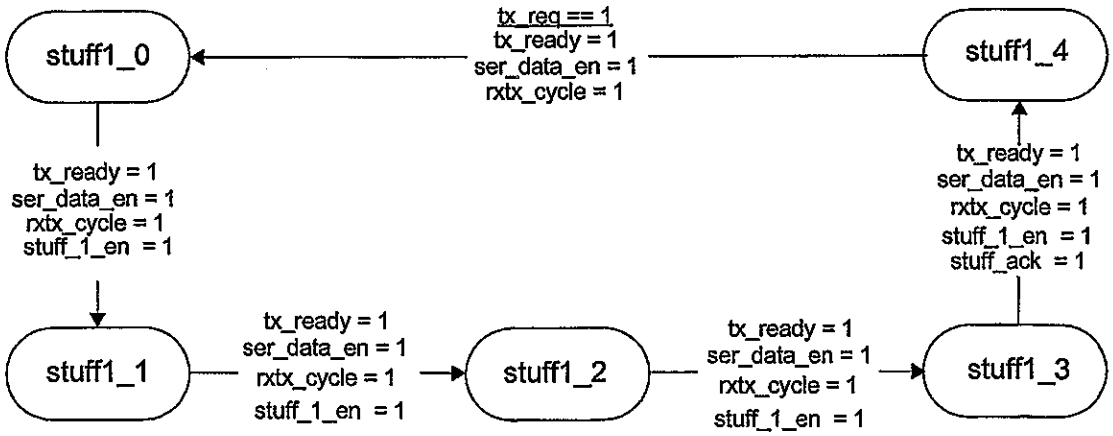

stuff1_0
tx_req == 1
tx_ready = 1
ser_data_en = 1
rxtx_cycle = 1
stuff1_4
tx_ready = 1
ser_data_en = 1
rxtx_cycle = 1
stuff_1_en = 1
tx_ready = 1
ser_data_en = 1
rxtx_cycle = 1
stuff_1_en = 1
stuff_ack = 1
stuff1_1
tx_ready = 1
ser_data_en = 1
rxtx_cycle = 1
stuff_1_en = 1
stuff1_2
tx_ready = 1
ser_data_en = 1
rxtx_cycle = 1
stuff_1_en = 1
stuff1_3

FIG. 44

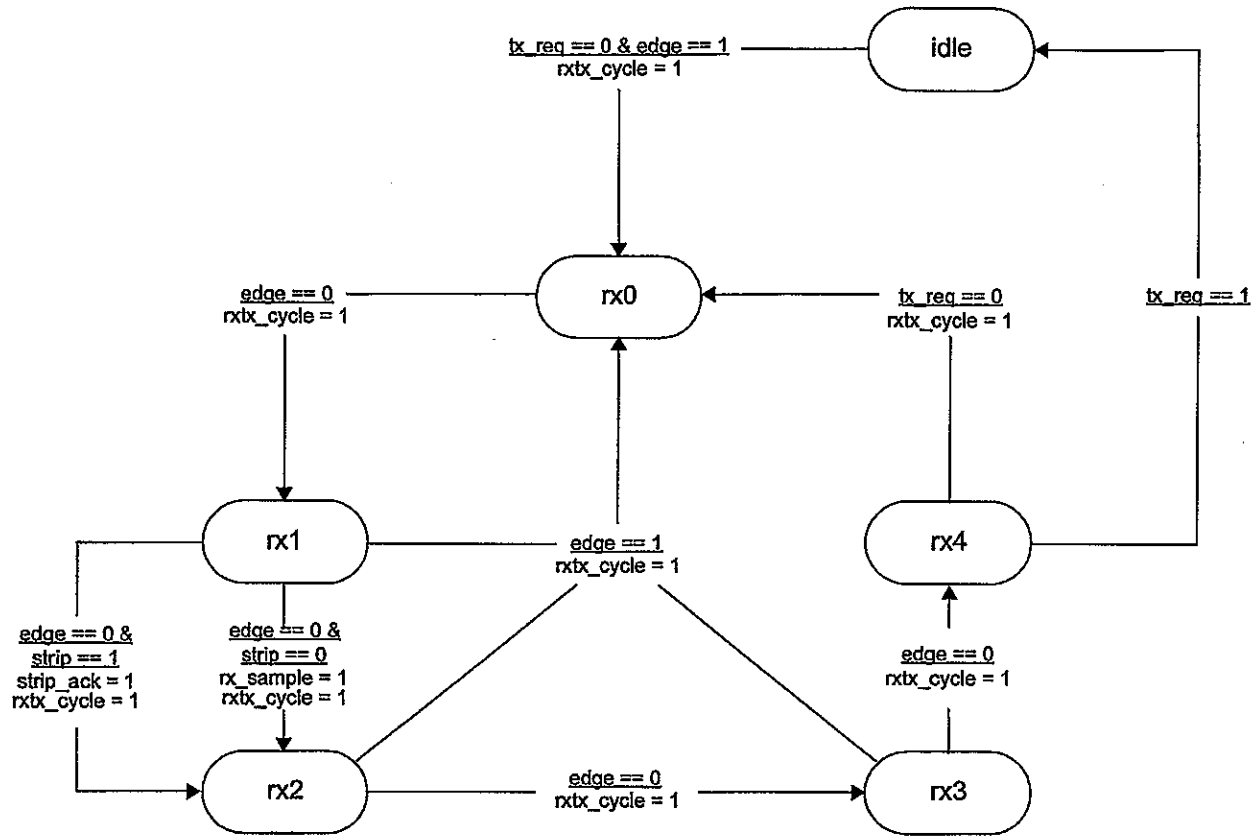

idle
tx_req == 0 & edge == 1
rxtx_cycle = 1
rx0
edge == 0
rxtx_cycle = 1
tx_req == 0
rxtx_cycle = 1
tx_req == 1
rx1
edge == 1
rxtx_cycle = 1
rx4
edge == 0 &
strip == 1
strip_ack = 1
rxtx_cycle = 1
edge == 0 &
strip == 0
rx_sample = 1
rxtx_cycle = 1
edge == 0
rxtx_cycle = 1
rx2
edge == 0
rxtx_cycle = 1
rx3

FIG. 45

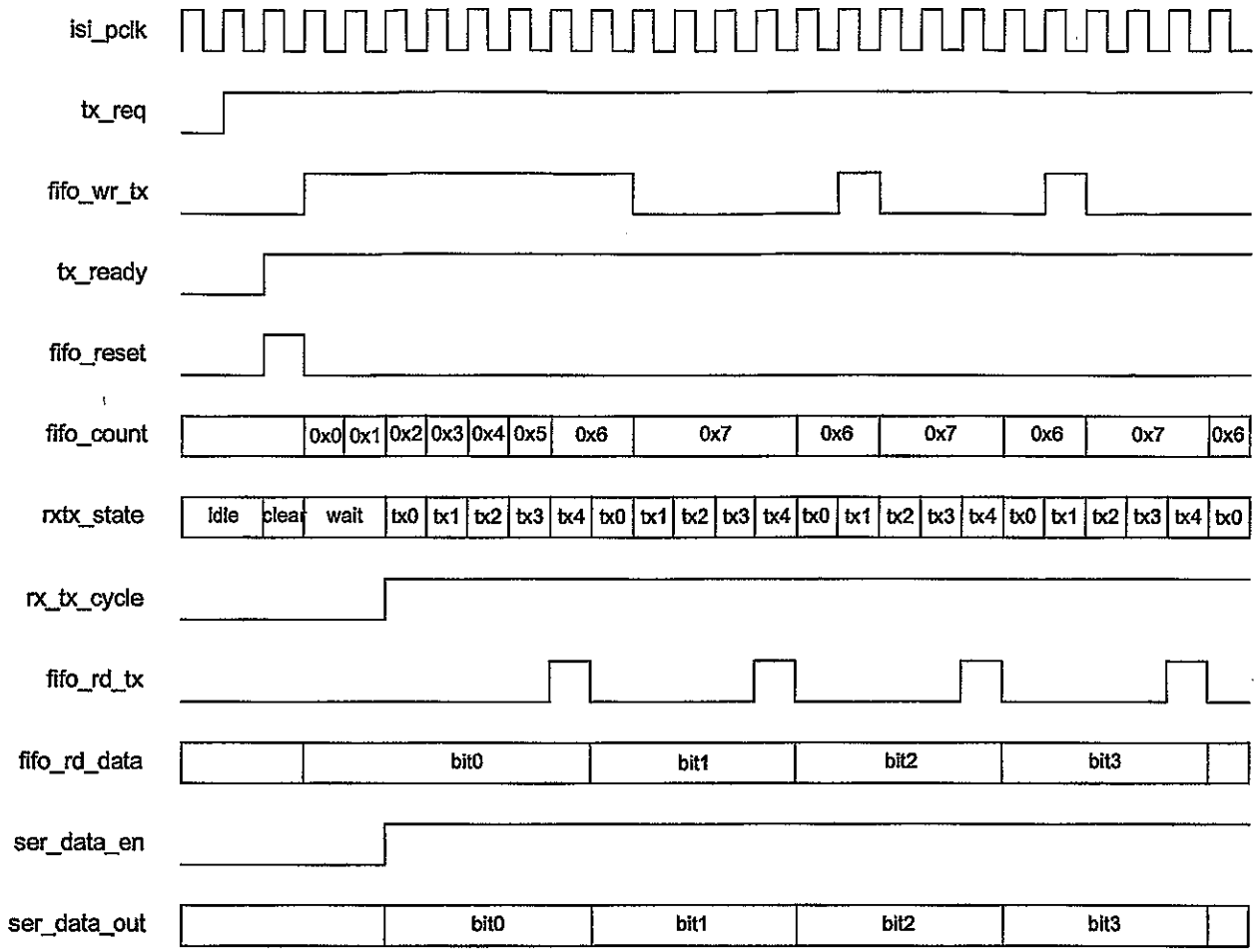

isi_pclk
tx_req
fifo_wr_tx
tx_ready
fifo_reset
fifo_count
0x0 0x1 0x2 0x3 0x4 0x5 0x6 0x7 0x6 0x7 0x6 0x7 0x6
rxtx_state
Idle clear wait tx0 tx1 tx2 tx3 tx4 tx0 tx1 tx2 tx3 tx4 tx0 tx1 tx2 tx3 tx4 tx0 tx1 tx2 tx3 tx4 tx0
rx_tx_cycle
fifo_rd_tx
fifo_rd_data
bit0 bit1 bit2 bit3
ser_data_en
ser_data_out
bit0 bit1 bit2 bit3

FIG. 46

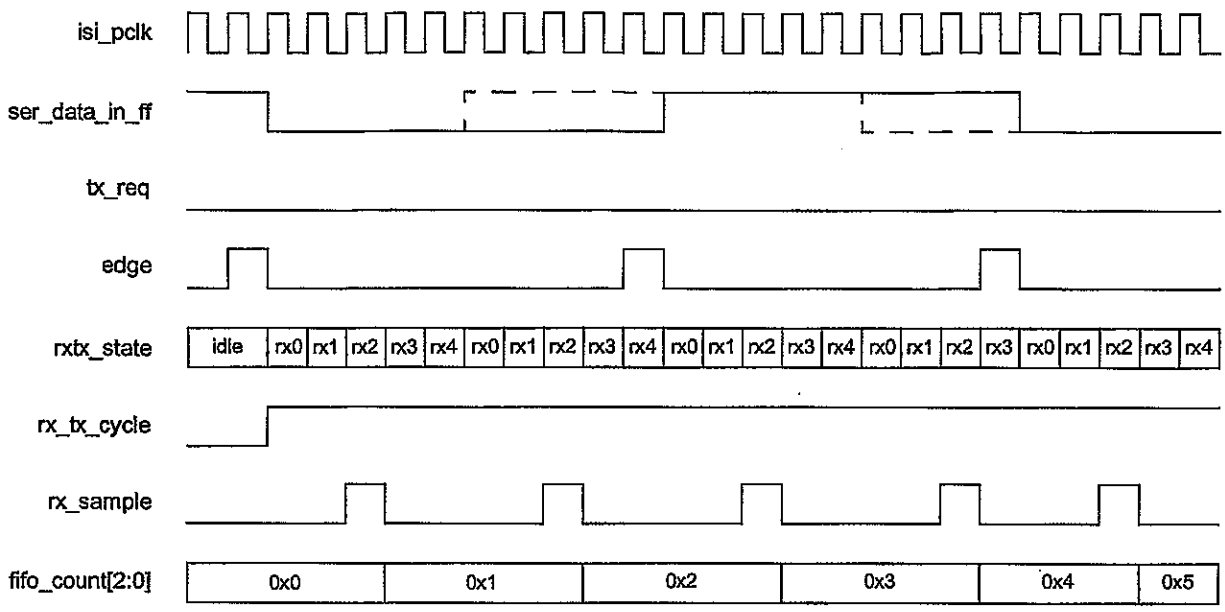

isi_pclk
ser_data_in_ff
tx_req
edge
rxtx_state
idle
rx0
rx1
rx2
rx3
rx4
rx_tx_cycle
rx_sample
fifo_count[2:0]
0x0
0x1
0x2
0x3
0x4
0x5

FIG. 47

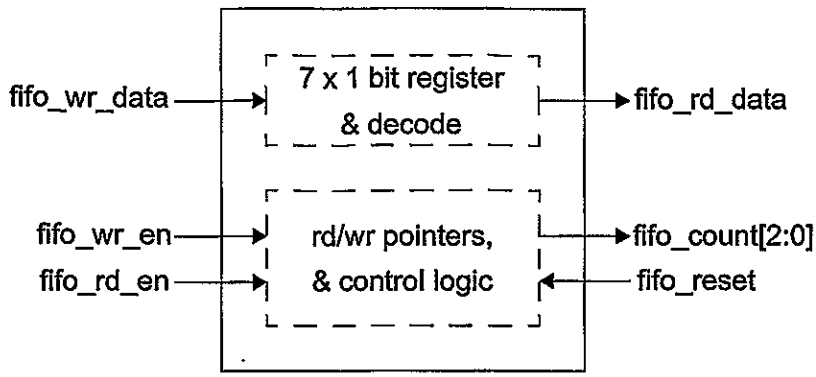

fifo_wr_data
7 x 1 bit register
& decode
fifo_rd_data
fifo_wr_en
fifo_rd_en
rd/wr pointers,
& control logic
fifo_count[2:0]
fifo_reset

FIG. 48

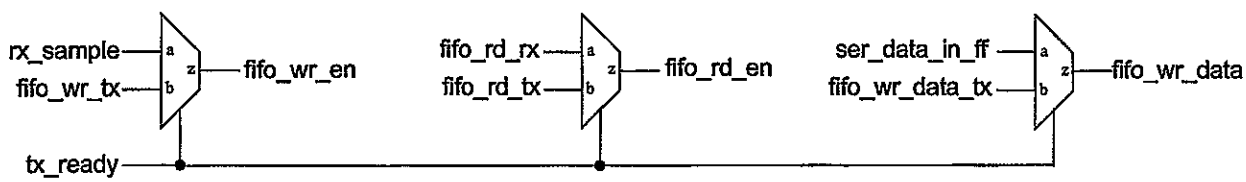

rx_sample
fifo_wr_tx
a
b
z
fifo_wr_en
tx_ready
fifo_rd_rx
fifo_rd_tx
fifo_rd_en
ser_data_in_ff
fifo_wr_data_tx
fifo_wr_data

FIG. 49

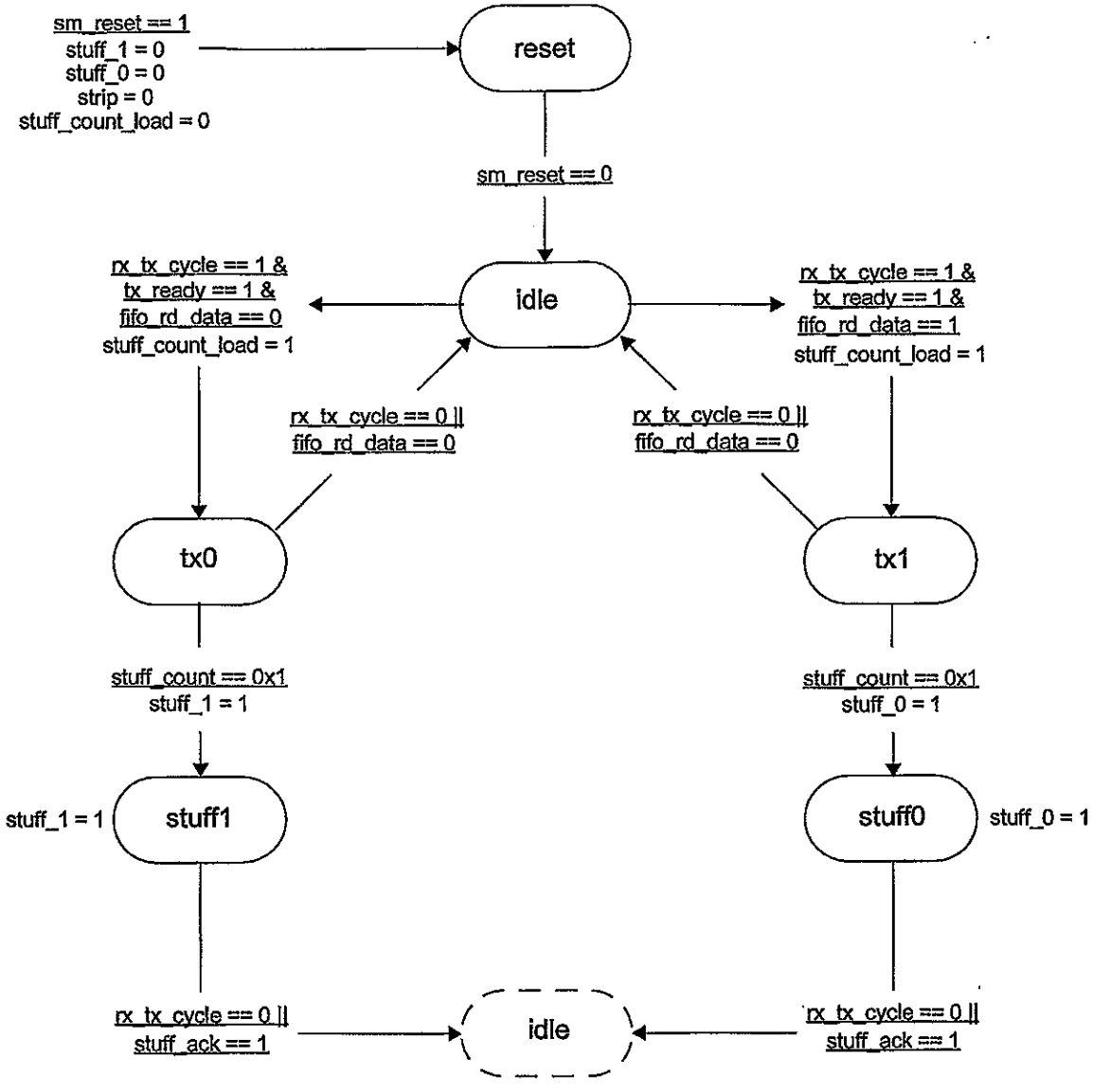

sm_reset == 1
stuff_1 = 0
stuff_0 = 0
strip = 0
stuff_count_load = 0
reset
sm_reset == 0
idle
rx_tx_cycle == 1 &
tx_ready == 1 &
fifo_rd_data == 0
stuff_count_load = 1
rx_tx_cycle == 1 &
tx_ready == 1 &
fifo_rd_data == 1
stuff_count_load = 1
rx_tx_cycle == 0 ||
fifo_rd_data == 0
rx_tx_cycle == 0 ||
fifo_rd_data == 0
tx0
tx1
stuff_count == 0x1
stuff_1 = 1
stuff_count == 0x1
stuff_0 = 1
stuff_1 = 1
stuff1
stuff0
stuff_0 = 1
rx_tx_cycle == 0 ||
stuff_ack == 1
idle
rx_tx_cycle == 0 ||
stuff_ack == 1

FIG. 50

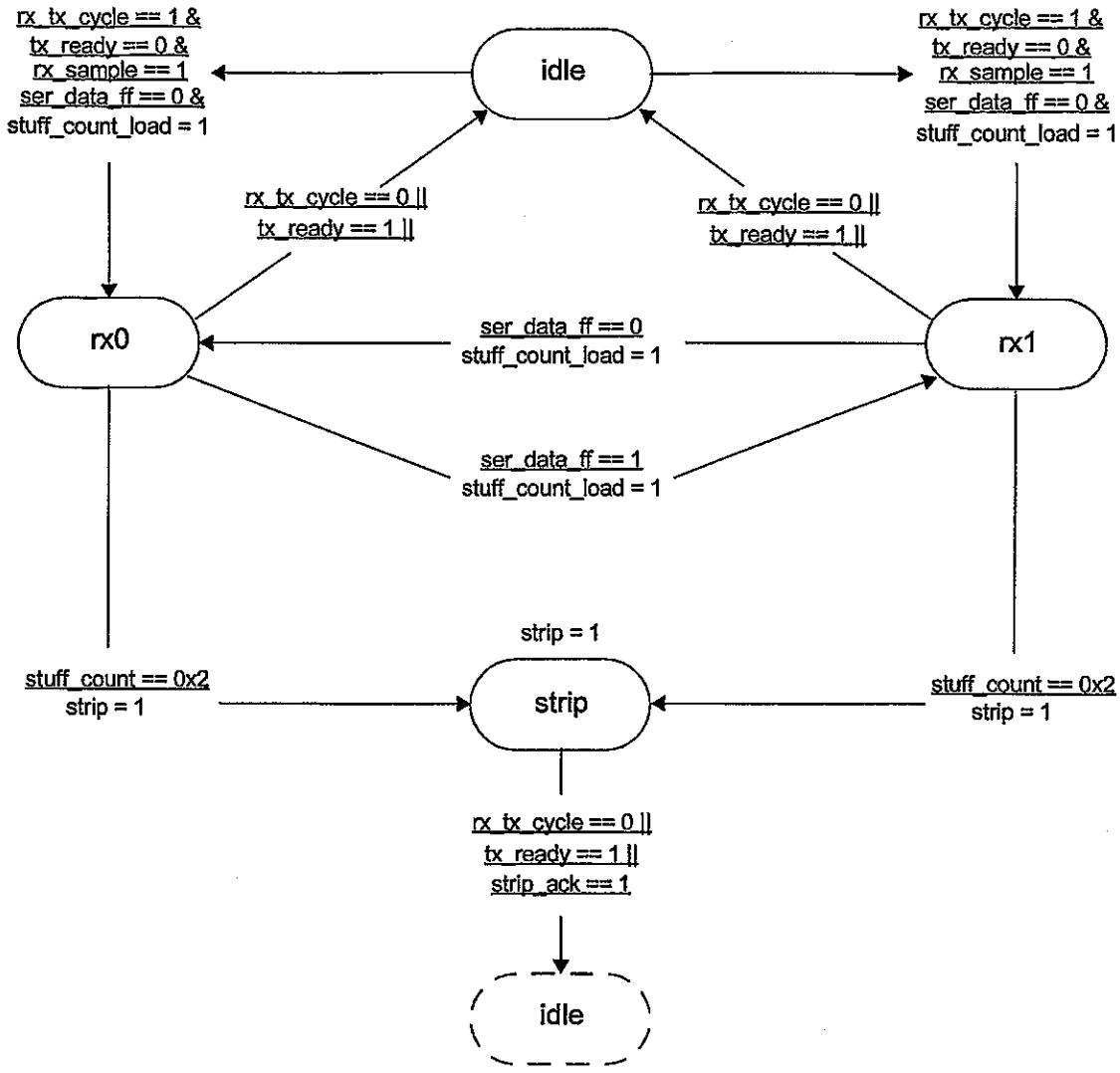

idle
rx_tx_cycle == 1 &
tx_ready == 0 &
rx_sample == 1
ser_data_ff == 0 &
stuff_count_load = 1
rx_tx_cycle == 1 &
tx_ready == 0 &
rx_sample == 1
ser_data_ff == 0 &
stuff_count_load = 1
rx_tx_cycle == 0 ||
tx_ready == 1 ||
rx_tx_cycle == 0 ||
tx_ready == 1 ||
rx0
rx1
ser_data_ff == 0
stuff_count_load = 1
ser_data_ff == 1
stuff_count_load = 1
strip = 1
stuff_count == 0x2
strip = 1
strip
stuff_count == 0x2
strip = 1
rx_tx_cycle == 0 ||
tx_ready == 1 ||
strip_ack == 1
idle

FIG. 51

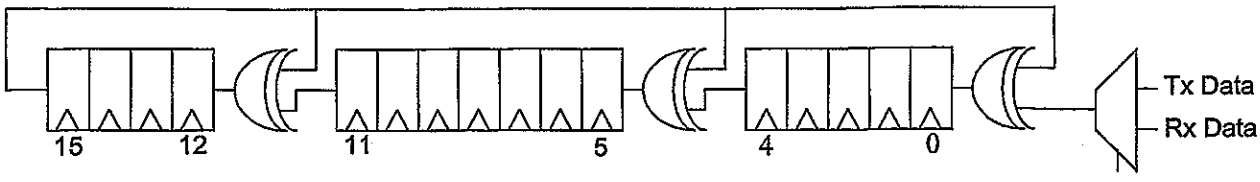

15
12
11
5
4
0
Tx Data
Rx Data

FIG. 52

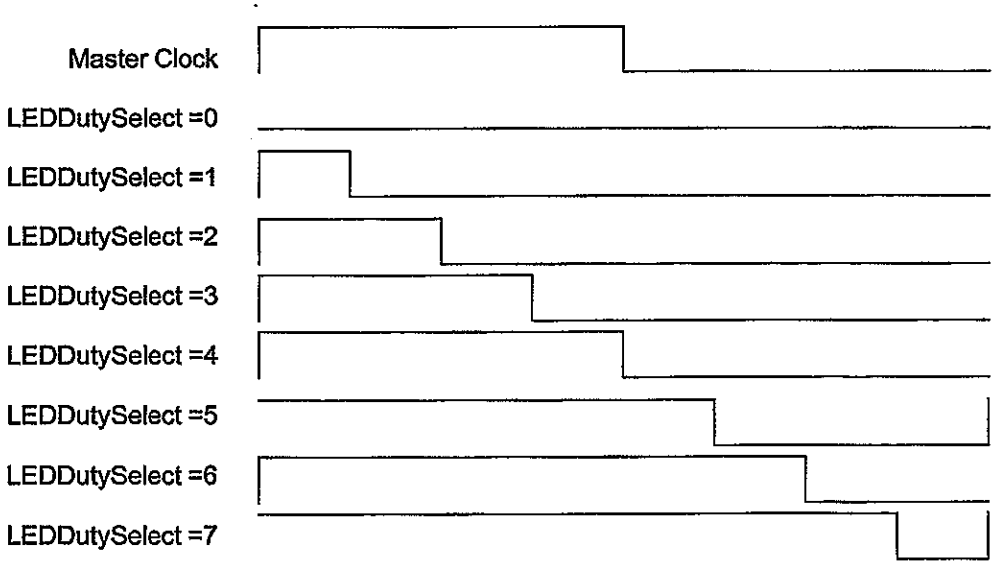

Master Clock
LEDDutySelect =0
LEDDutySelect =1
LEDDutySelect =2
LEDDutySelect =3
LEDDutySelect =4
LEDDutySelect =5
LEDDutySelect =6
LEDDutySelect =7

FIG. 54

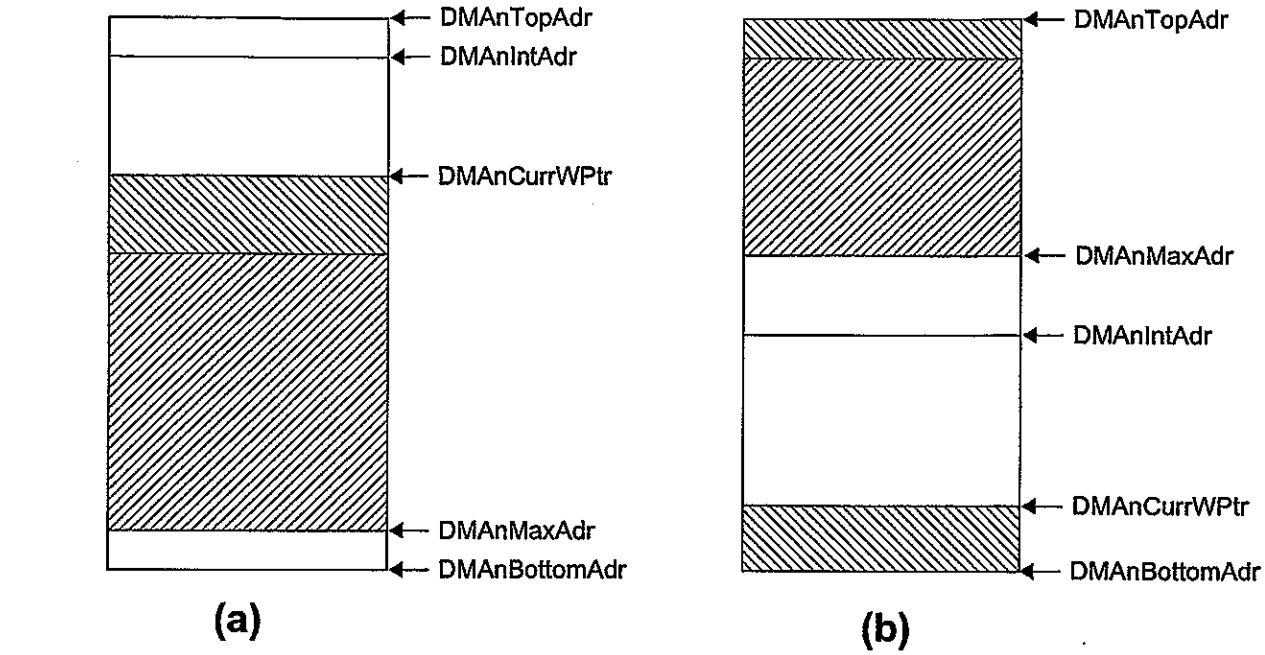

DMAnTopAdr
DMAnIntAdr
DMAnCurrWPtr
DMAnMaxAdr
DMAnBottomAdr
(a)
DMAnTopAdr
DMAnMaxAdr
DMAnIntAdr
DMAnCurrWPtr
DMAnBottomAdr
(b)
Key:
Free buffer space
Filled buffer space (unprocessed data)
Buffer space filled since last write to the DMAnIntAdr/DMAnMaxAdr registers

FIG. 53

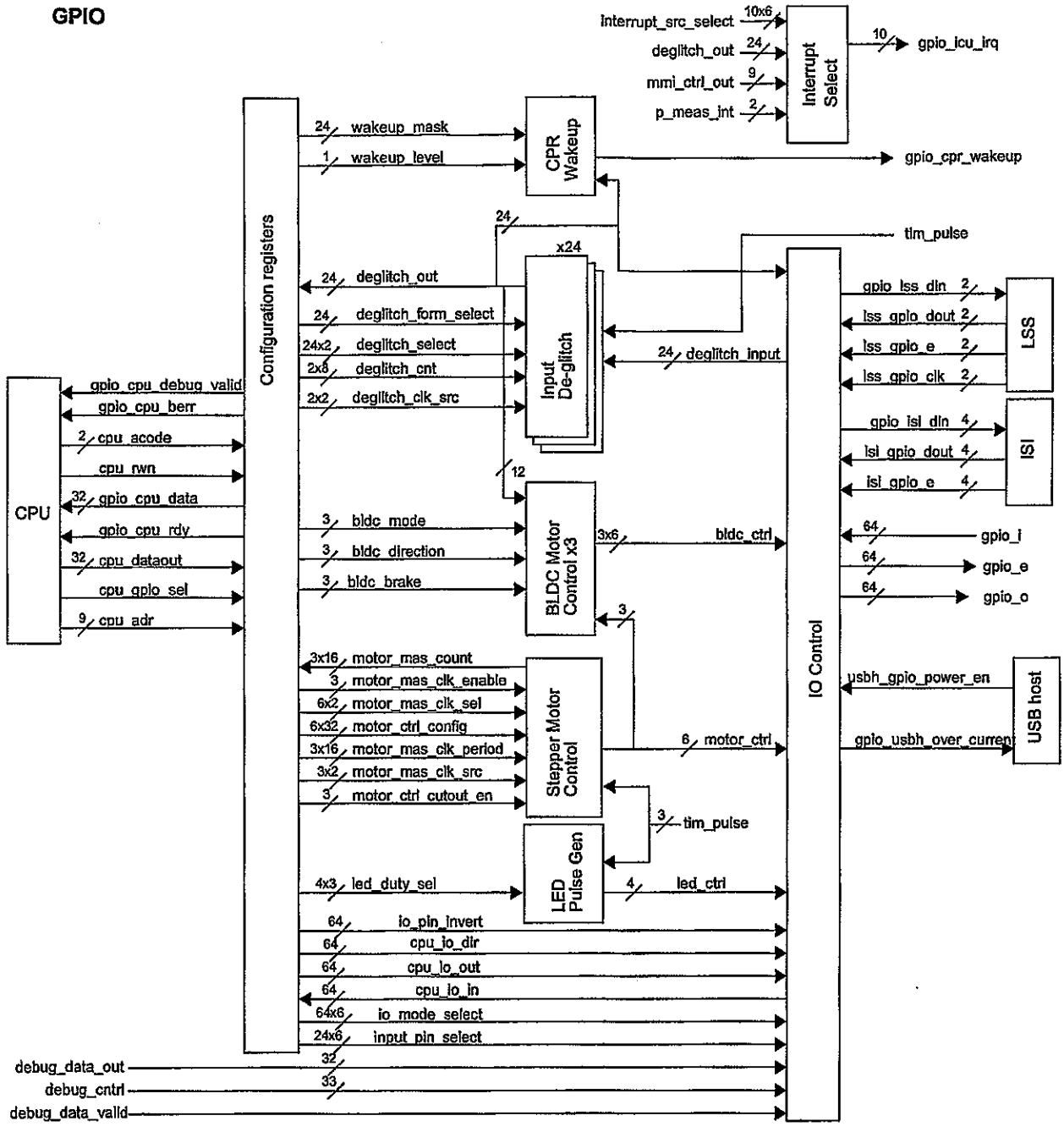

GPIO
interrupt_src_select
deglitch_out
mmi_ctrl_out
p_meas_int
Interrupt Select
gpio_icu_irq
CPR Wakeup
wakeup_mask
wakeup_level
gpio_cpr_wakeup
tim_pulse
Configuration registers
deglitch_out
deglitch_form_select
deglitch_select
deglitch_cnt
deglitch_clk_src
Input De-glitch
deglitch_input
gpio_lss_din
lss_gpio_dout
lss_gpio_e
lss_gpio_clk
LSS
gpio_isi_din
isi_gpio_dout
isi_gpio_e
ISI
CPU
gpio_cpu_debug_valid
gpio_cpu_berr
cpu_acode
cpu_rwn
gpio_cpu_data
gpio_cpu_rdy
cpu_dataout
cpu_gpio_sel
cpu_adr
bldc_mode
bldc_direction
bldc_brake
BLDC Motor Control x3
bldc_ctrl
gpio_i
gpio_e
gpio_o
IO Control
motor_mas_count
motor_mas_clk_enable
motor_mas_clk_sel
motor_ctrl_config
motor_mas_clk_period
motor_mas_clk_src
motor_ctrl_cutout_en
Stepper Motor Control
motor_ctrl
usbh_gpio_power_en
gpio_usbh_over_current
USB host
tim_pulse
LED Pulse Gen
led_duty_sel
led_ctrl
io_pin_invert
cpu_io_dir
cpu_io_out
cpu_io_in
io_mode_select
input_pin_select
debug_data_out
debug_cntrl
debug_data_valid

FIG. 55

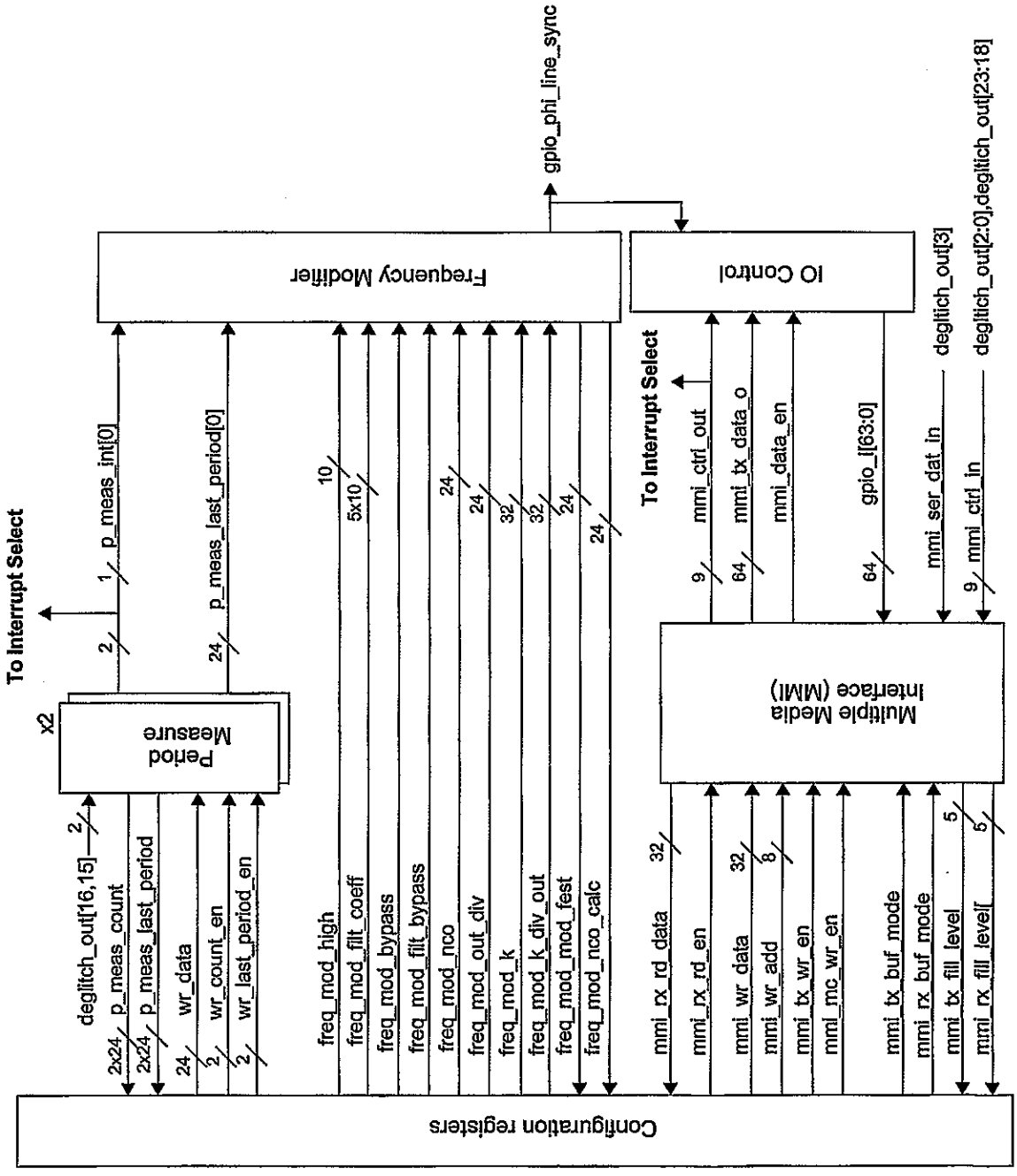

Configuration registers
deglitch_out[16,15]
2x24 p_meas_count
2x24 p_meas_last_period
24 wr_data
2 wr_count_en
2 wr_last_period_en
x2
Period Measure
2
1 p_meas_int[0]
24 p_meas_last_period[0]
To Interrupt Select
Frequency Modifier
gpio_phi_line_sync
freq_mod_high 10
freq_mod_filt_coeff 5x10
freq_mod_bypass
freq_mod_filt_bypass
freq_mod_nco 24
freq_mod_out_div 24
freq_mod_k 32
freq_mod_k_div_out 32
freq_mod_mod_fest 24
freq_mod_nco_calc 24
IO Control
To Interrupt Select
mmi_ctrl_out 9
mmi_tx_data_o 64
mmi_data_en
gpio_i[63:0] 64
mmi_ser_dat_in
deglitch_out[3]
mmi_ctrl_in 9
deglitch_out[2:0],deglitch_out[23:18]
Multiple Media Interface (MMI)
mmi_rx_rd_data 32
mmi_rx_rd_en
mmi_wr_data 32
mmi_wr_add 8
mmi_tx_wr_en
mmi_mc_wr_en
mmi_tx_buf_mode
mmi_rx_buf_mode
mmi_tx_fill_level 5
mmi_rx_fill_level 5

FIG. 56

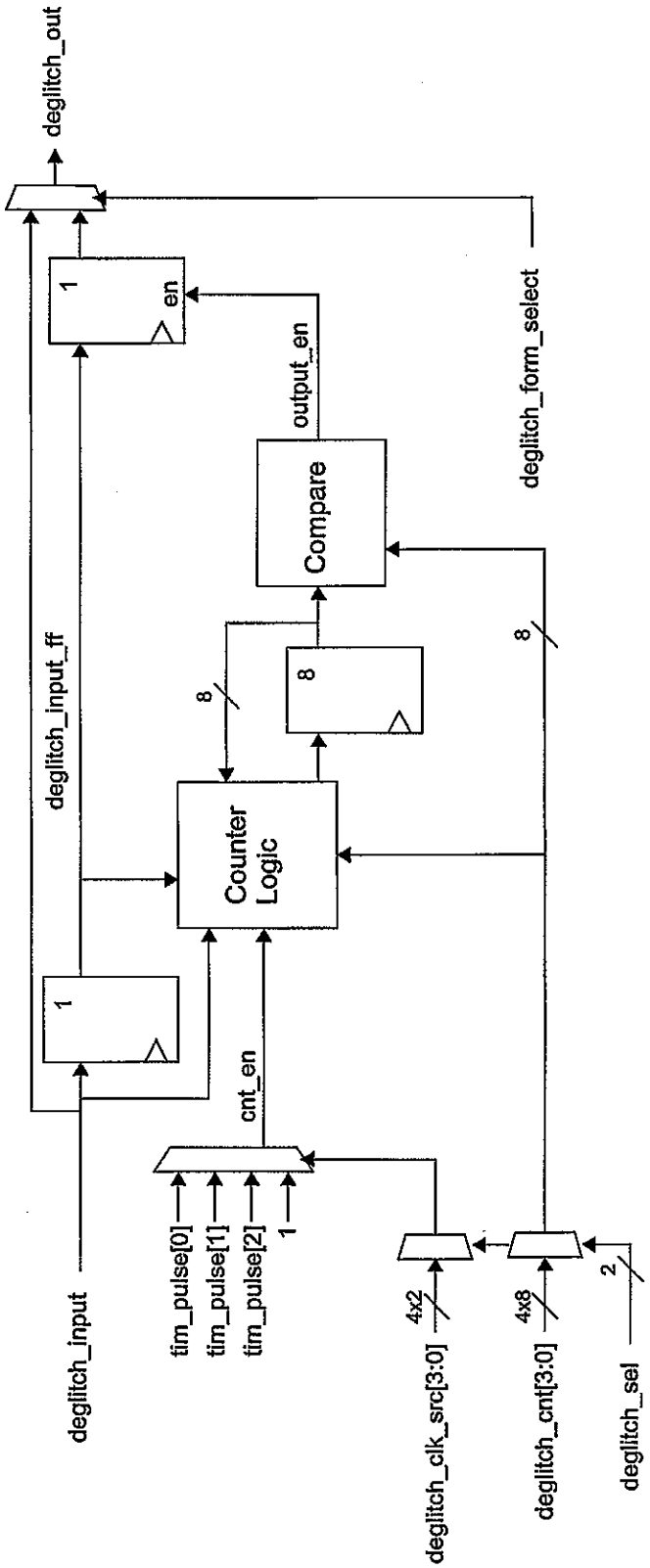

deglitch_out
1
en
output_en
Compare
deglitch_form_select
deglitch_input_ff
8
8
8
Counter Logic
1
cnt_en
deglitch_input
tim_pulse[0]
tim_pulse[1]
tim_pulse[2]
1
4x2
4x8
2
deglitch_clk_src[3:0]
deglitch_cnt[3:0]
deglitch_sel

FIG. 57

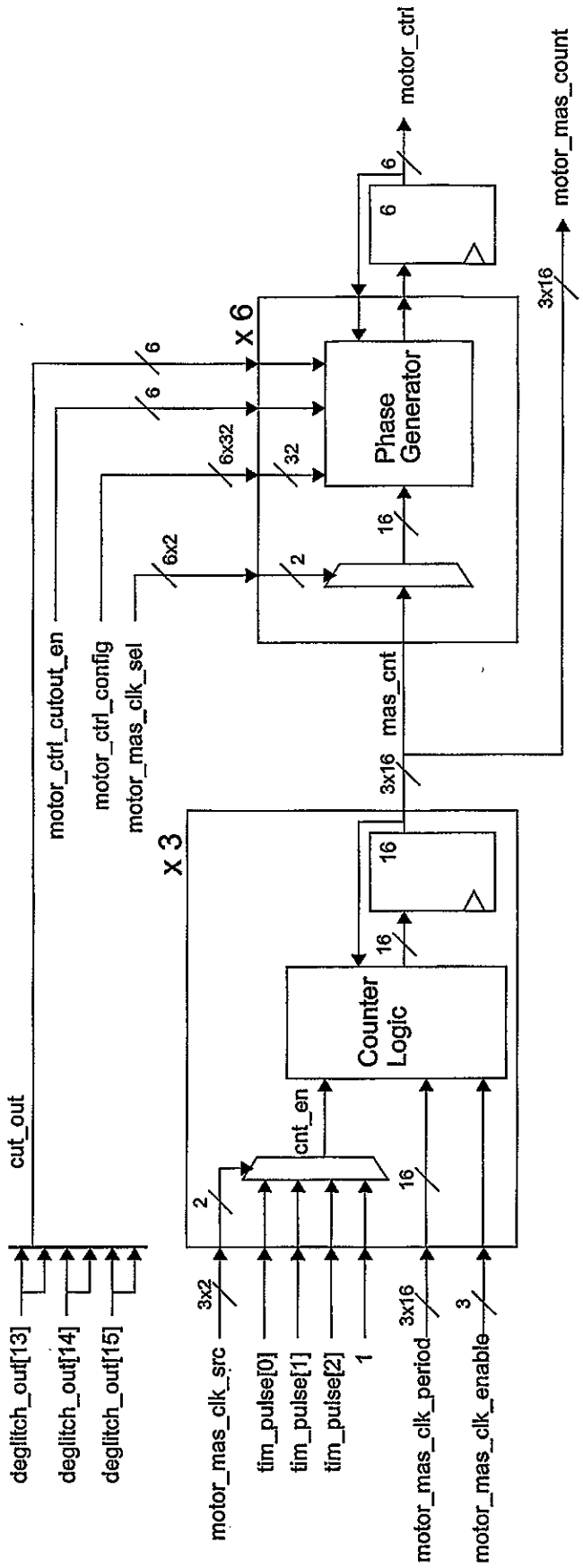

deglitch_out[13]
deglitch_out[14]
deglitch_out[15]
cut_out
motor_ctrl_cutout_en
motor_ctrl_config
motor_mas_clk_sel
x 3
x 6
motor_mas_clk_src
tim_pulse[0]
tim_pulse[1]
tim_pulse[2]
1
motor_mas_clk_period
motor_mas_clk_enable
cnt_en
Counter Logic
Phase Generator
mas_cnt
motor_ctrl
motor_mas_count
3x2
2
3x16
3
16
6
6x32
32
6x2
motor_ctrl_cutout_en
motor_ctrl_config
motor_mas_clk_sel

FIG. 58

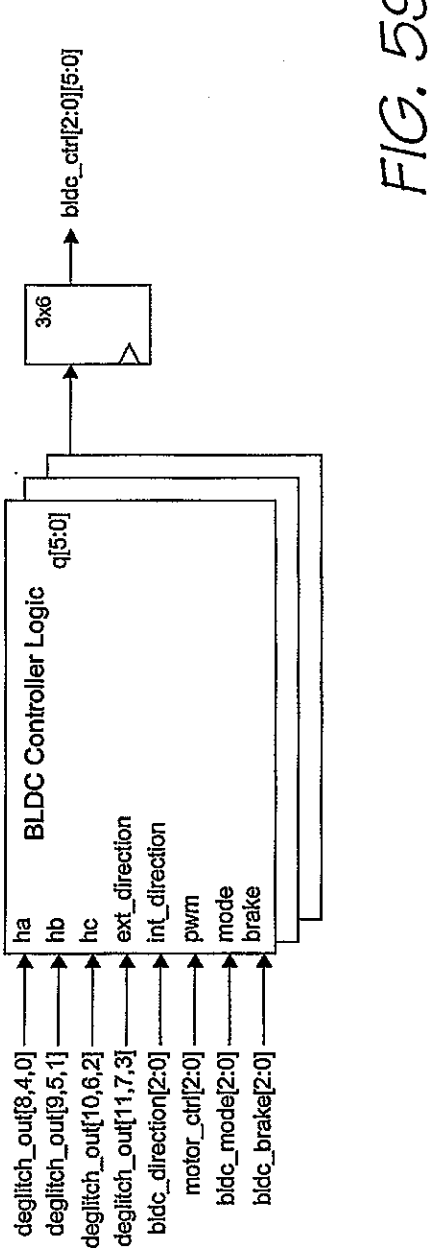

Controller x 3
BLDC Controller Logic
ha
hb
hc
ext_direction
int_direction
pwm
mode
brake
q[5:0]
3x6
bldc_ctrl[2:0][5:0]
deglitch_out[8,4,0]
deglitch_out[9,5,1]
deglitch_out[10,6,2]
deglitch_out[11,7,3]
bldc_direction[2:0]
motor_ctrl[2:0]
bldc_mode[2:0]
bldc_brake[2:0]

FIG. 59

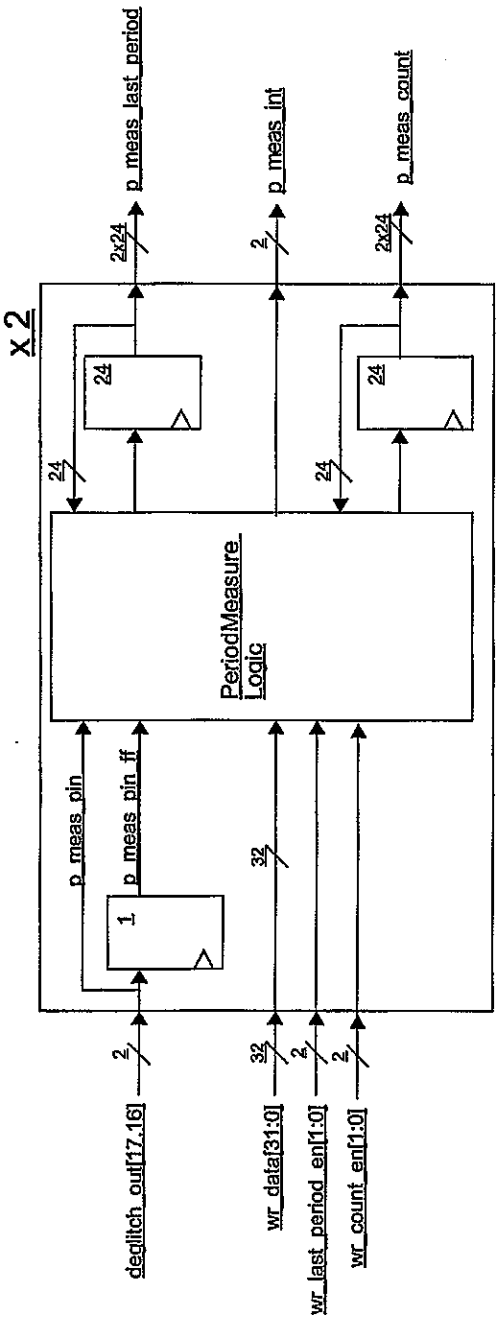

x 2
PeriodMeasure Logic
p_meas_pin
p_meas_pin_ff
p_meas_last_period
p_meas_int
p_meas_count
deglitch_out[17,16]
wr_data[31:0]
wr_last_period_en[1:0]
wr_count_en[1:0]

FIG. 60

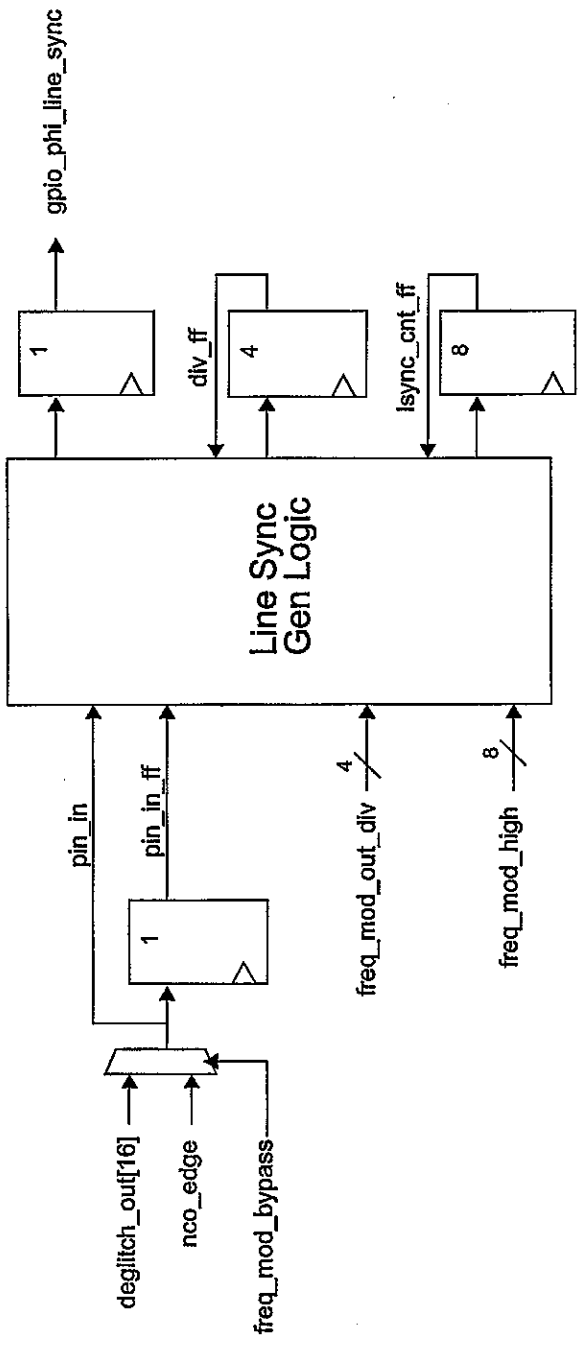

gpio_phi_line_sync
1
div_ff
4
lsync_cnt_ff
8
Line Sync
Gen Logic
pin_in
pin_in_ff
1
4
freq_mod_out_div
8
freq_mod_high
deglitch_out[16]
nco_edge
freq_mod_bypass

FIG. 61

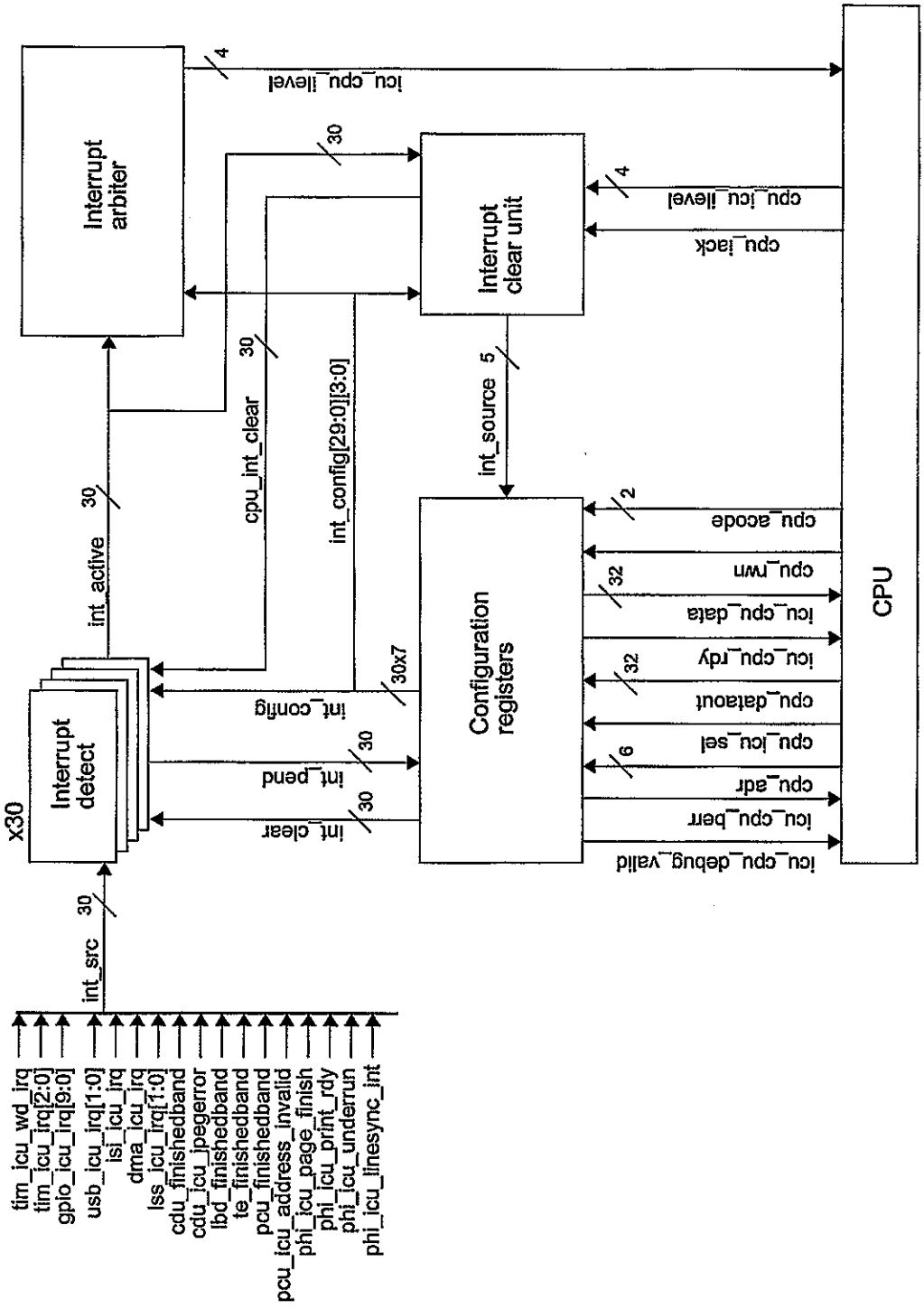

tim_icu_wd_irq
tim_icu_irq[2:0]
gpio_icu_irq[9:0]
usb_icu_irq[1:0]
isi_icu_irq
dma_icu_irq
lss_icu_irq[1:0]
cdu_finishedband
cdu_icu_jpegerror
lbd_finishedband
te_finishedband
pcu_finishedband
pcu_icu_address_invalid
phi_icu_page_finish
phi_icu_print_rdy
phi_icu_underrun
phi_icu_linesync_int
int_src
30
x30
Interrupt detect
int_active
30
int_config
30x7
int_pend
30
int_clear
30
cpu_int_clear
30
int_config[29:0][3:0]
Interrupt arbiter
4
icu_cpu_ilevel
30
Interrupt clear unit
4
cpu_ilevel
cpu_iack
int_source 5
Configuration registers
2
cpu_acode
cpu_rwn
32
icu_cpu_data
32
icu_cpu_rdy
cpu_datаout
6
cpu_icu_sel
cpu_adr
icu_cpu_berr
icu_cpu_debug_valid
CPU

FIG. 62

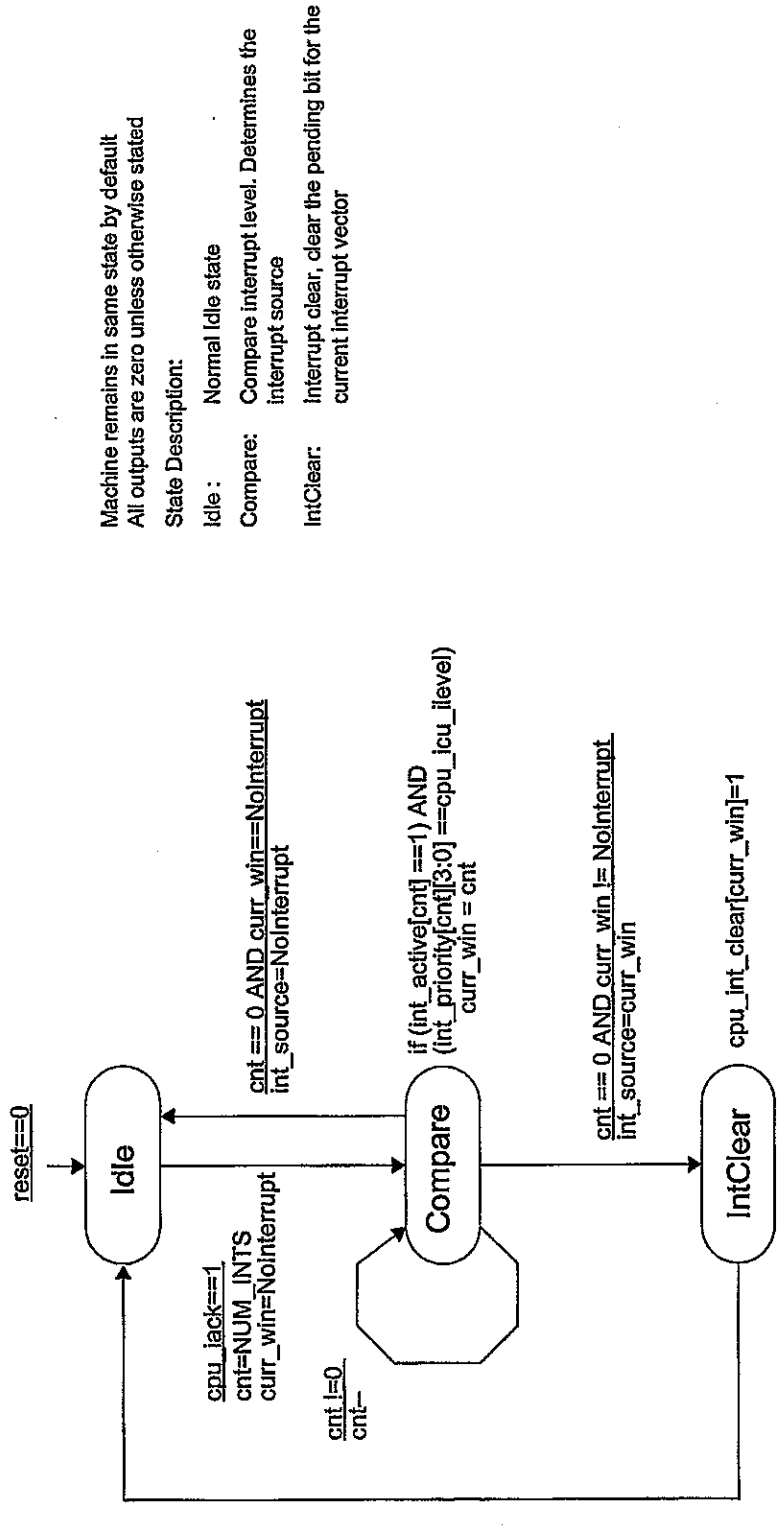

reset==0
Idle
cpu_iack==1
cnt=NUM_INTS
curr_win=NoInterrupt
cnt != 0
cnt--
cnt == 0 AND curr_win==NoInterrupt
int_source=NoInterrupt
Compare
if (int_active[cnt] ==1) AND
(int_priority[cnt][3:0] ==cpu_icu_ilevel)
curr_win = cnt
cnt == 0 AND curr_win != NoInterrupt
int_source=curr_win
IntClear
cpu_int_clear[curr_win]=1
Machine remains in same state by default
All outputs are zero unless otherwise stated
State Description:
Idle : Normal Idle state
Compare: Compare interrupt level. Determines the interrupt source
IntClear: Interrupt clear, clear the pending bit for the current interrupt vector

FIG. 63

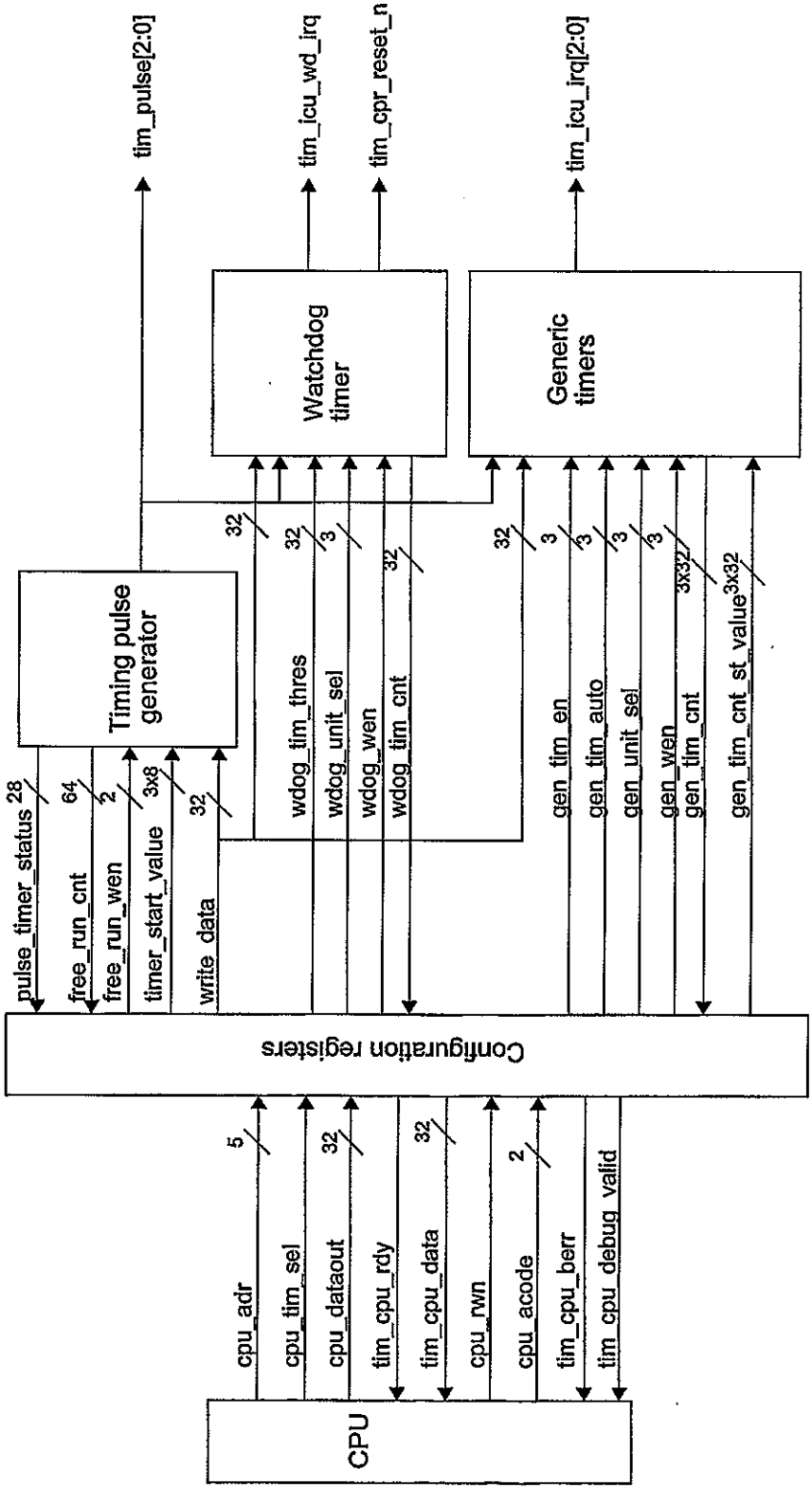

CPU
cpu_adr
cpu_tim_sel
cpu_dataout
tim_cpu_rdy
tim_cpu_data
cpu_rwn
cpu_acode
tim_cpu_berr
tim_cpu_debug_valid
5
32
32
2
Configuration registers
pulse_timer_status
free_run_cnt
free_run_wen
timer_start_value
write_data
28
64
2
3x8
32
Timing pulse generator
tim_pulse[2:0]
Watchdog timer
wdog_tim_thres
wdog_unit_sel
wdog_wen
wdog_tim_cnt
32
3
32
tim_icu_wd_irq
tim_cpr_reset_n
Generic timers
gen_tim_en
gen_tim_auto
gen_unit_sel
gen_wen
gen_tim_cnt
gen_tim_cnt_st_value
32
3
3
3
3
3x32
3x32
tim_icu_irq[2:0]

FIG. 63A

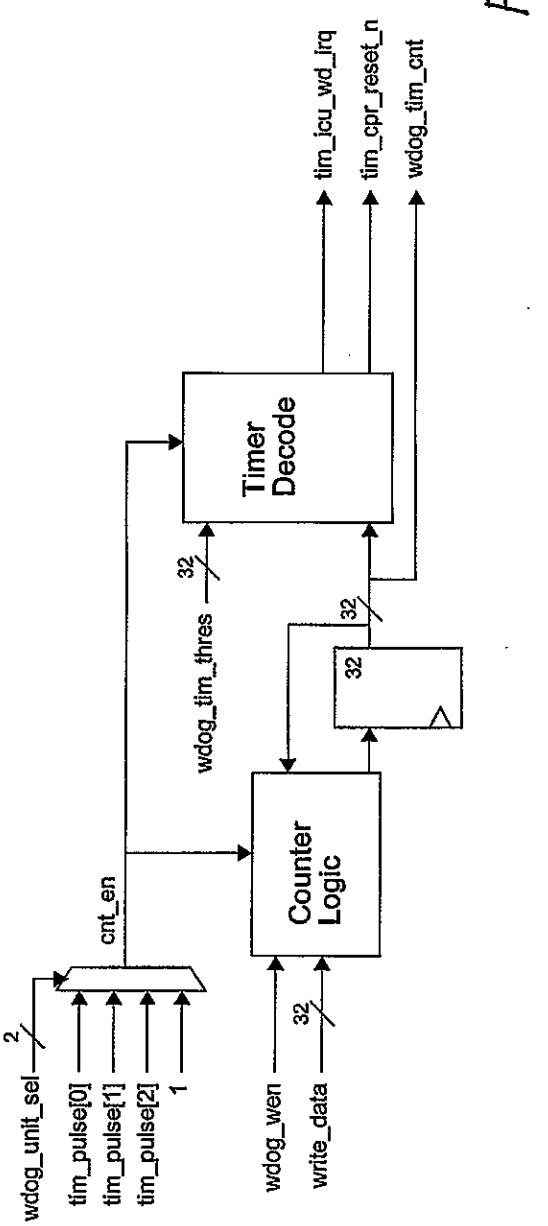

wdog_unit_sel
2
tim_pulse[0]
tim_pulse[1]
tim_pulse[2]
1
cnt_en
wdog_wen
write_data
32
Counter Logic
wdog_tim_thres
32
32
Timer Decode
tim_icu_wd_irq
tim_cpr_reset_n
wdog_tim_cnt

FIG. 64

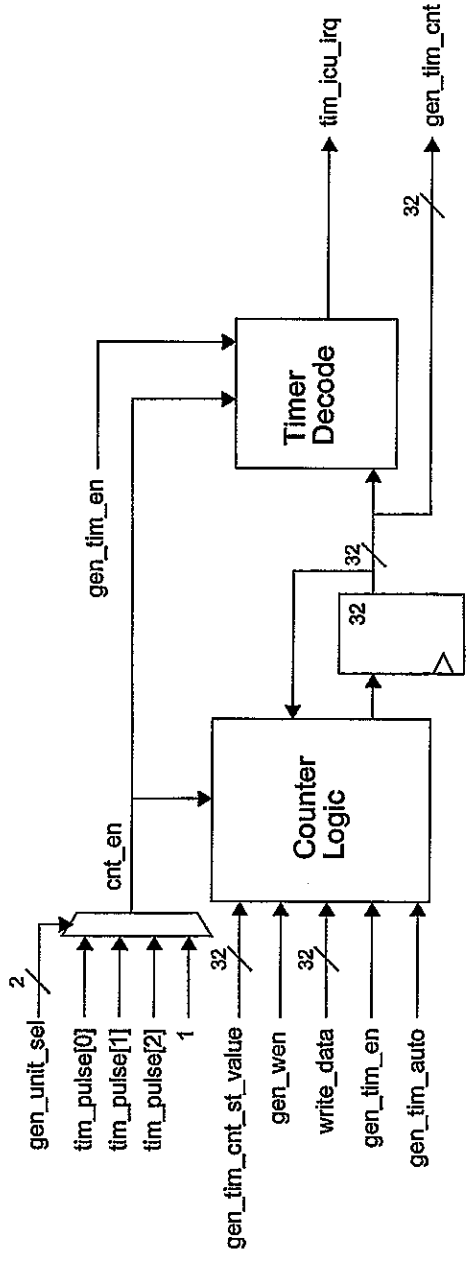

gen_unit_sel
2
tim_pulse[0]
tim_pulse[1]
tim_pulse[2]
1
cnt_en
gen_tim_en
gen_tim_cnt_st_value
32
gen_wen
write_data
gen_tim_en
gen_tim_auto
Counter Logic
32
32
Timer Decode
tim_icu_irq
gen_tim_cnt

FIG. 65

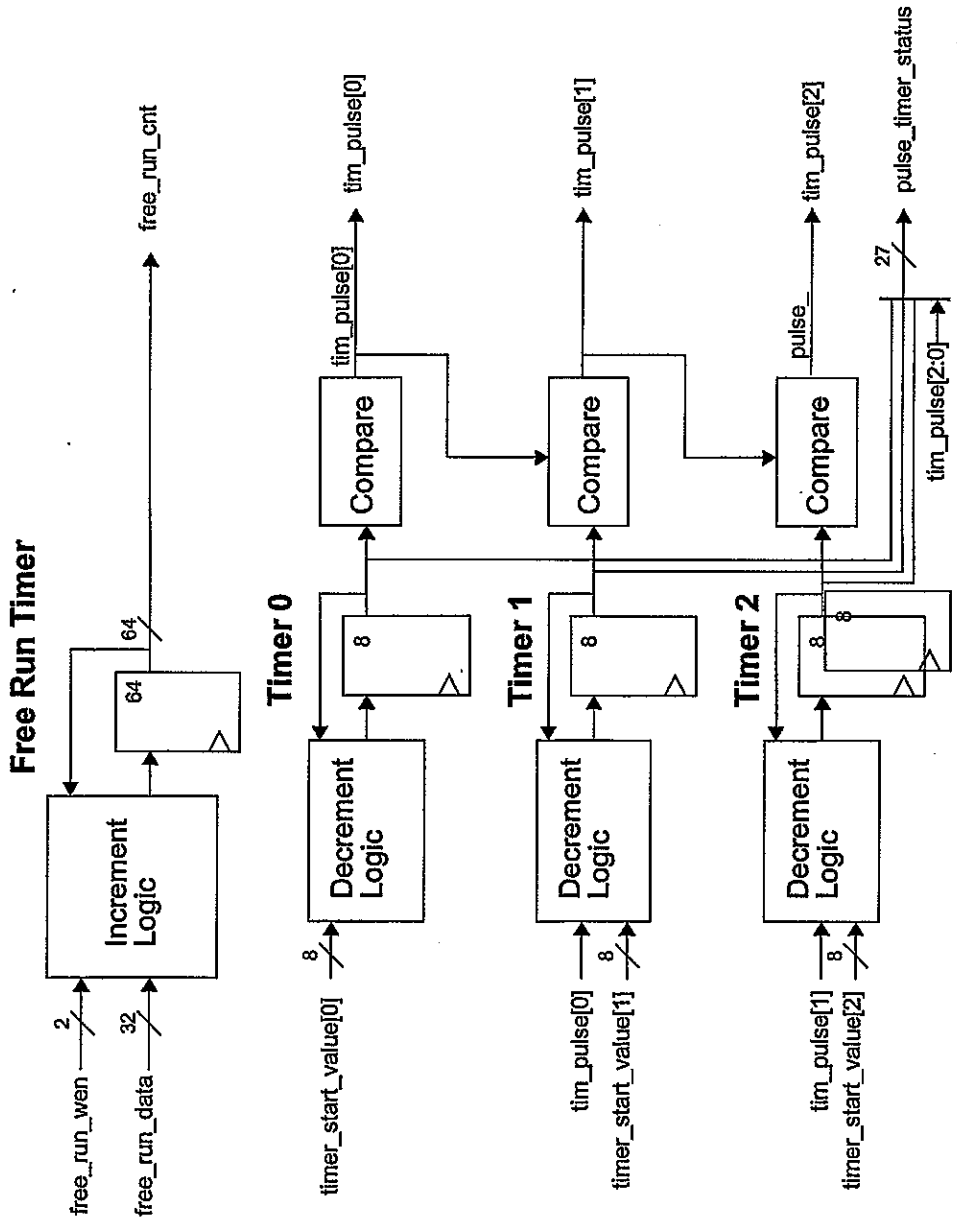

Free Run Timer
Increment Logic
free_run_wen
free_run_data
2
32
64
64
free_run_cnt
Timer 0
Decrement Logic
timer_start_value[0]
8
8
Compare
tim_pulse[0]
tim_pulse[0]
Timer 1
Decrement Logic
tim_pulse[0]
timer_start_value[1]
8
8
Compare
tim_pulse[1]
Timer 2
Decrement Logic
tim_pulse[1]
timer_start_value[2]
8
8
Compare
pulse_
tim_pulse[2]
27
pulse_timer_status
tim_pulse[2:0]

FIG. 66

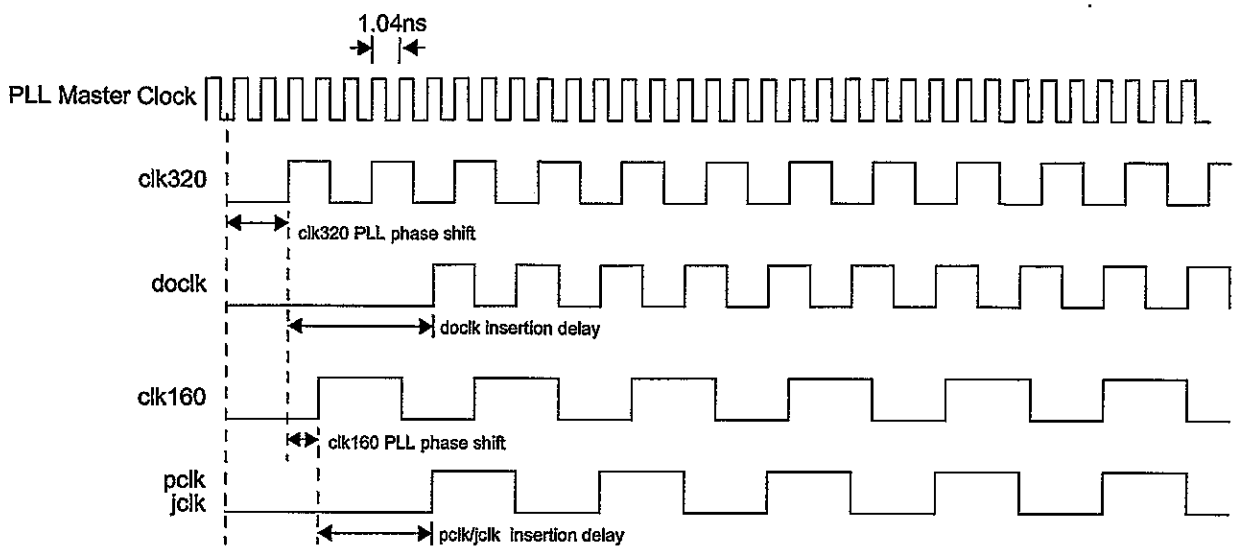

1.04ns
PLL Master Clock
clk320
clk320 PLL phase shift
doclk
doclk insertion delay
clk160
clk160 PLL phase shift
pclk
jclk
pclk/jclk insertion delay

FIG. 67

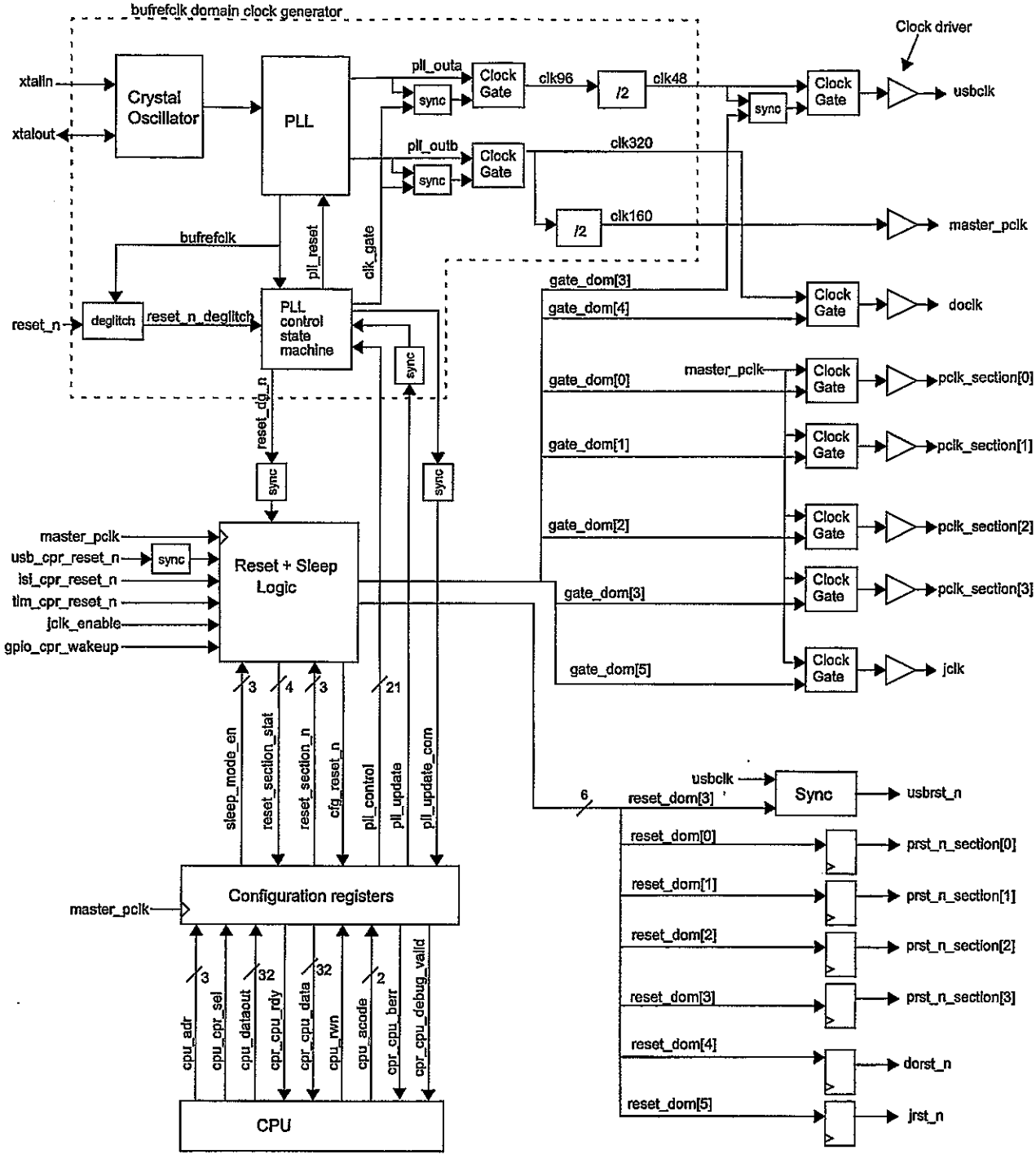

bufrefclk domain clock generator
xtalin
xtalout
Crystal Oscillator
PLL
pll_outa
pll_outb
sync
Clock Gate
clk96
/2
clk48
clk320
clk160
Clock driver
usbclk
master_pclk
bufrefclk
pll_reset
clk_gate
reset_n
deglitch
reset_n_deglitch
PLL control state machine
reset_dg_n
gate_dom[3]
gate_dom[4]
gate_dom[0]
gate_dom[1]
gate_dom[2]
gate_dom[5]
doclk
pclk_section[0]
pclk_section[1]
pclk_section[2]
pclk_section[3]
jclk
usb_cpr_reset_n
isi_cpr_reset_n
tim_cpr_reset_n
jclk_enable
gpio_cpr_wakeup
Reset + Sleep Logic
sleep_mode_en
reset_section_stat
reset_section_n
cfg_reset_n
pll_control
pll_update
pll_update_com
Configuration registers
usbclk
Sync
usbrst_n
reset_dom[3]
reset_dom[0]
reset_dom[1]
reset_dom[2]
reset_dom[4]
reset_dom[5]
prst_n_section[0]
prst_n_section[1]
prst_n_section[2]
prst_n_section[3]
dorst_n
jrst_n
cpu_adr
cpu_cpr_sel
cpu_dataout
cpr_cpu_rdy
cpr_cpu_data
cpu_rwn
cpu_acode
cpr_cpu_berr
cpr_cpu_debug_valid
CPU

FIG. 68

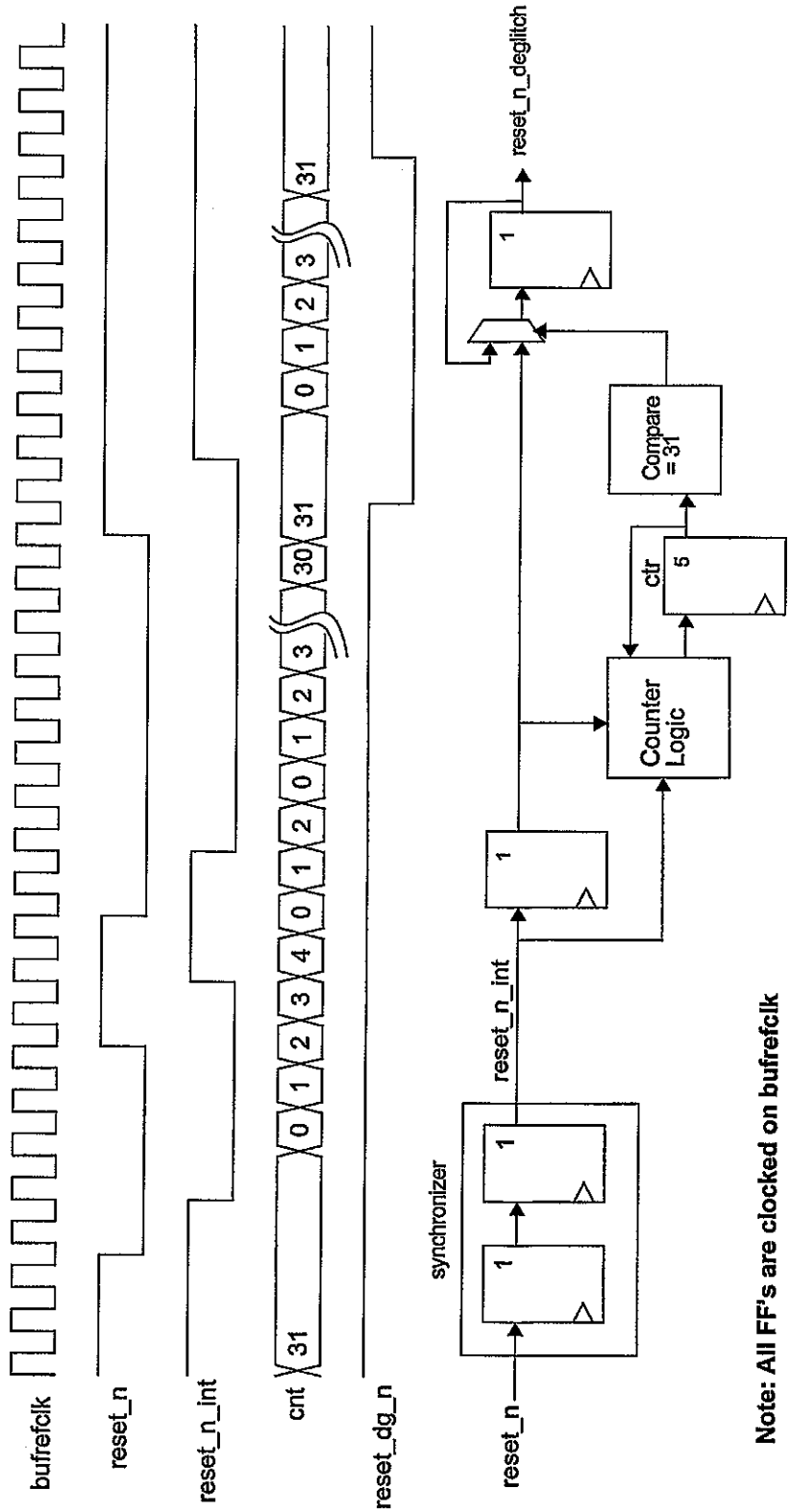

bufrefclk
reset_n
reset_n_int
cnt
reset_dg_n
reset_n
synchronizer
reset_n_int
reset_n_deglitch
Counter Logic
ctr
Compare = 31
Note: All FF's are clocked on bufrefclk

FIG. 69

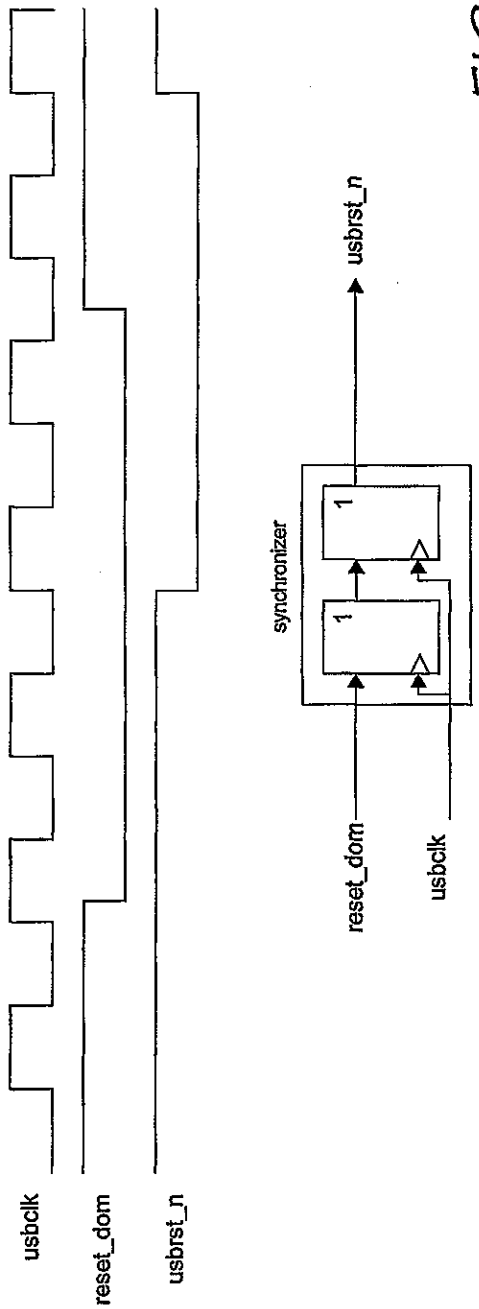

usbclk
reset_dom
usbrst_n
synchronizer
1
1
reset_dom
usbclk
usbrst_n

FIG. 70

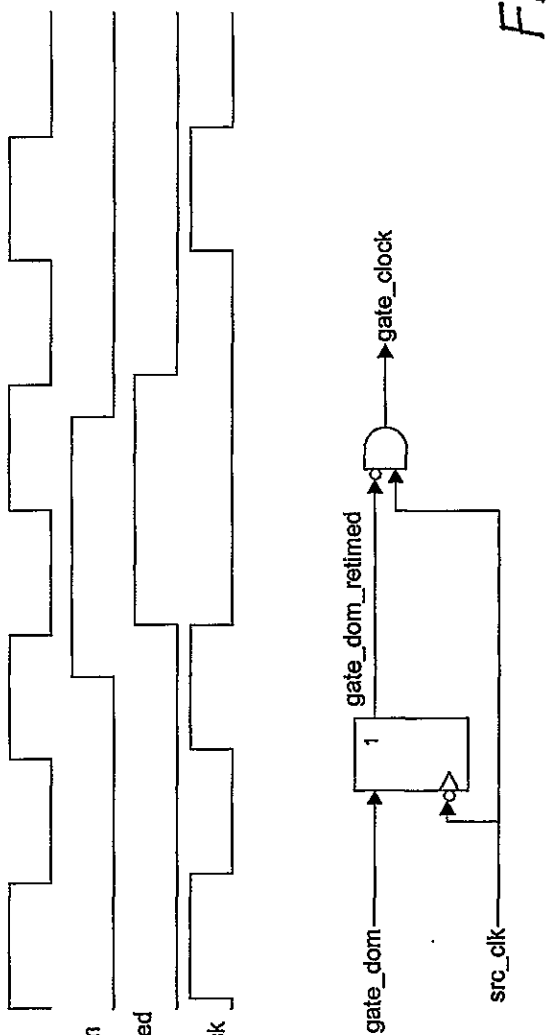

src_clk
gate_dom
gate_dom_retimed
gate_clock
gate_dom
1
gate_dom_retimed
gate_clock
src_clk

FIG. 71

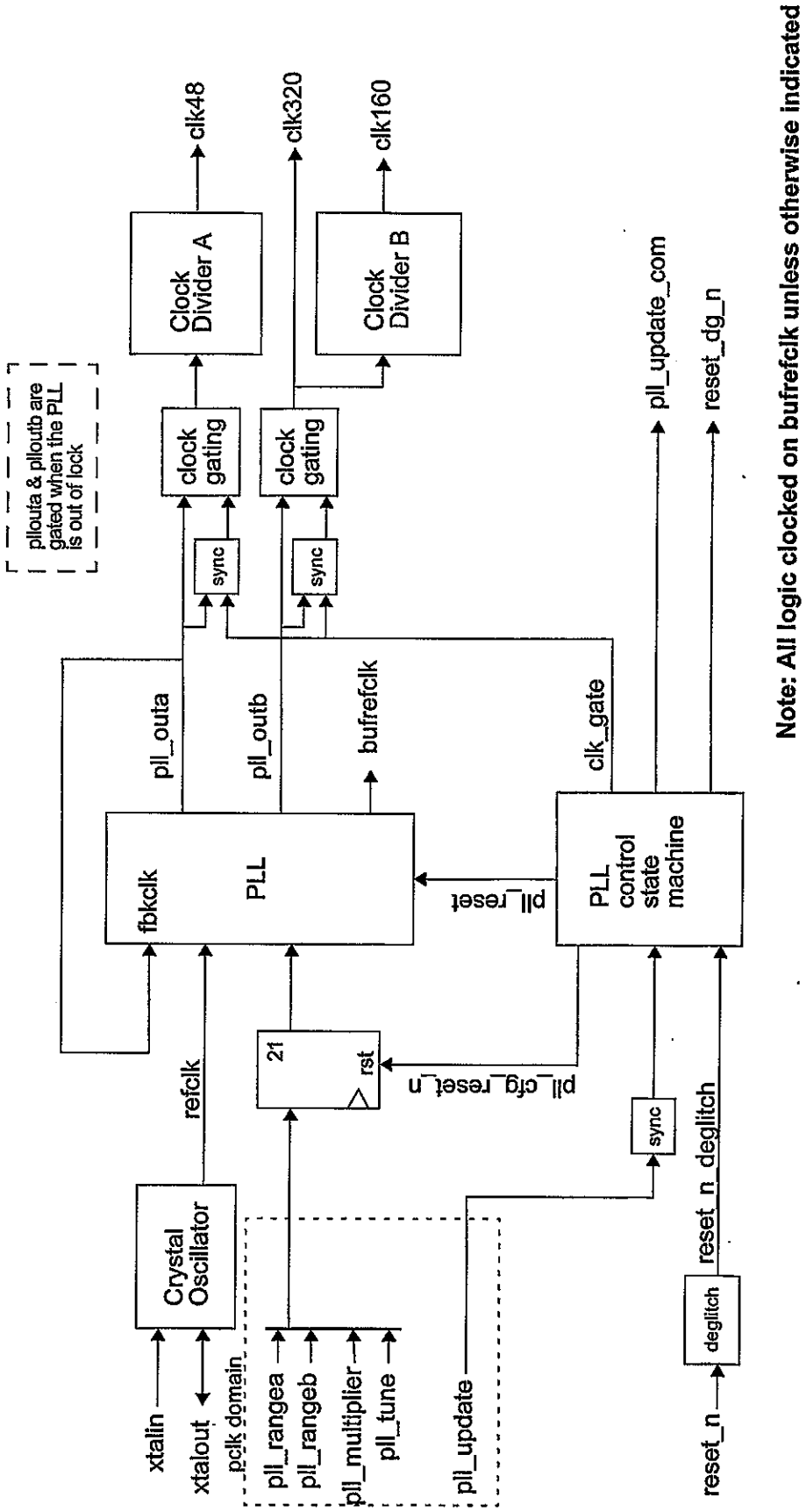

plloutb are gated when the PLL is out of lock
plla & plloutb are gated when the PLL is out of lock
clk48
clk320
clk160
Clock Divider A
Clock Divider B
clock gating
clock gating
sync
sync
pll_outa
pll_outb
bufrefclk
clk_gate
pll_update_com
reset_dg_n
fbkclk
PLL
pll_reset
PLL control state machine
refclk
21
rst
pll_cfg_reset_n
sync
reset_n_deglitch
deglitch
Crystal Oscillator
xtalin
xtalout
pclk domain
pll_rangea
pll_rangeb
pll_multiplier
pll_tune
pll_update
reset_n
Note: All logic clocked on bufrefclk unless otherwise indicated

FIG. 72

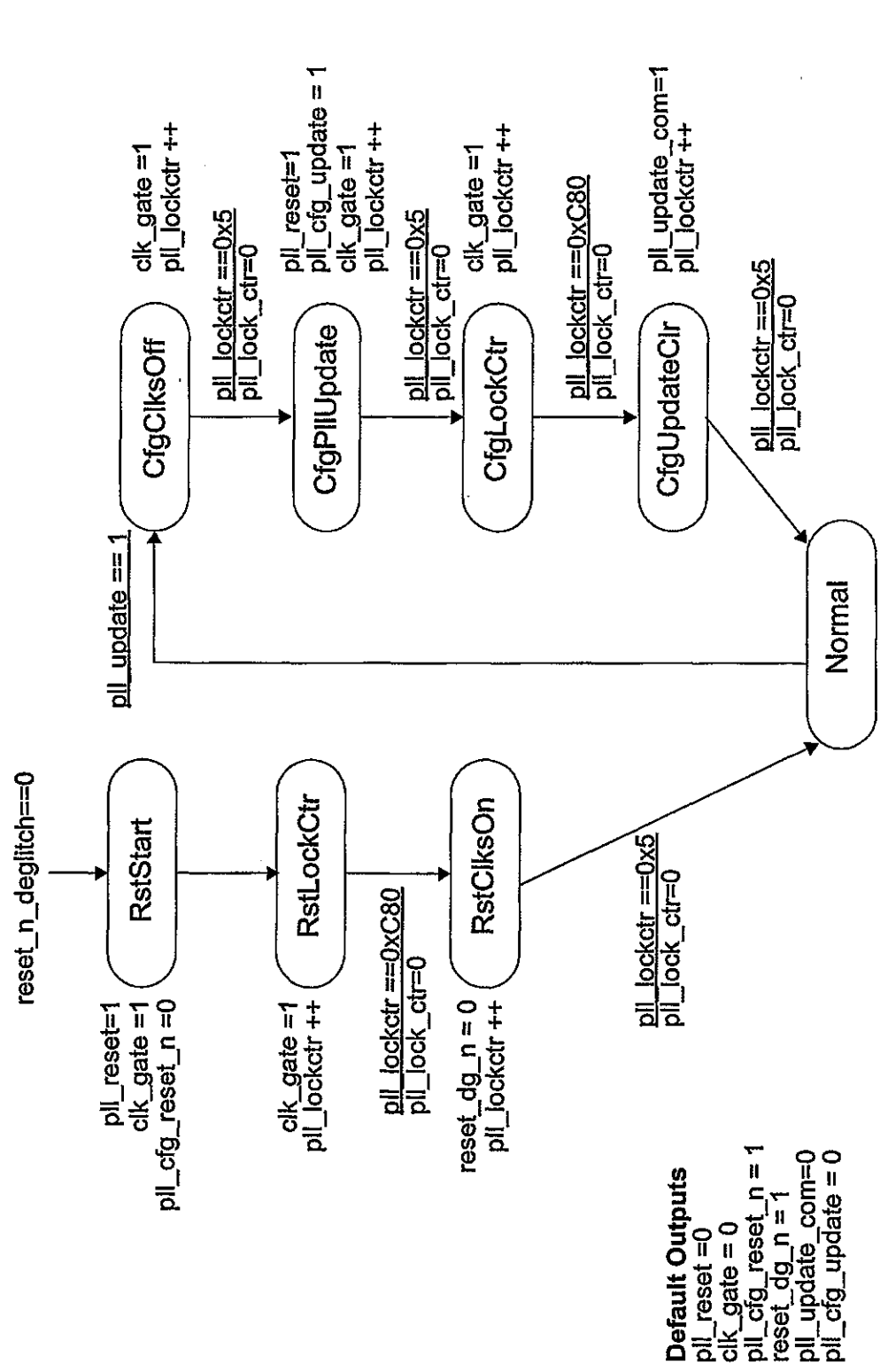

reset_n_deglitch==0
RstStart
pll_reset=1
clk_gate =1
pll_cfg_reset_n =0
RstLockCtr
clk_gate =1
pll_lockctr ++
pll_lockctr ==0xC80
pll_lock_ctr=0
RstClksOn
reset_dg_n = 0
pll_lockctr ++
pll_lockctr ==0x5
pll_lock_ctr=0
pll_update == 1
CfgClksOff
clk_gate =1
pll_lockctr ++
pll_lockctr ==0x5
pll_lock_ctr=0
CfgPllUpdate
pll_reset=1
pll_cfg_update = 1
clk_gate =1
pll_lockctr ++
pll_lockctr ==0x5
pll_lock_ctr=0
CfgLockCtr
clk_gate =1
pll_lockctr ++
pll_lockctr ==0xC80
pll_lock_ctr=0
CfgUpdateClr
pll_update_com=1
pll_lockctr ++
pll_lockctr ==0x5
pll_lock_ctr=0
Normal
Default Outputs
pll_reset =0
clk_gate = 0
pll_cfg_reset_n = 1
reset_dg_n = 1
pll_update_com=0
pll_cfg_update = 0

FIG. 73

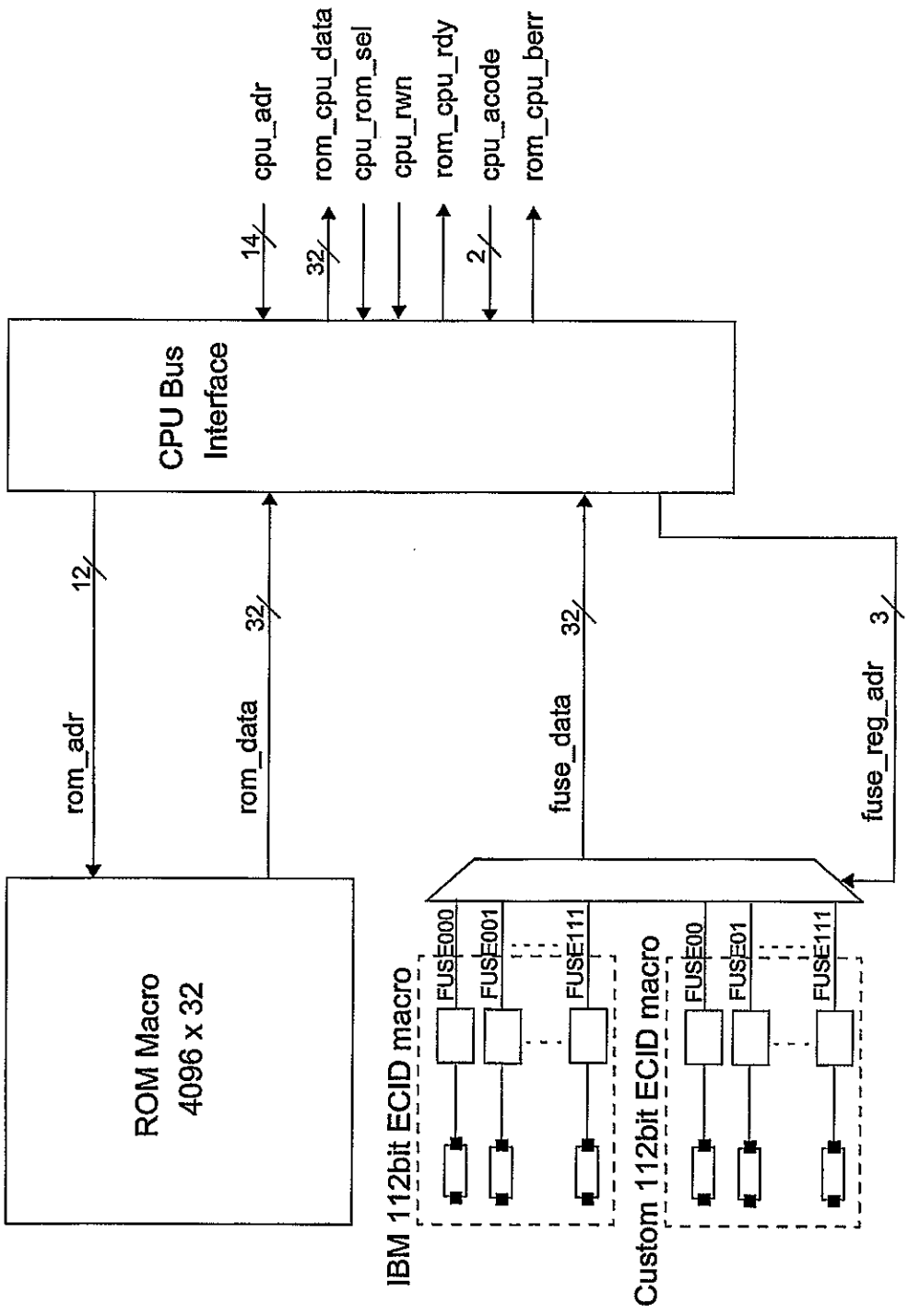

cpu_adr
rom_cpu_data
cpu_rom_sel
cpu_rwn
rom_cpu_rdy
cpu_acode
rom_cpu_berr
14
32
2
CPU Bus Interface
12
32
32
3
rom_adr
rom_data
fuse_data
fuse_reg_adr
ROM Macro 4096 x 32
IBM 112bit ECID macro
FUSE000
FUSE001
FUSE111
Custom 112bit ECID macro
FUSE00
FUSE01
FUSE111

FIG. 74

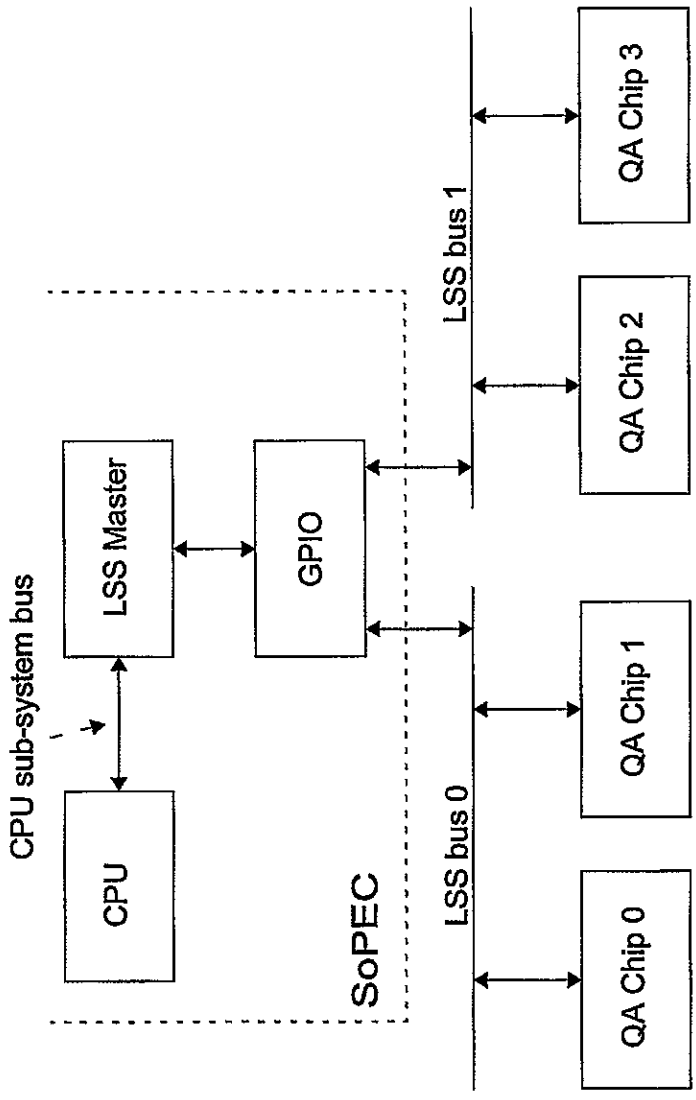

CPU sub-system bus
CPU
LSS Master
GPIO
SoPEC
LSS bus 0
LSS bus 1
QA Chip 0
QA Chip 1
QA Chip 2
QA Chip 3

FIG. 75

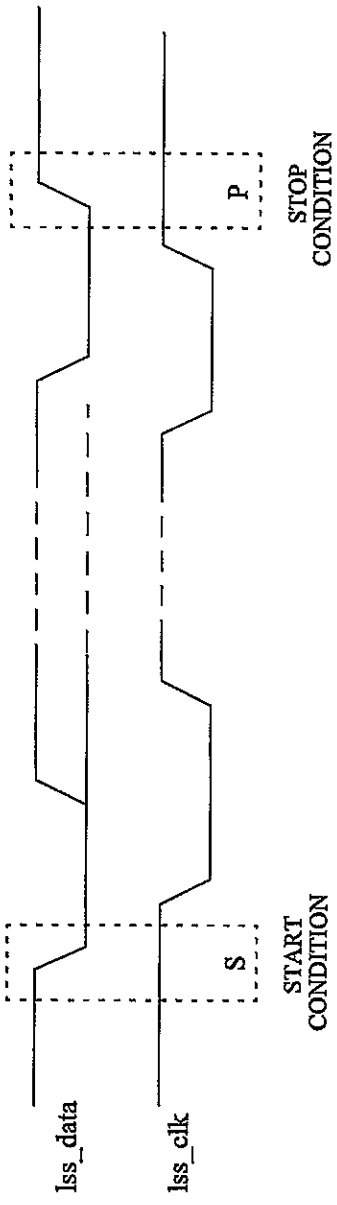

lss_data
lss_clk
S
START CONDITION
P
STOP CONDITION

FIG. 76

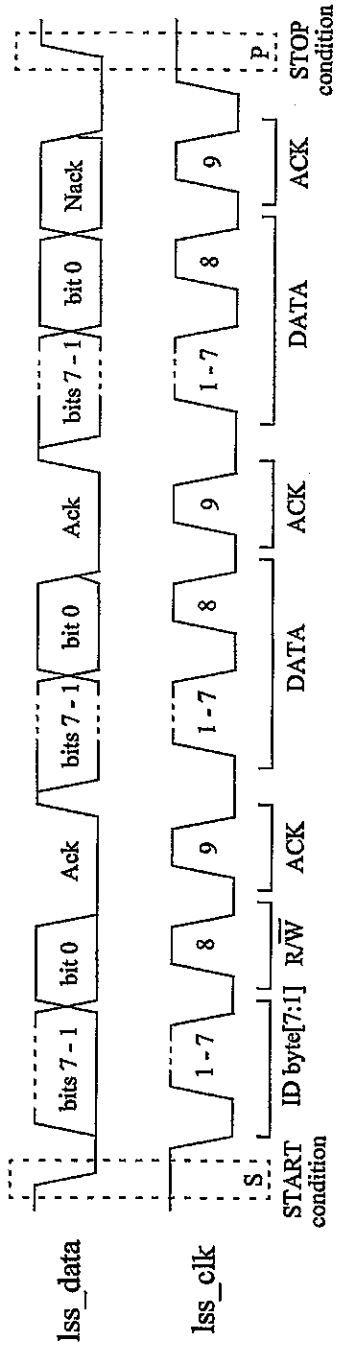

lss_data
lss_clk
S
START condition
1 - 7
ID byte[7:1]
R/W
ACK
bits 7 - 1
bit 0
Ack
Nack
DATA
P
STOP condition

FIG. 77

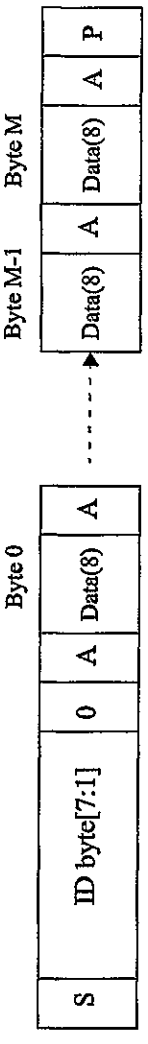

S = Start condition
A = Ack
N = Nack
P = Stop condition
Shaded bits driven by slave
Byte 0
Byte M-1
Byte M
S
ID byte[7:1]
0
A
Data(8)
A
Data(8)
A
P

FIG. 78

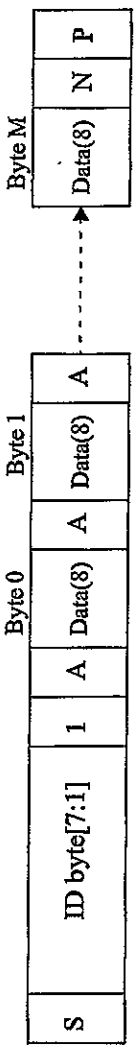

S = Start condition
A = Ack
N = Nack
P = Stop condition
Shaded bits driven by slave
Byte 0
Byte 1
Byte M
S
ID byte[7:1]
1
A
Data(8)
A
Data(8)
A
Data(8)
N
P

FIG. 79

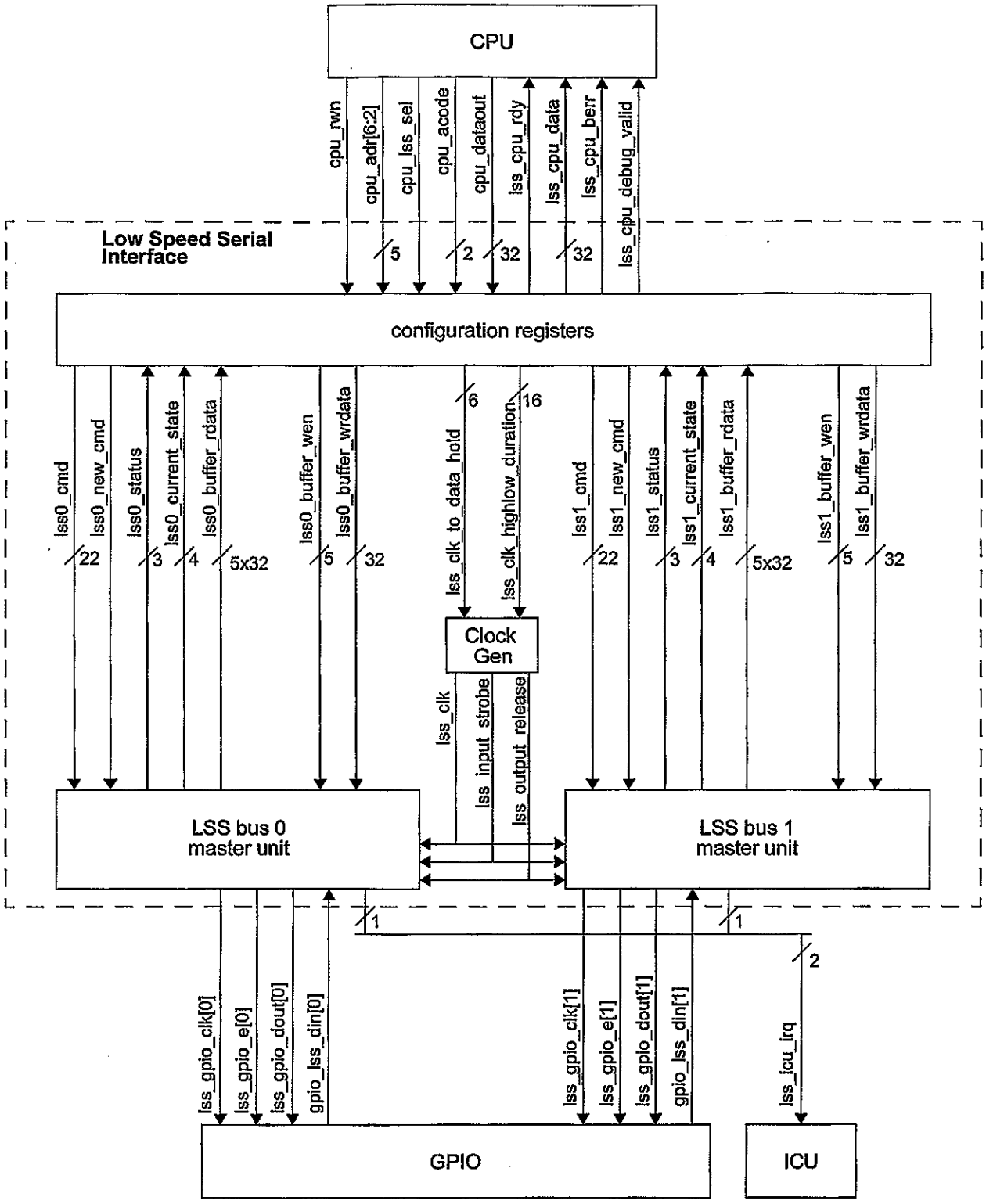

CPU
cpu_rwn
cpu_adr[6:2]
cpu_lss_sel
cpu_acode
cpu_dataout
lss_cpu_rdy
lss_cpu_data
lss_cpu_berr
lss_cpu_debug_valid
5
2
32
32
Low Speed Serial Interface
configuration registers
lss0_cmd
lss0_new_cmd
lss0_status
lss0_current_state
lss0_buffer_rdata
lss0_buffer_wen
lss0_buffer_wrdata
22
3
4
5x32
5
32
lss_clk_to_data_hold
lss_clk_highlow_duration
6
16
Clock Gen
lss_clk
lss_input_strobe
lss_output_release
lss1_cmd
lss1_new_cmd
lss1_status
lss1_current_state
lss1_buffer_rdata
lss1_buffer_wen
lss1_buffer_wrdata
22
3
4
5x32
5
32
LSS bus 0 master unit
LSS bus 1 master unit
1
1
2
lss_gpio_clk[0]
lss_gpio_e[0]
lss_gpio_dout[0]
gpio_lss_din[0]
lss_gpio_clk[1]
lss_gpio_e[1]
lss_gpio_dout[1]
gpio_lss_din[1]
lss_icu_irq
GPIO
ICU

FIG. 80

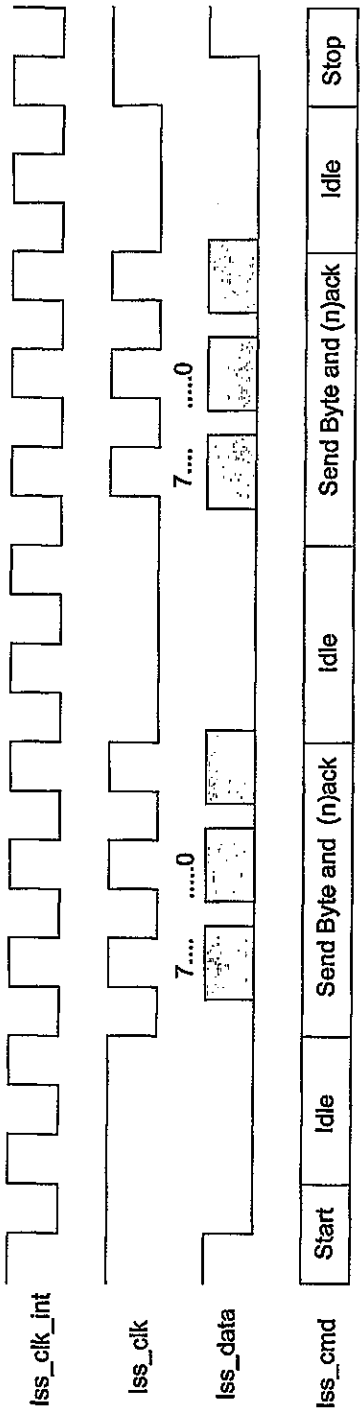

Tp
lss_clk_int
lss_clk
lss_data
lss_cmd
7.... 0
7.... 0
Start
Idle
Send Byte and (n)ack
Idle
Send Byte and (n)ack
Idle
Stop

FIG. 81

Start Condition
Stop Condition
Bus state case 1
(lss_clk=1,lss_data=1)
Bus state case 2
(lss_clk=1,lss_data=0)
Bus state case 3
(lss_clk=0,lss_data=1)
Bus state case 4
(lss_clk=0,lss_data=0)
lss_clk_int
lss_data
lss_clk

FIG. 82

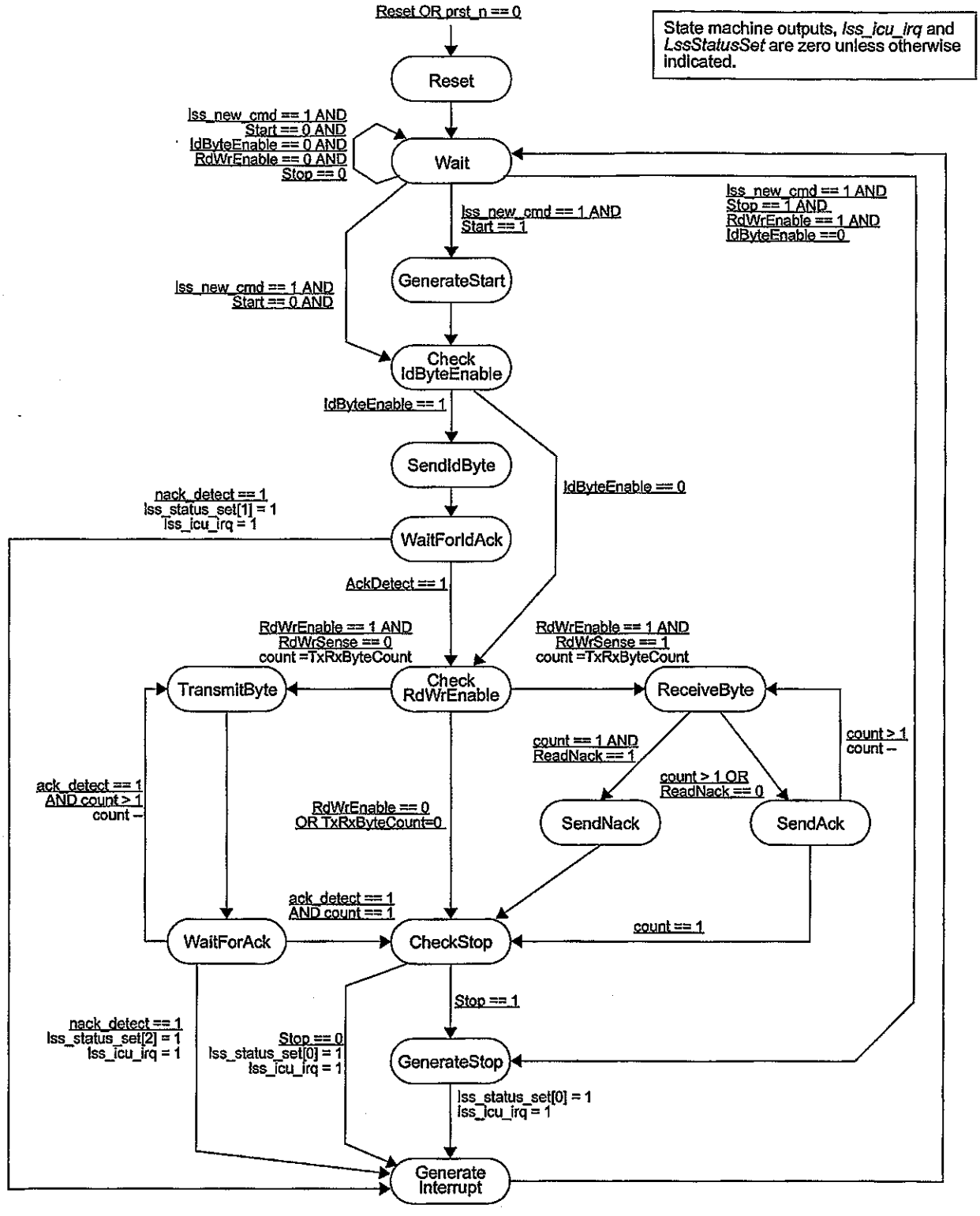

Reset OR prst_n == 0
State machine outputs, lss_icu_irq and LssStatusSet are zero unless otherwise indicated.
Reset
Wait
lss_new_cmd == 1 AND Start == 0 AND IdByteEnable == 0 AND RdWrEnable == 0 AND Stop == 0
lss_new_cmd == 1 AND Start == 1
lss_new_cmd == 1 AND Stop == 1 AND RdWrEnable == 1 AND IdByteEnable ==0
GenerateStart
lss_new_cmd == 1 AND Start == 0 AND
Check IdByteEnable
IdByteEnable == 1
IdByteEnable == 0
SendIdByte
nack_detect == 1 lss_status_set[1] = 1 lss_icu_irq = 1
WaitForIdAck
AckDetect == 1
RdWrEnable == 1 AND RdWrSense == 0 count =TxRxByteCount
RdWrEnable == 1 AND RdWrSense == 1 count =TxRxByteCount
TransmitByte
Check RdWrEnable
ReceiveByte
count == 1 AND ReadNack == 1
count > 1 OR ReadNack == 0
count > 1 count --
ack_detect == 1 AND count > 1 count --
RdWrEnable == 0 OR TxRxByteCount=0
SendNack
SendAck
ack_detect == 1 AND count == 1
WaitForAck
CheckStop
count == 1
Stop == 1
nack_detect == 1 lss_status_set[2] = 1 lss_icu_irq = 1
Stop == 0 lss_status_set[0] = 1 lss_icu_irq = 1
GenerateStop
lss_status_set[0] = 1 lss_icu_irq = 1
Generate Interrupt

FIG. 83

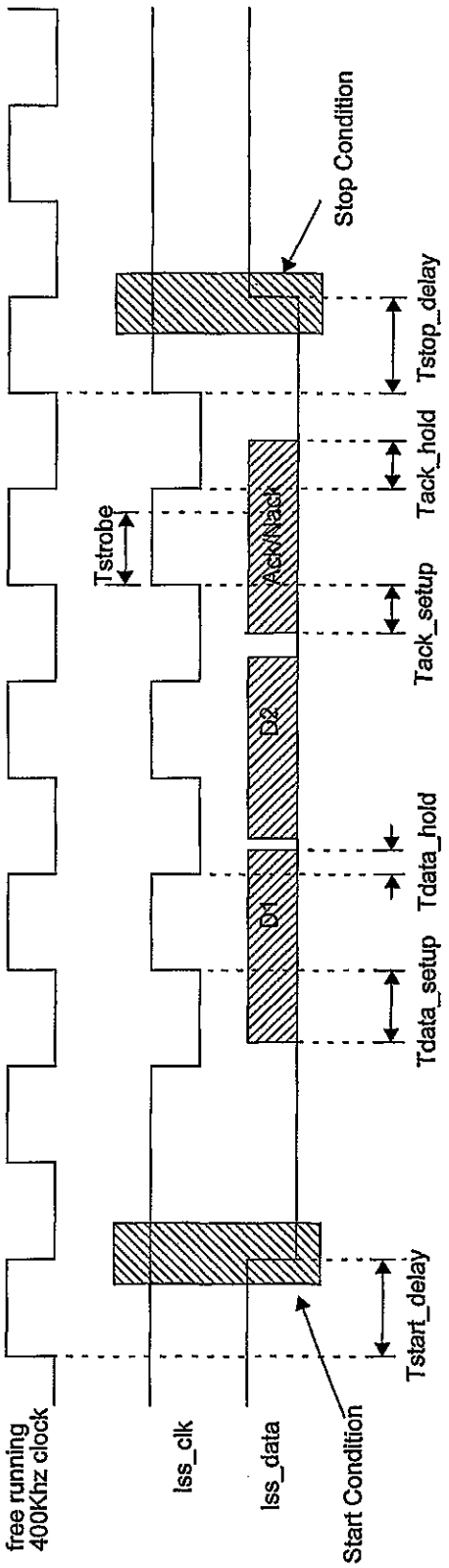

free running
400Khz clock
Tp
Tp
lss_clk
lss_data
Start Condition
Tstart_delay
Tdata_setup
D1
Tdata_hold
D2
Tack_setup
Ack/Nack
Tstrobe
Tack_hold
Tstop_delay
Stop Condition

FIG. 84

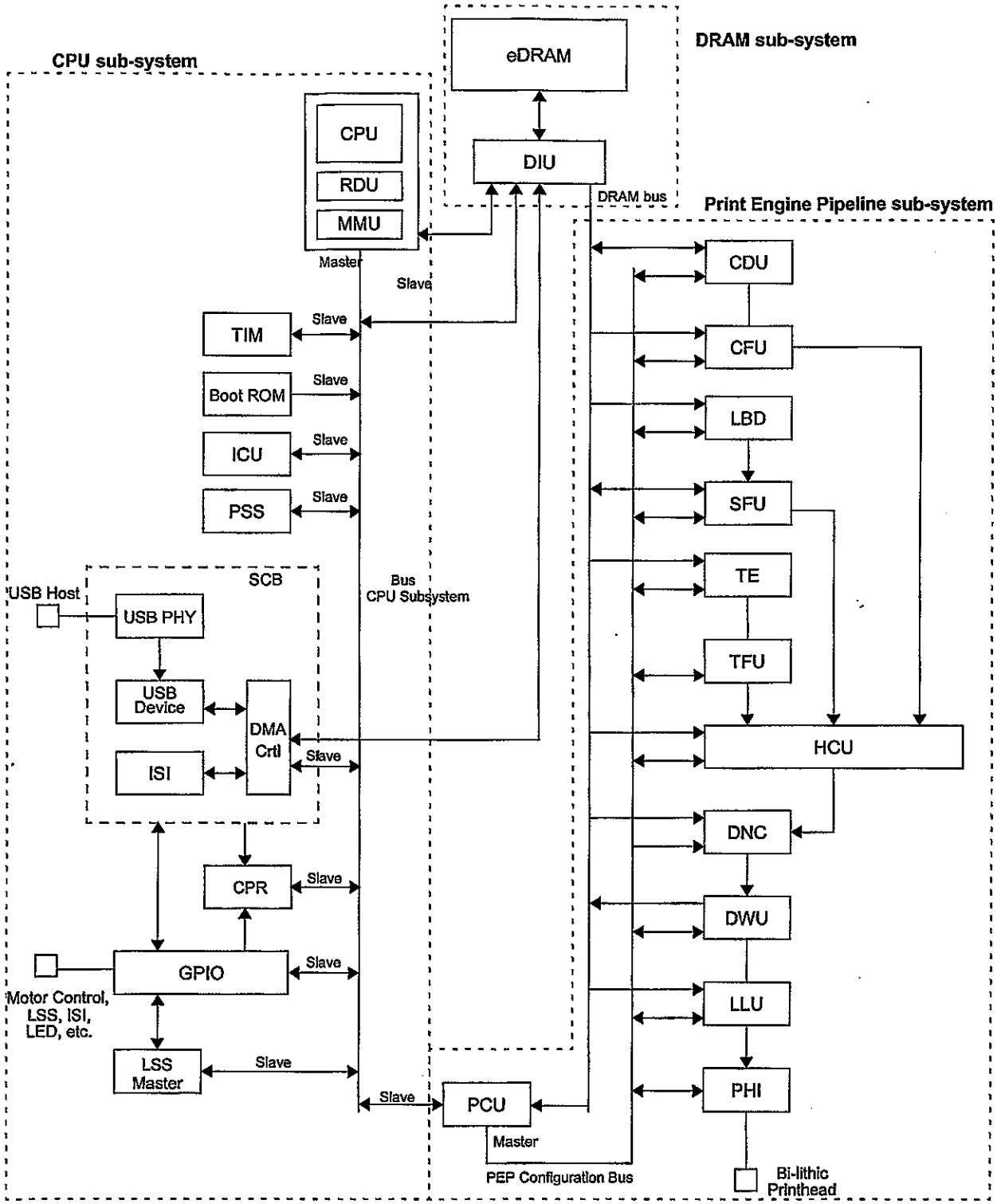

CPU sub-system
DRAM sub-system
eDRAM
DIU
DRAM bus
CPU
RDU
MMU
Master
Slave
Print Engine Pipeline sub-system
CDU
CFU
LBD
SFU
TE
TFU
HCU
DNC
DWU
LLU
PHI
TIM
Boot ROM
ICU
PSS
SCB
USB Host
USB PHY
USB Device
ISI
DMA Crtl
Bus
CPU Subsystem
CPR
GPIO
Motor Control, LSS, ISI, LED, etc.
LSS Master
PCU
PEP Configuration Bus
Bi-lithic Printhead

FIG. 85

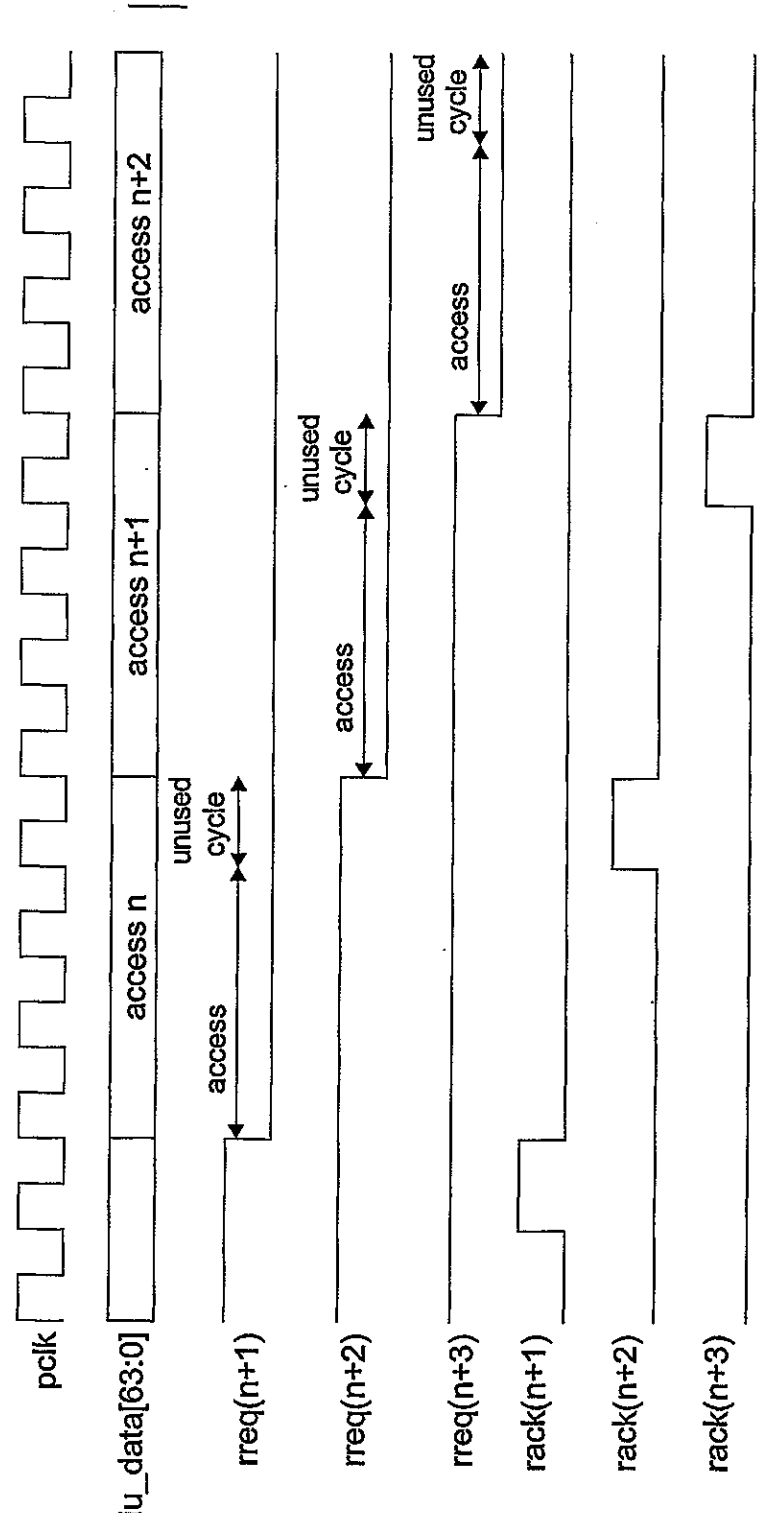

pclk
diu_data[63:0]
access n
access n+1
access n+2
unused cycle
access
unused cycle
access
unused cycle
access
rreq(n+1)
rreq(n+2)
rreq(n+3)
rack(n+1)
rack(n+2)
rack(n+3)

FIG. 86

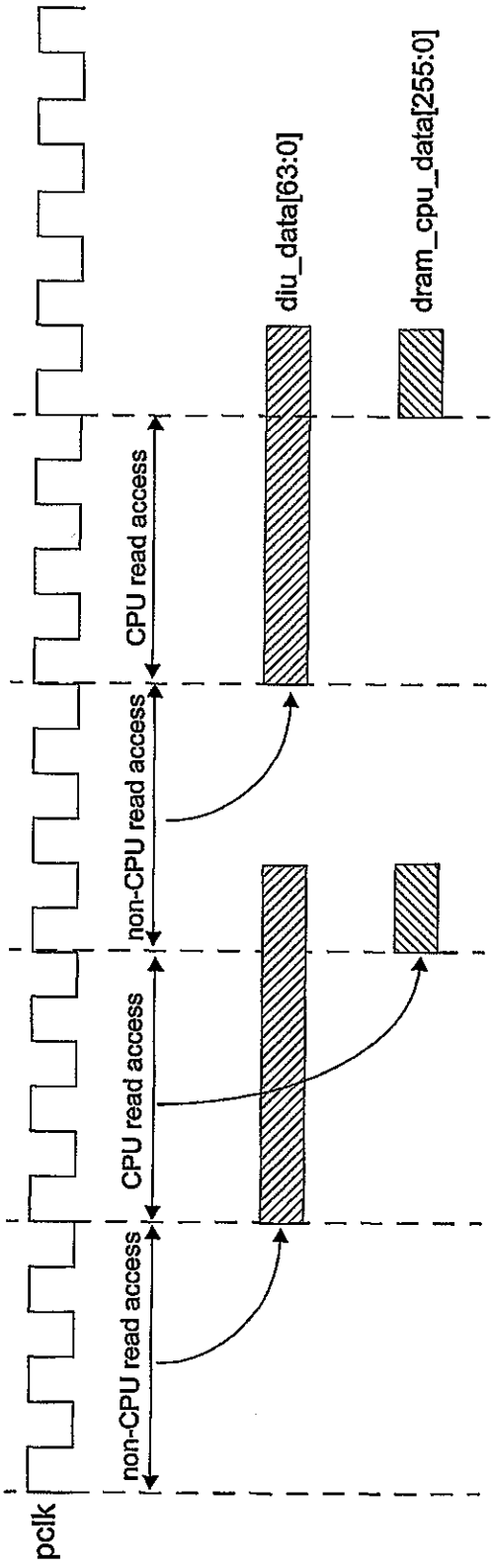

pclk
non-CPU read access
CPU read access
non-CPU read access
CPU read access
diu_data[63:0]
dram_cpu_data[255:0]

FIG. 87

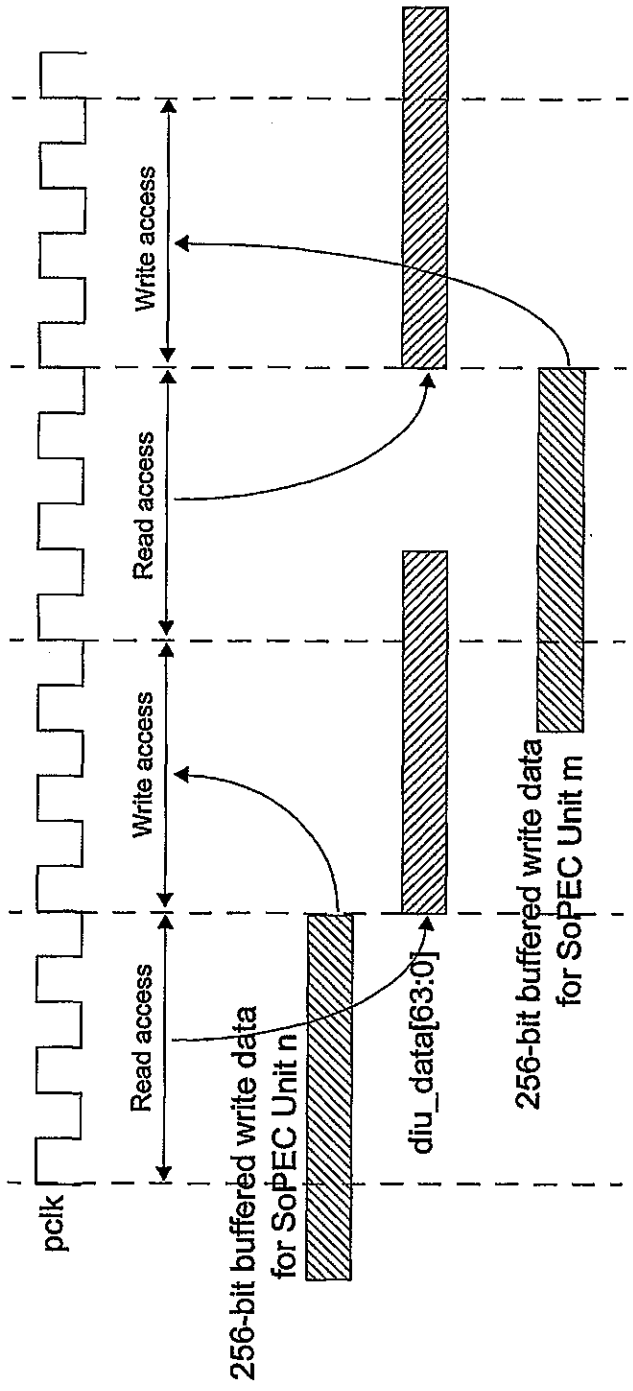

pclk
Read access
Write access
Read access
Write access
256-bit buffered write data for SoPEC Unit n
diu_data[63:0]
256-bit buffered write data for SoPEC Unit m

FIG. 88

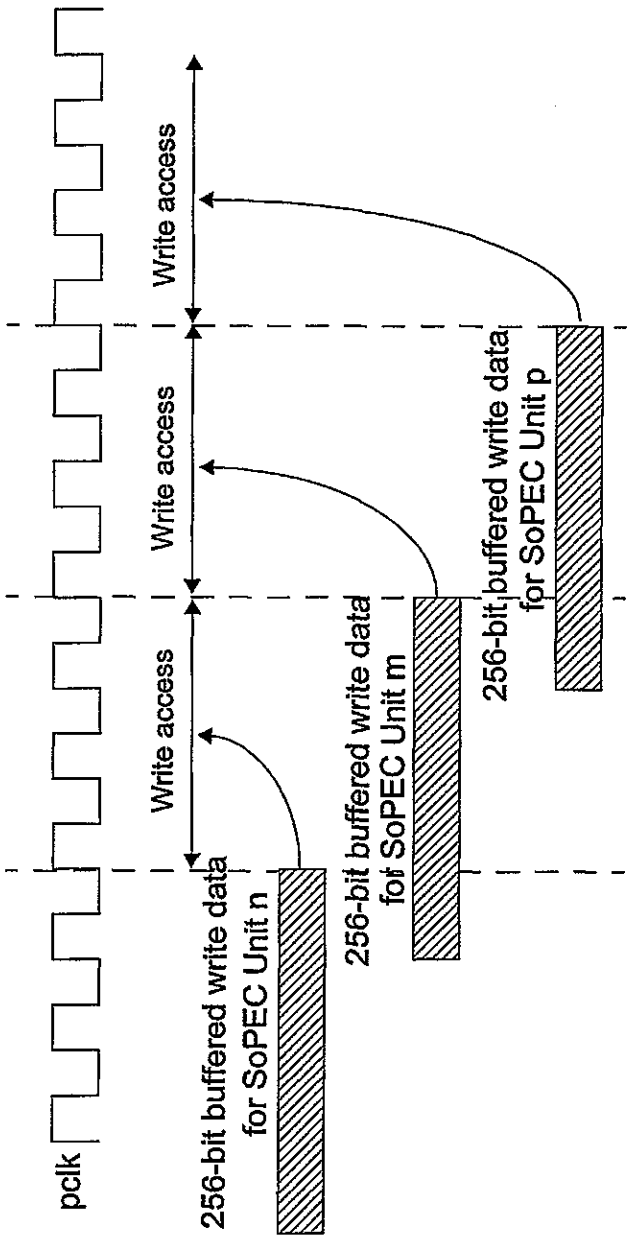

pclk
256-bit buffered write data for SoPEC Unit n
256-bit buffered write data for SoPEC Unit m
256-bit buffered write data for SoPEC Unit p
Write access
Write access
Write access

FIG. 89

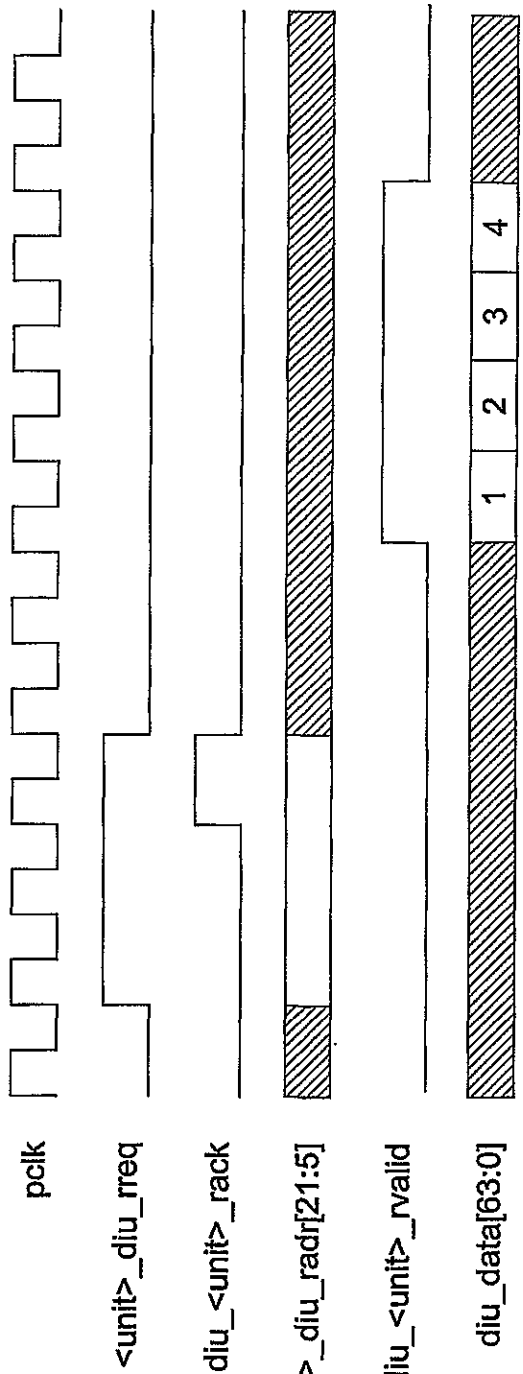

pclk
<unit>_diu_rreq
diu_<unit>_rack
<unit>_diu_radr[21:5]
diu_<unit>_rvalid
diu_data[63:0]
1
2
3
4

FIG. 90

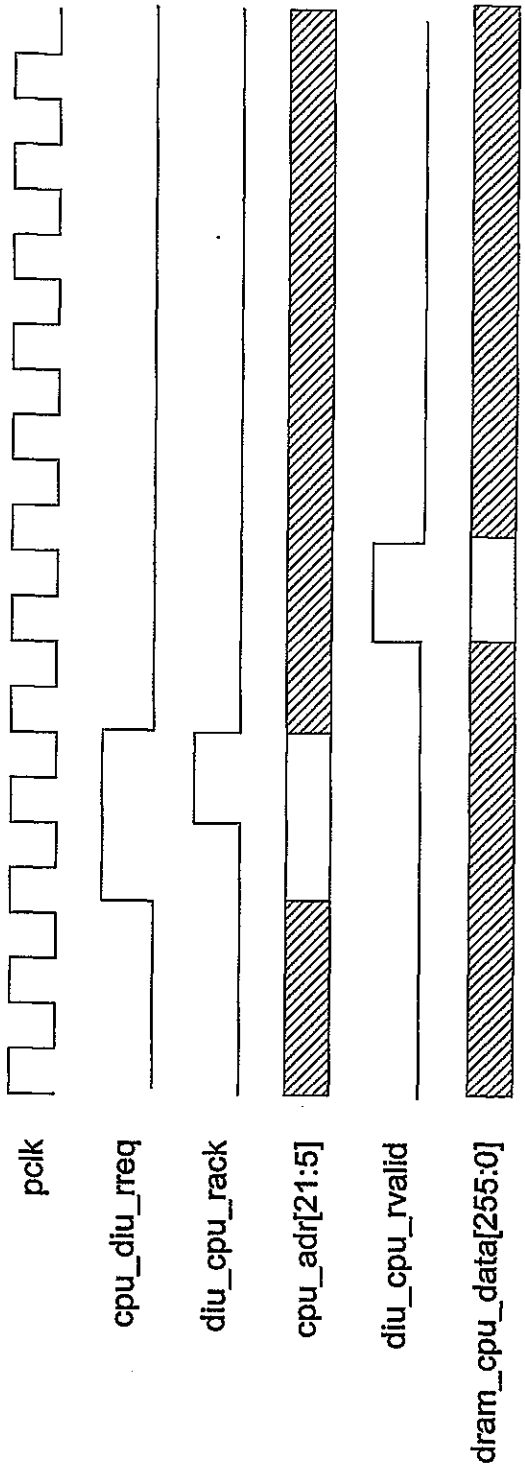

pclk
cpu_diu_rreq
diu_cpu_rack
cpu_adr[21:5]
diu_cpu_rvalid
dram_cpu_data[255:0]

FIG. 91

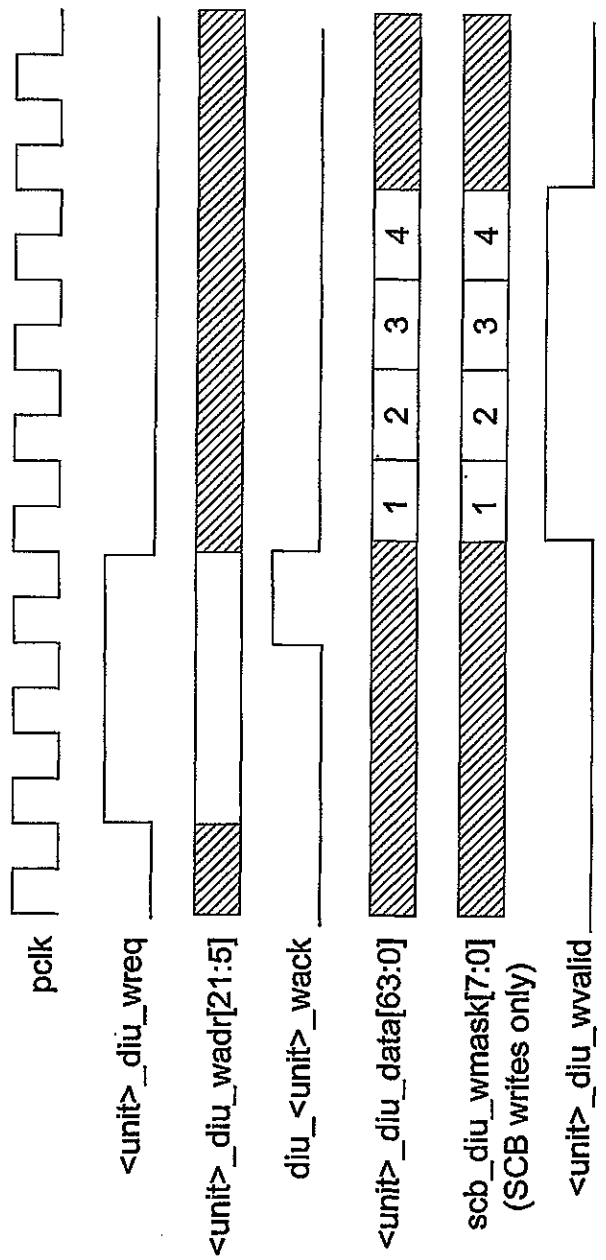

pclk
<unit>_diu_wreq
<unit>_diu_wadr[21:5]
diu_<unit>_wack
<unit>_diu_data[63:0]
1
2
3
4
scb_diu_wmask[7:0]
(SCB writes only)
1
2
3
4
<unit>_diu_wvalid

FIG. 92

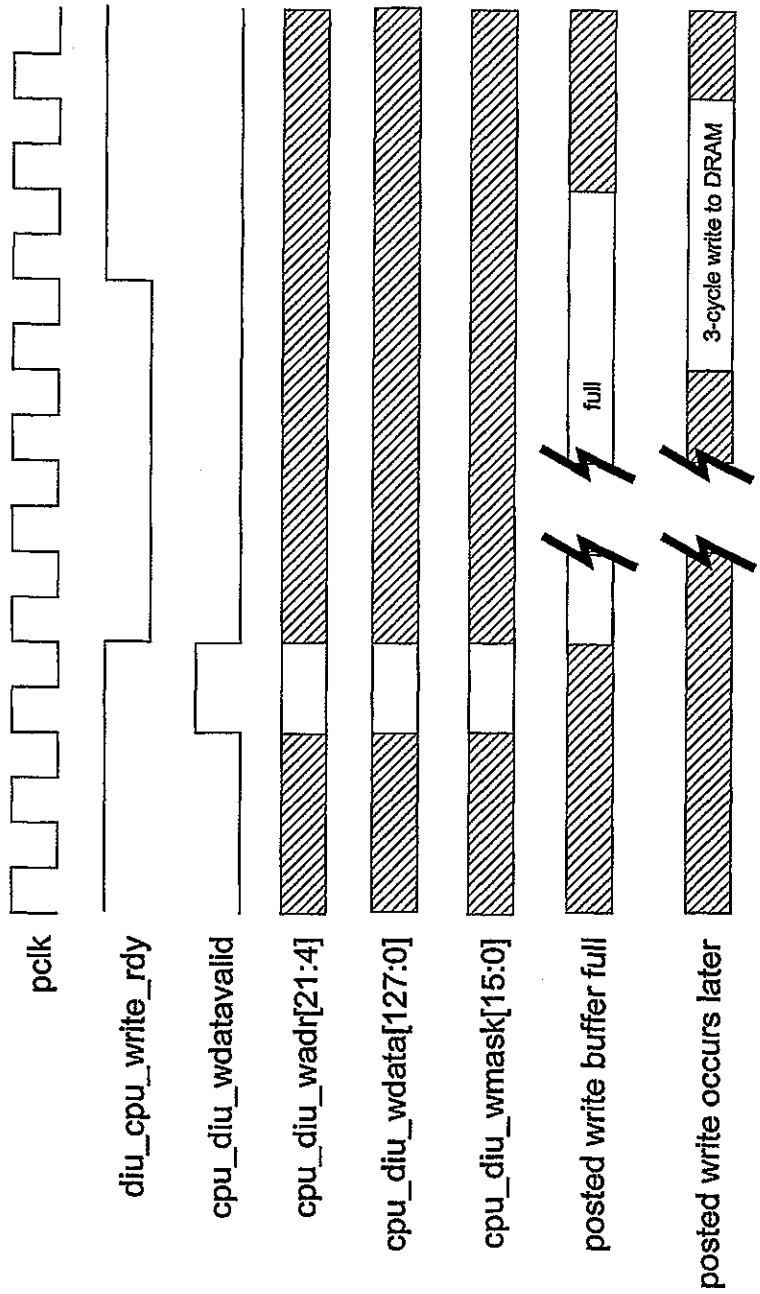

pclk
diu_cpu_write_rdy
cpu_diu_wdatavalid
cpu_diu_wadr[21:4]
cpu_diu_wdata[127:0]
cpu_diu_wmask[15:0]
posted write buffer full
full
posted write occurs later
3-cycle write to DRAM

FIG. 93

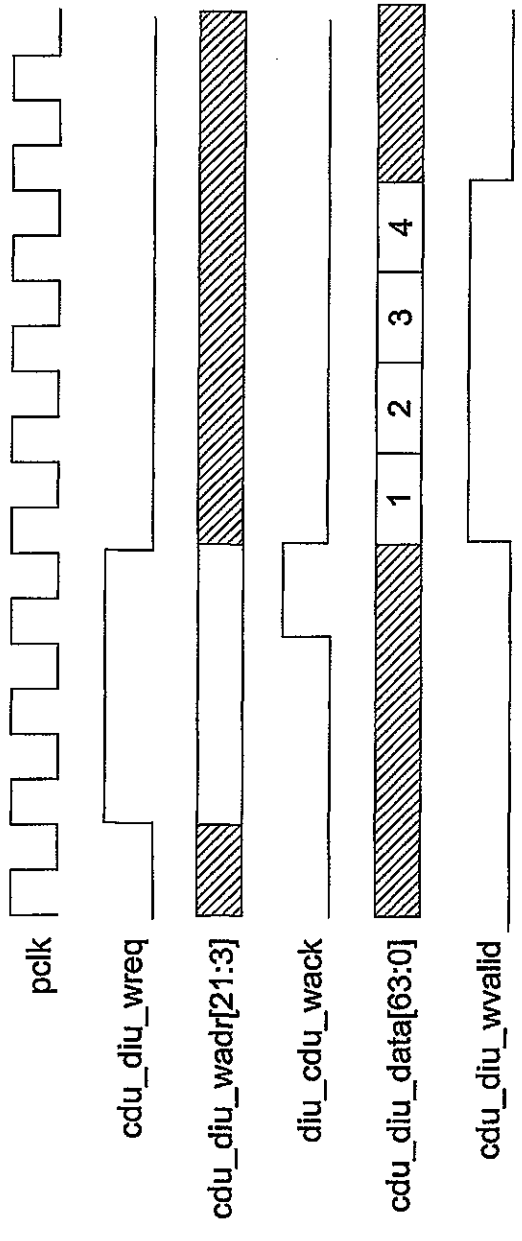

pclk
cdu_diu_wreq
cdu_diu_wadr[21:3]
diu_cdu_wack
cdu_diu_data[63:0]
1
2
3
4
cdu_diu_wvalid

FIG. 94

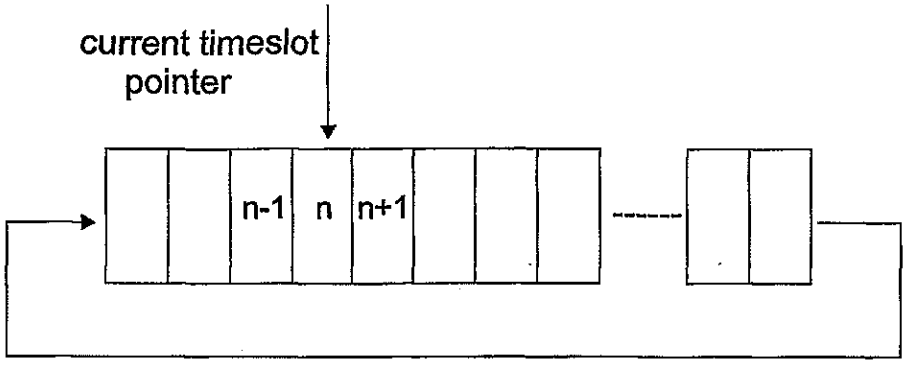


FIG. 95

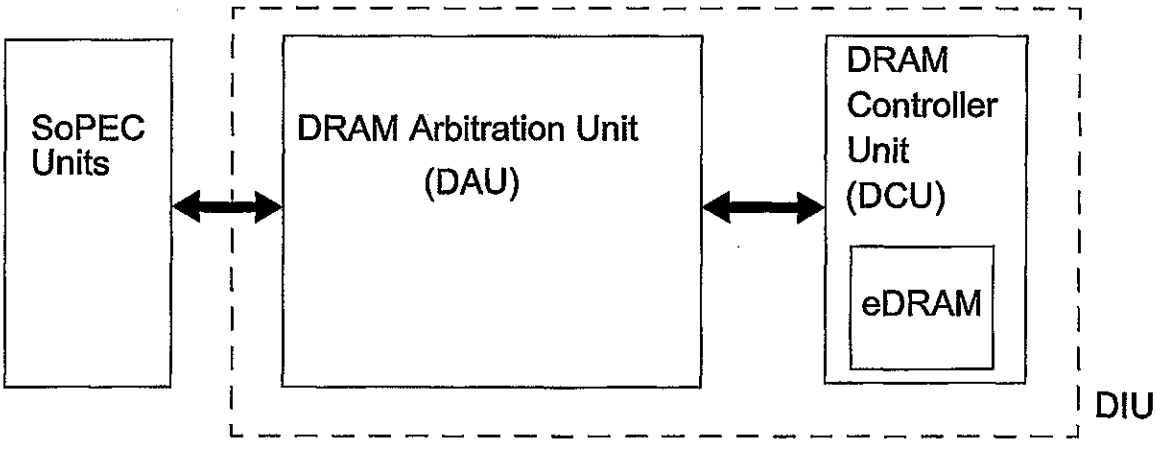


FIG. 100

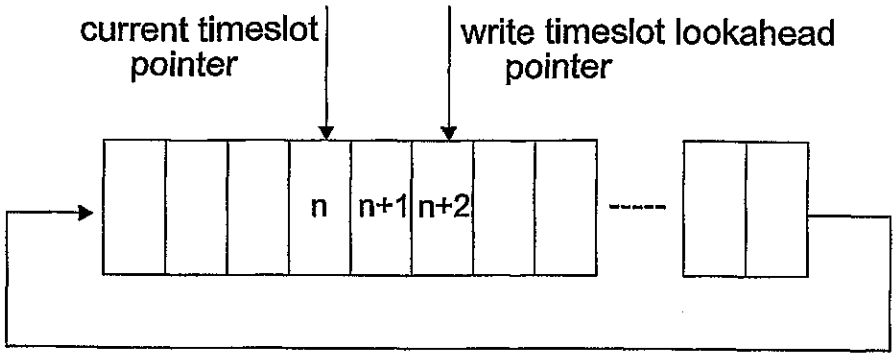

current timeslot pointer
write timeslot lookahead pointer
n
n+1
n+2

FIG. 96

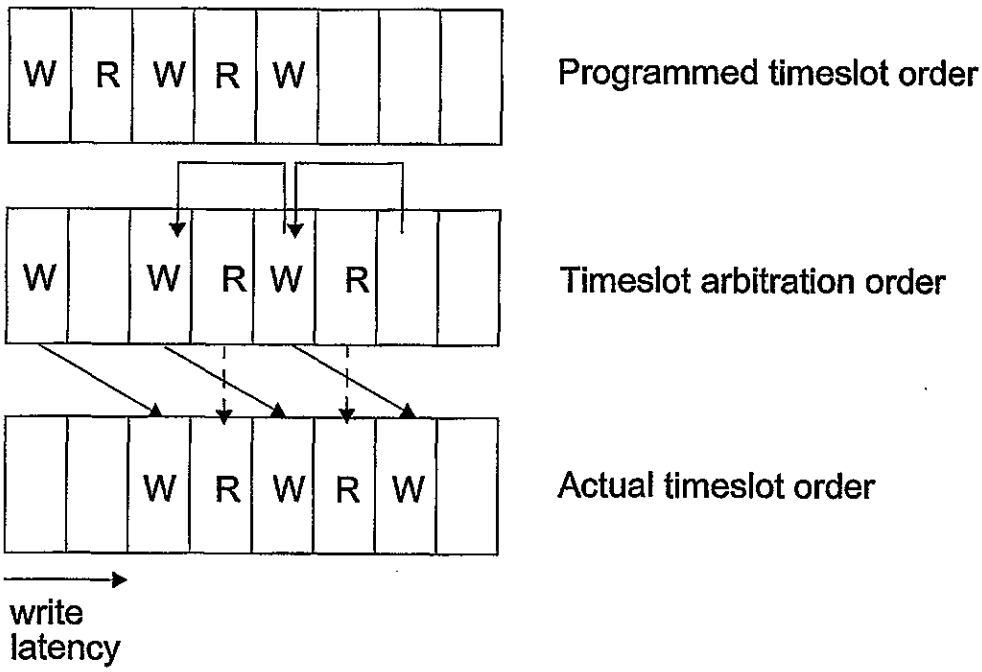

W R W R W
Programmed timeslot order
W W R W R
Timeslot arbitration order
W R W R W
Actual timeslot order
write latency

FIG. 97

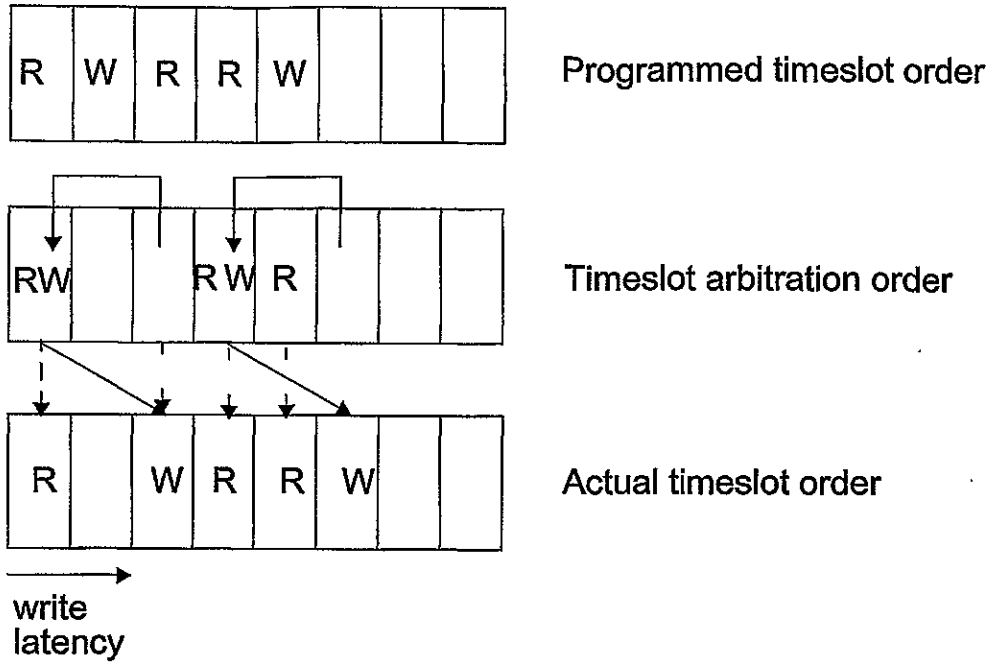

R W R R W
Programmed timeslot order
RW RW R
Timeslot arbitration order
R W R R W
Actual timeslot order
write latency

FIG. 98

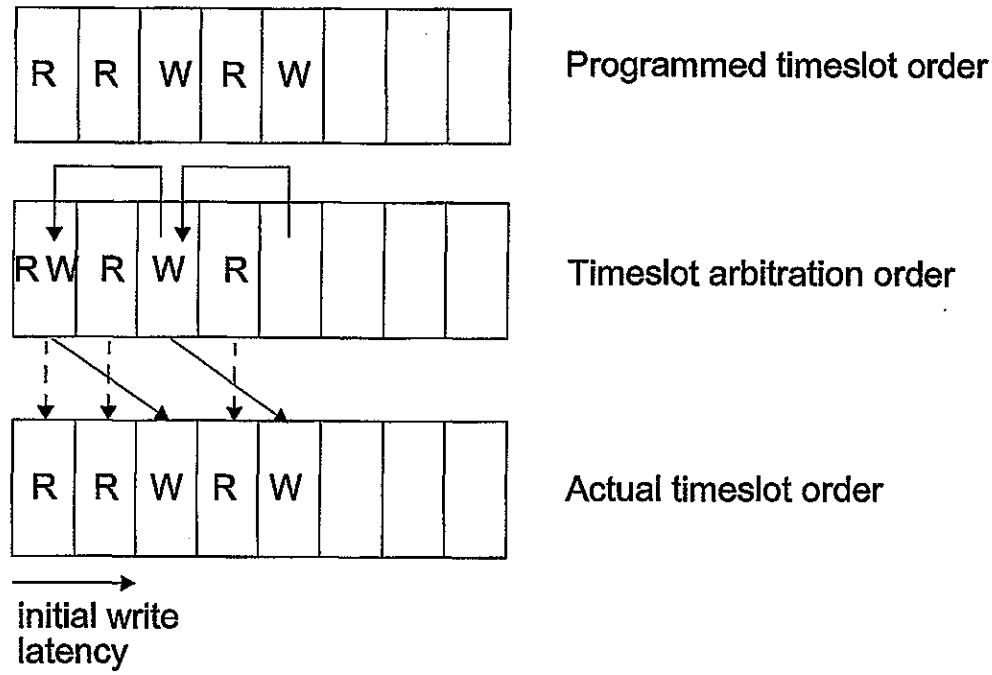

R R W R W
Programmed timeslot order
RW R W R
Timeslot arbitration order
R R W R W
Actual timeslot order
initial write latency

FIG. 99

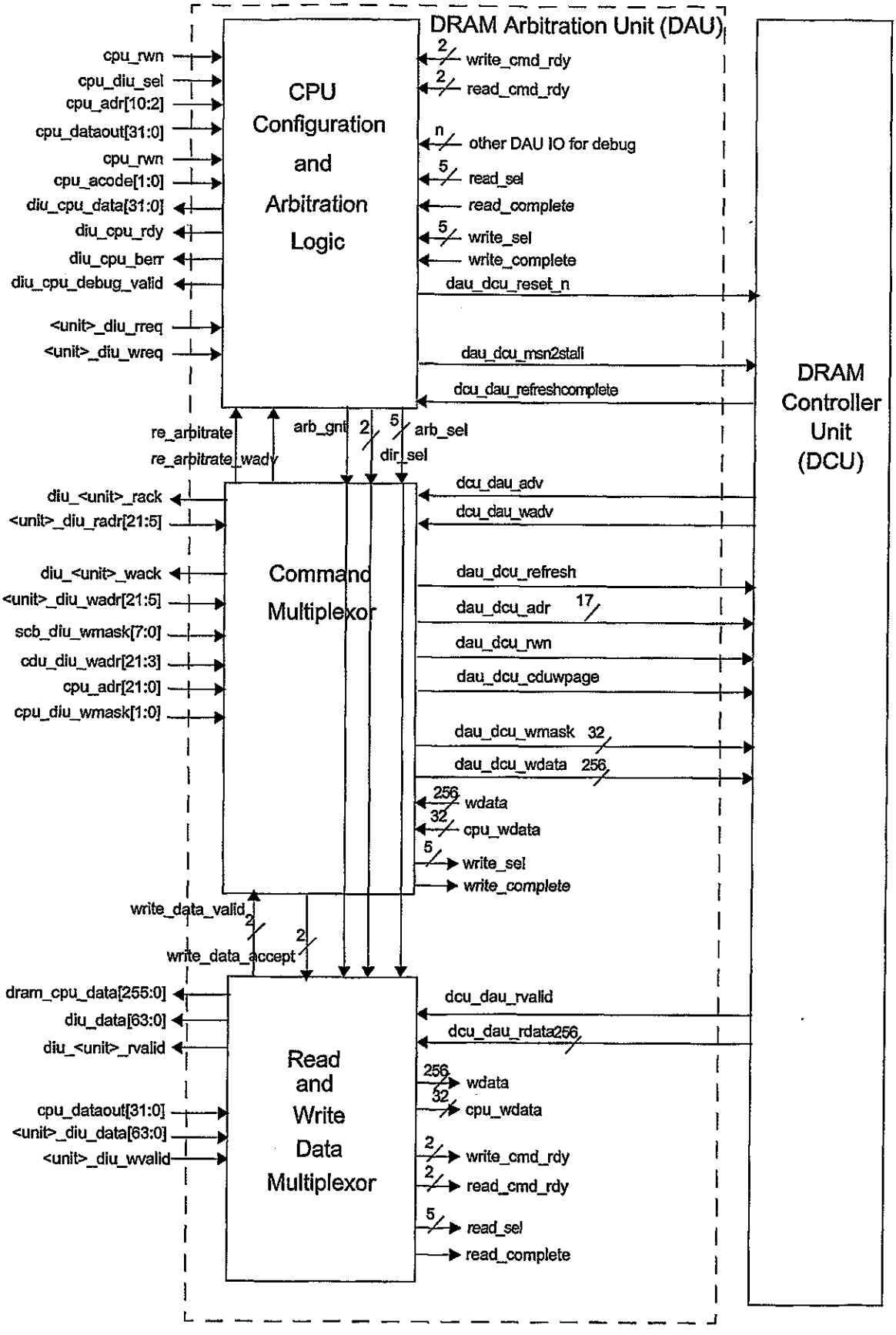

DRAM Arbitration Unit (DAU)
CPU Configuration and Arbitration Logic
Command Multiplexor
Read and Write Data Multiplexor
DRAM Controller Unit (DCU)
cpu_rwn
cpu_diu_sel
cpu_adr[10:2]
cpu_dataout[31:0]
cpu_rwn
cpu_acode[1:0]
diu_cpu_data[31:0]
diu_cpu_rdy
diu_cpu_berr
diu_cpu_debug_valid
<unit>_diu_rreq
<unit>_diu_wreq
write_cmd_rdy
read_cmd_rdy
other DAU IO for debug
read_sel
read_complete
write_sel
write_complete
dau_dcu_reset_n
dau_dcu_msn2stall
dcu_dau_refreshcomplete
re_arbitrate
re_arbitrate_wadv
arb_gnt
arb_sel
dir_sel
diu_<unit>_rack
<unit>_diu_radr[21:5]
diu_<unit>_wack
<unit>_diu_wadr[21:5]
scb_diu_wmask[7:0]
cdu_diu_wadr[21:3]
cpu_adr[21:0]
cpu_diu_wmask[1:0]
dcu_dau_adv
dcu_dau_wadv
dau_dcu_refresh
dau_dcu_adr
dau_dcu_rwn
dau_dcu_cduwpage
dau_dcu_wmask
dau_dcu_wdata
wdata
cpu_wdata
write_sel
write_complete
write_data_valid
write_data_accept
dram_cpu_data[255:0]
diu_data[63:0]
diu_<unit>_rvalid
cpu_dataout[31:0]
<unit>_diu_data[63:0]
<unit>_diu_wvalid
dcu_dau_rvalid
dcu_dau_rdata256
wdata
cpu_wdata
write_cmd_rdy
read_cmd_rdy
read_sel
read_complete

FIG. 101

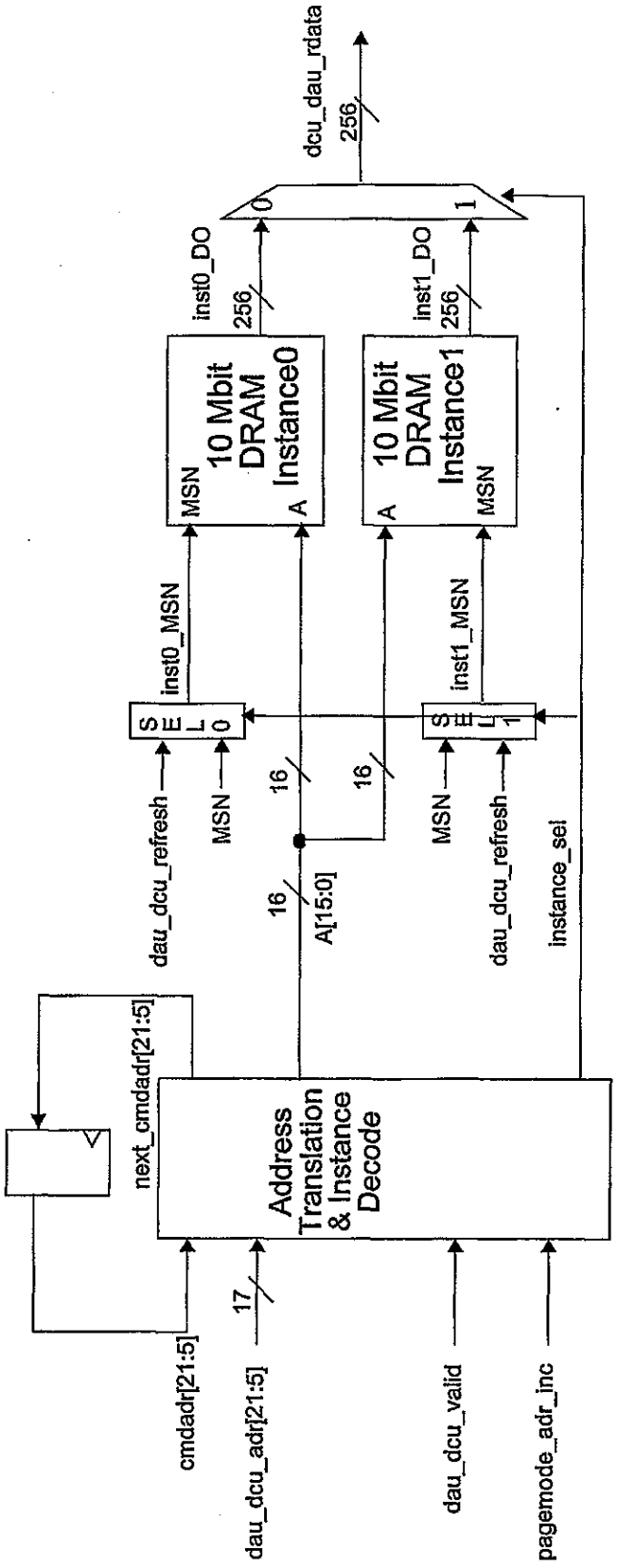

dcu_dau_rdata
256
0
1
inst0_DO
inst1_DO
256
256
10 Mbit DRAM Instance0
10 Mbit DRAM Instance1
MSN
A
A
MSN
inst0_MSN
inst1_MSN
SEL 0
SEL 1
dau_dcu_refresh
MSN
16
16
16
A[15:0]
MSN
dau_dcu_refresh
instance_sel
next_cmdadr[21:5]
Address Translation & Instance Decode
17
cmdadr[21:5]
dau_dcu_adr[21:5]
dau_dcu_valid
pagemode_adr_inc

FIG. 102

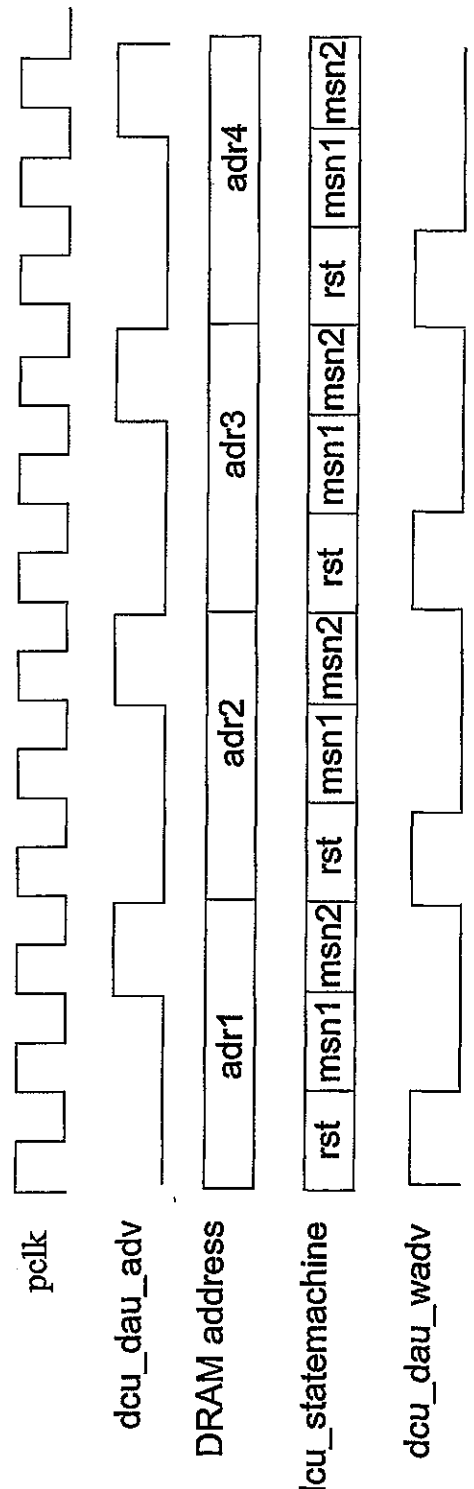

pclk
dcu_dau_adv
DRAM address
dcu_statemachine
dcu_dau_wadv
adr1
adr2
adr3
adr4
rst
msn1
msn2
rst
msn1
msn2
rst
msn1
msn2
rst
msn1
msn2

FIG. 103

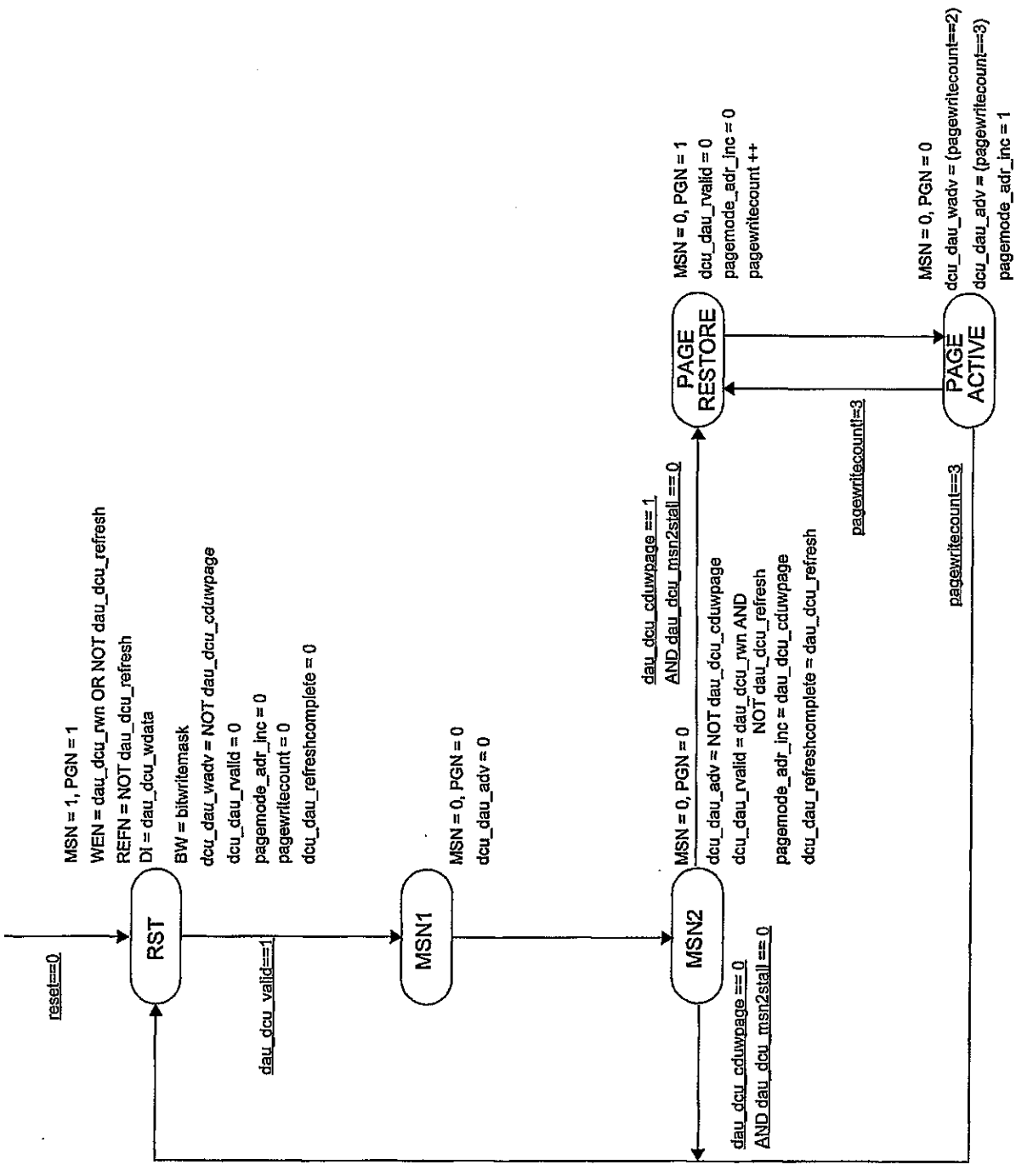

reset==0
RST
MSN = 1, PGN = 1
WEN = dau_dcu_rwn OR NOT dau_dcu_refresh
REFN = NOT dau_dcu_refresh
DI = dau_dcu_wdata
BW = bitwritemask
dcu_dau_wadv = NOT dau_dcu_cduwpage
dcu_dau_rvalid = 0
pagemode_adr_inc = 0
pagewritecount = 0
dcu_dau_refreshcomplete = 0
dau_dcu_valid==1
MSN1
MSN = 0, PGN = 0
dcu_dau_adv = 0
dau_dcu_cduwpage == 1
AND dau_dcu_msn2stall == 0
MSN2
MSN = 0, PGN = 0
dcu_dau_adv = NOT dau_dcu_cduwpage
dcu_dau_rvalid = dau_dcu_rwn AND
NOT dau_dcu_refresh
pagemode_adr_inc = dau_dcu_cduwpage
dcu_dau_refreshcomplete = dau_dcu_refresh
dau_dcu_cduwpage == 0
AND dau_dcu_msn2stall == 0
PAGE RESTORE
MSN = 0, PGN = 1
dcu_dau_rvalid = 0
pagemode_adr_inc = 0
pagewritecount ++
pagewritecount!=3
PAGE ACTIVE
MSN = 0, PGN = 0
dcu_dau_wadv = (pagewritecount==2)
dcu_dau_adv = (pagewritecount==3)
pagemode_adr_inc = 1
pagewritecount==3

FIG. 104

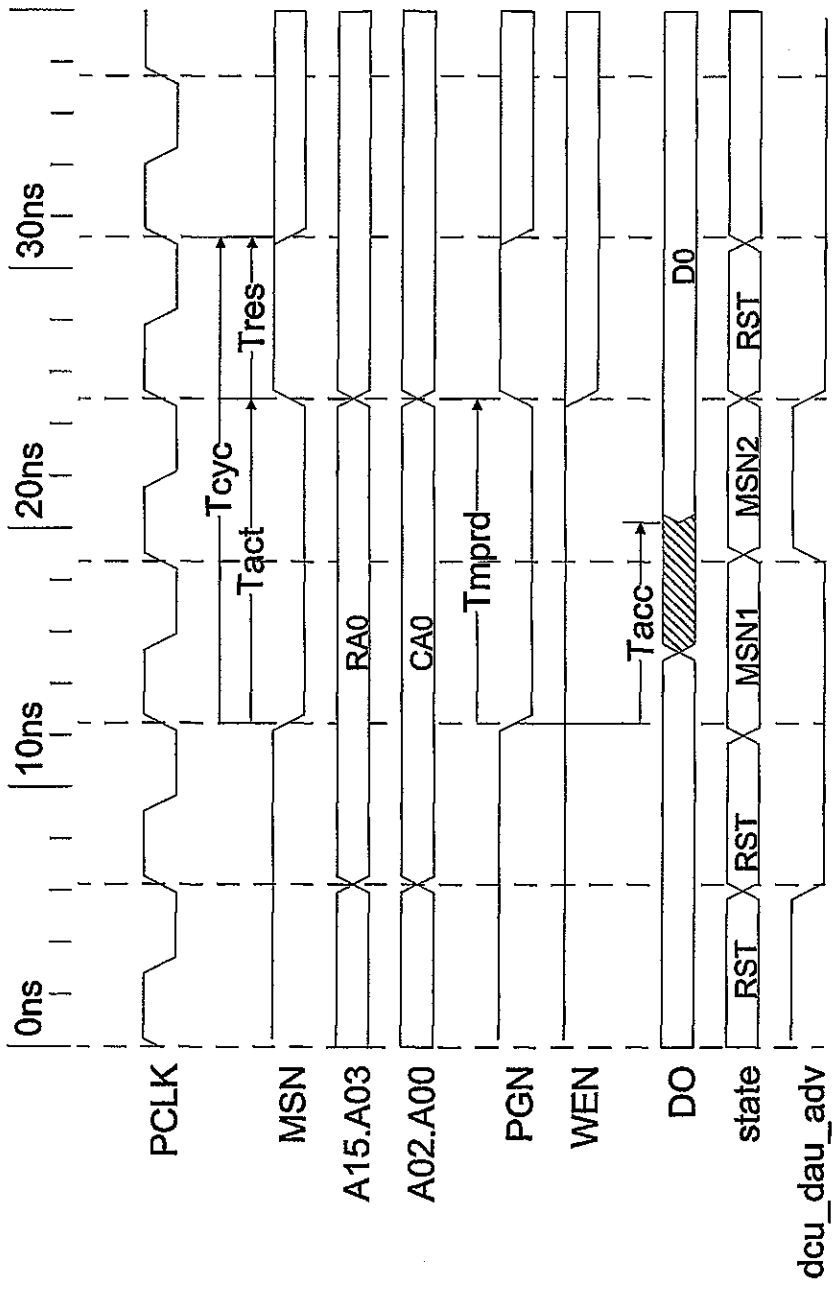

0ns
10ns
20ns
30ns
Tcyc
Tact
Tres
Tmprd
Tacc
RA0
CA0
D0
RST
MSN1
MSN2
RST
RST
PCLK
MSN
A15.A03
A02.A00
PGN
WEN
DO
state
dcu_dau_adv

FIG. 105

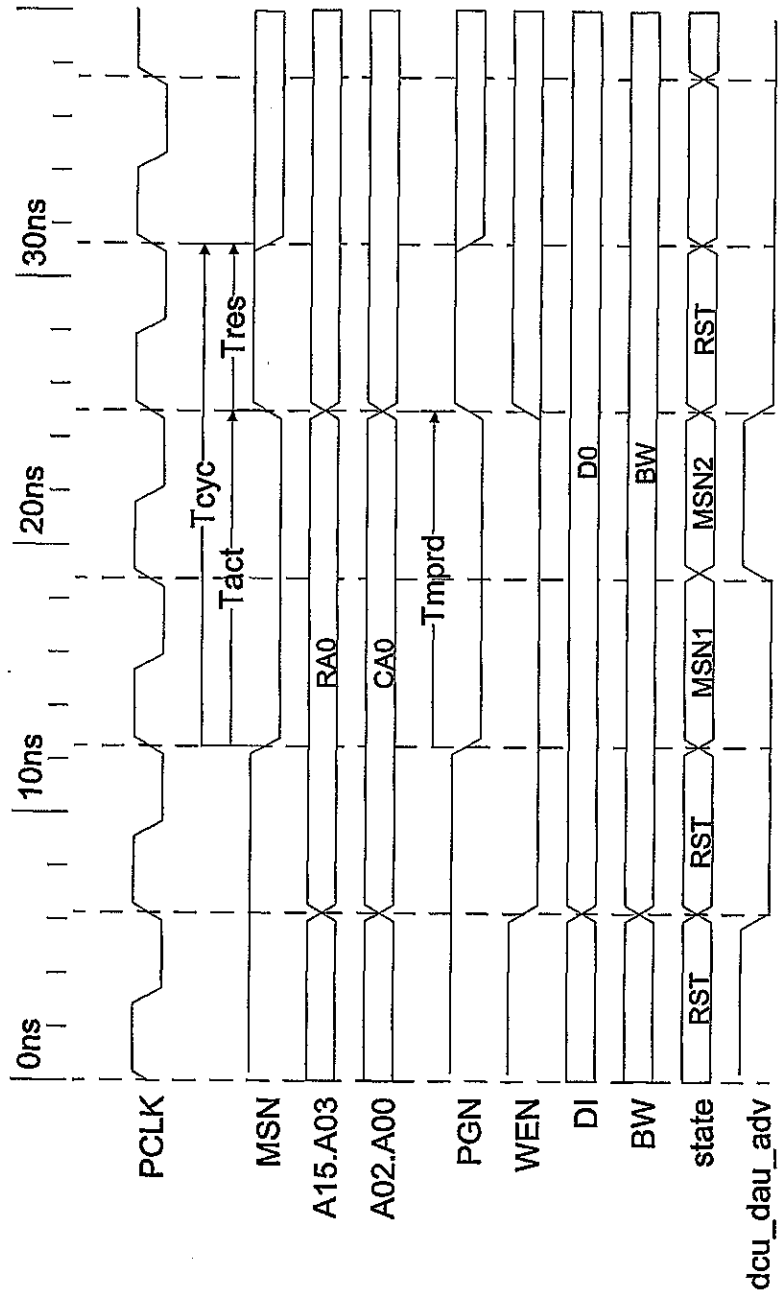

0ns
10ns
20ns
30ns
Tcyc
Tres
Tact
Tmprd
PCLK
MSN
A15.A03
A02.A00
PGN
WEN
DI
BW
state
dcu_dau_adv
RA0
CA0
D0
BW
RST
MSN1
MSN2
RST
RST

FIG. 106

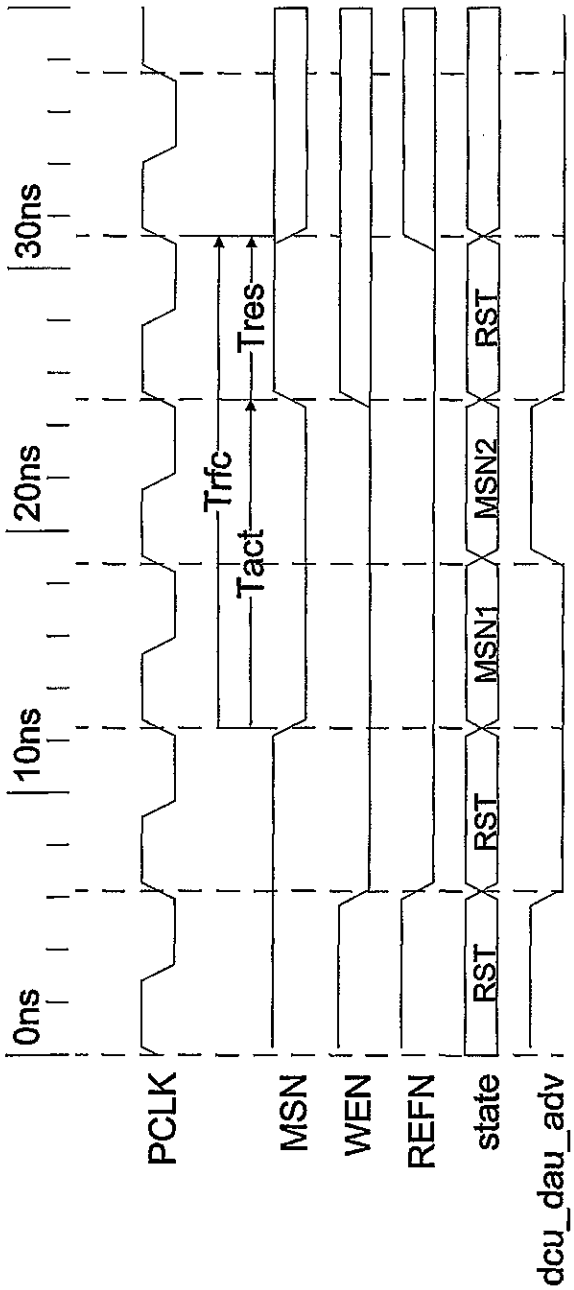

0ns
10ns
20ns
30ns
Trfc
Tact
Tres
PCLK
MSN
WEN
REFN
state
dcu_dau_adv
RST
RST
MSN1
MSN2
RST

FIG. 107

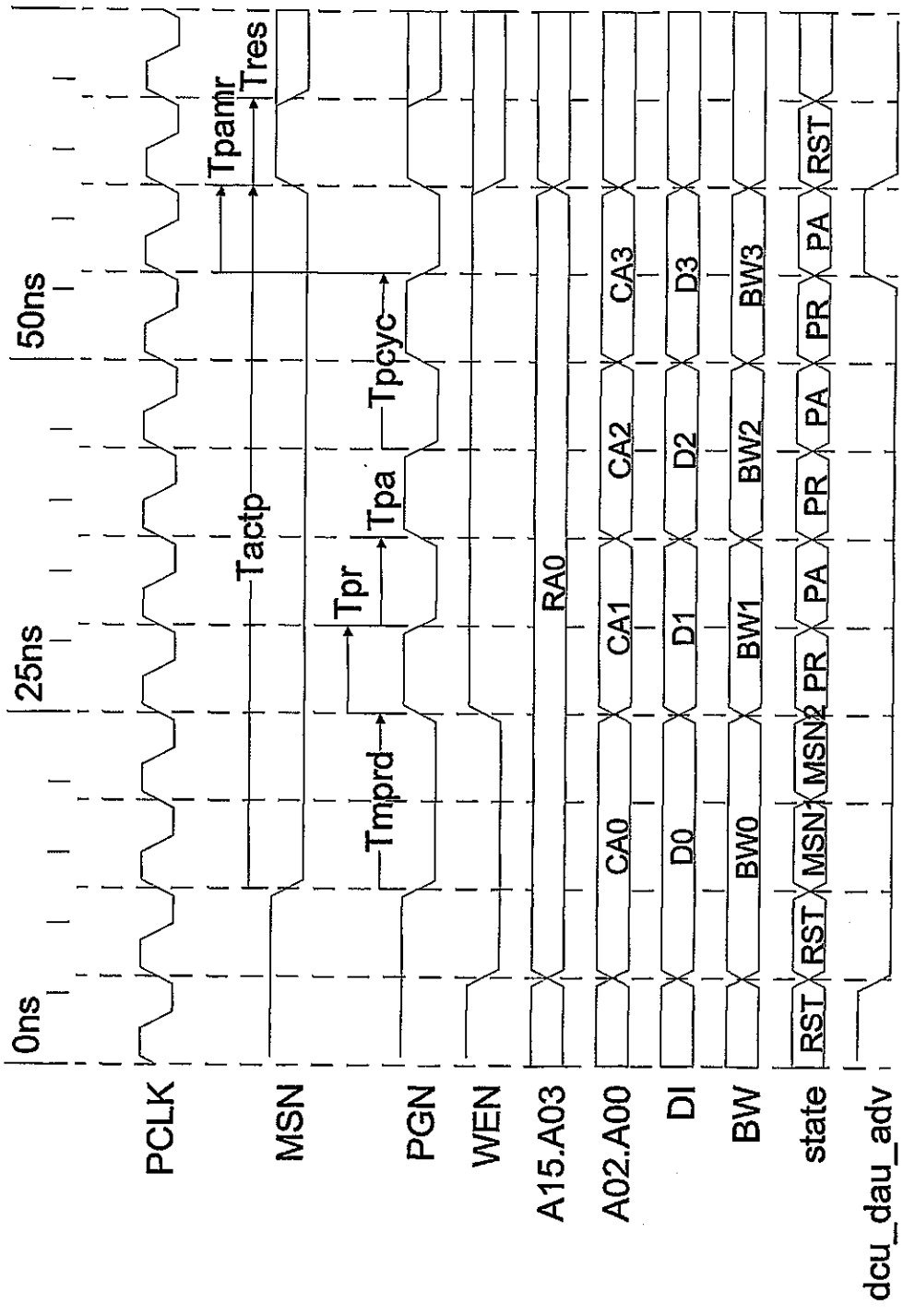

0ns
25ns
50ns
Tpamr
Tres
Tactp
Tpr
Tpa
Tpcyc
Tmprd
PCLK
MSN
PGN
WEN
A15.A03
A02.A00
DI
BW
state
dcu_dau_adv
RA0
CA0
CA1
CA2
CA3
D0
D1
D2
D3
BW0
BW1
BW2
BW3
RST
MSN1
MSN2
PR
PA

FIG. 108

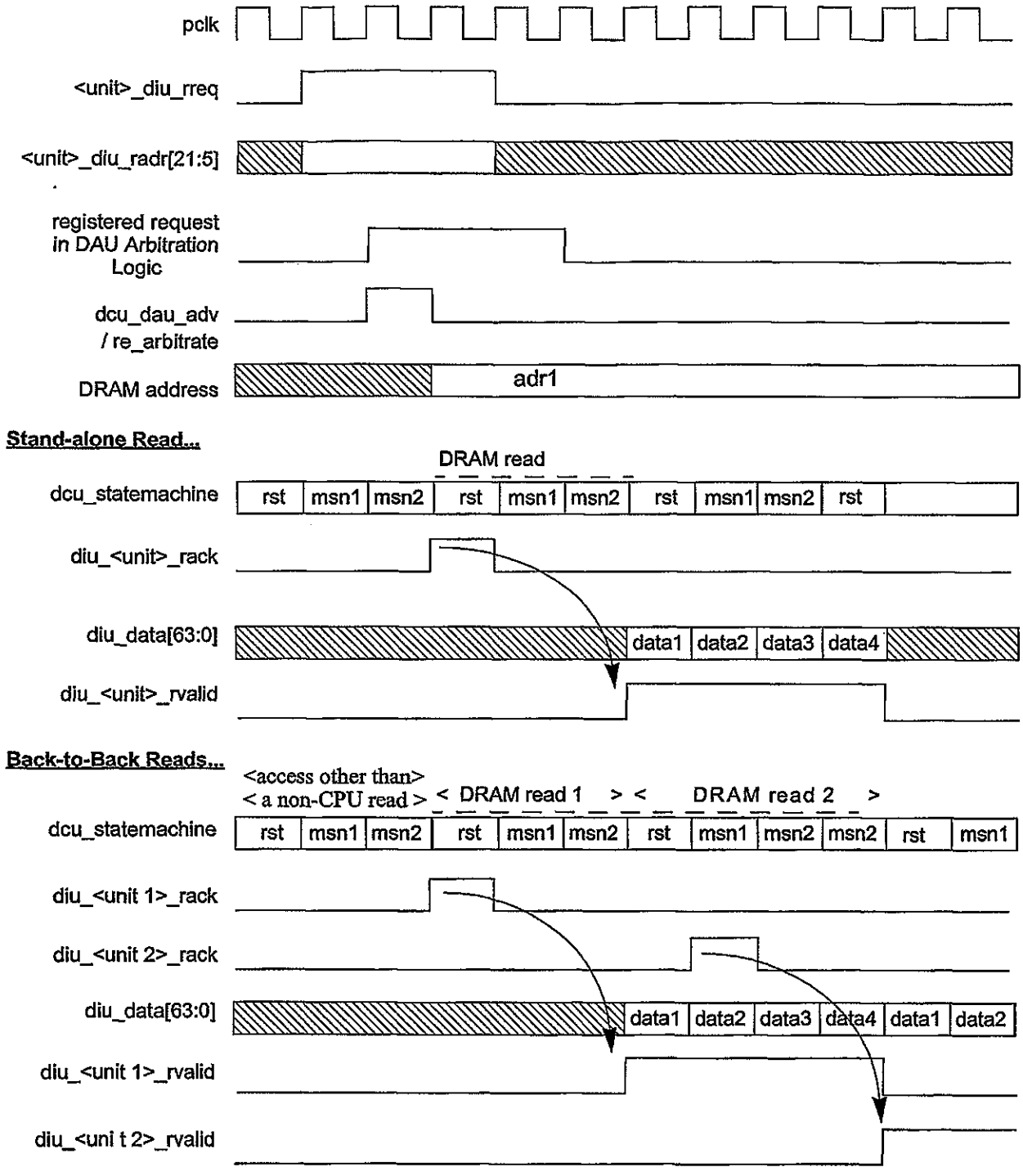

pclk
<unit>_diu_rreq
<unit>_diu_radr[21:5]
registered request in DAU Arbitration Logic
dcu_dau_adv / re_arbitrate
DRAM address
adr1
Stand-alone Read...
DRAM read
dcu_statemachine
rst msn1 msn2 rst msn1 msn2 rst msn1 msn2 rst
diu_<unit>_rack
diu_data[63:0]
data1 data2 data3 data4
diu_<unit>_rvalid
Back-to-Back Reads...
<access other than> < a non-CPU read >
< DRAM read 1 > < DRAM read 2 >
dcu_statemachine
rst msn1 msn2 rst msn1 msn2 rst msn1 msn2 msn2 rst msn1
diu_<unit 1>_rack
diu_<unit 2>_rack
diu_data[63:0]
data1 data2 data3 data4 data1 data2
diu_<unit 1>_rvalid
diu_<uni t 2>_rvalid

FIG. 109

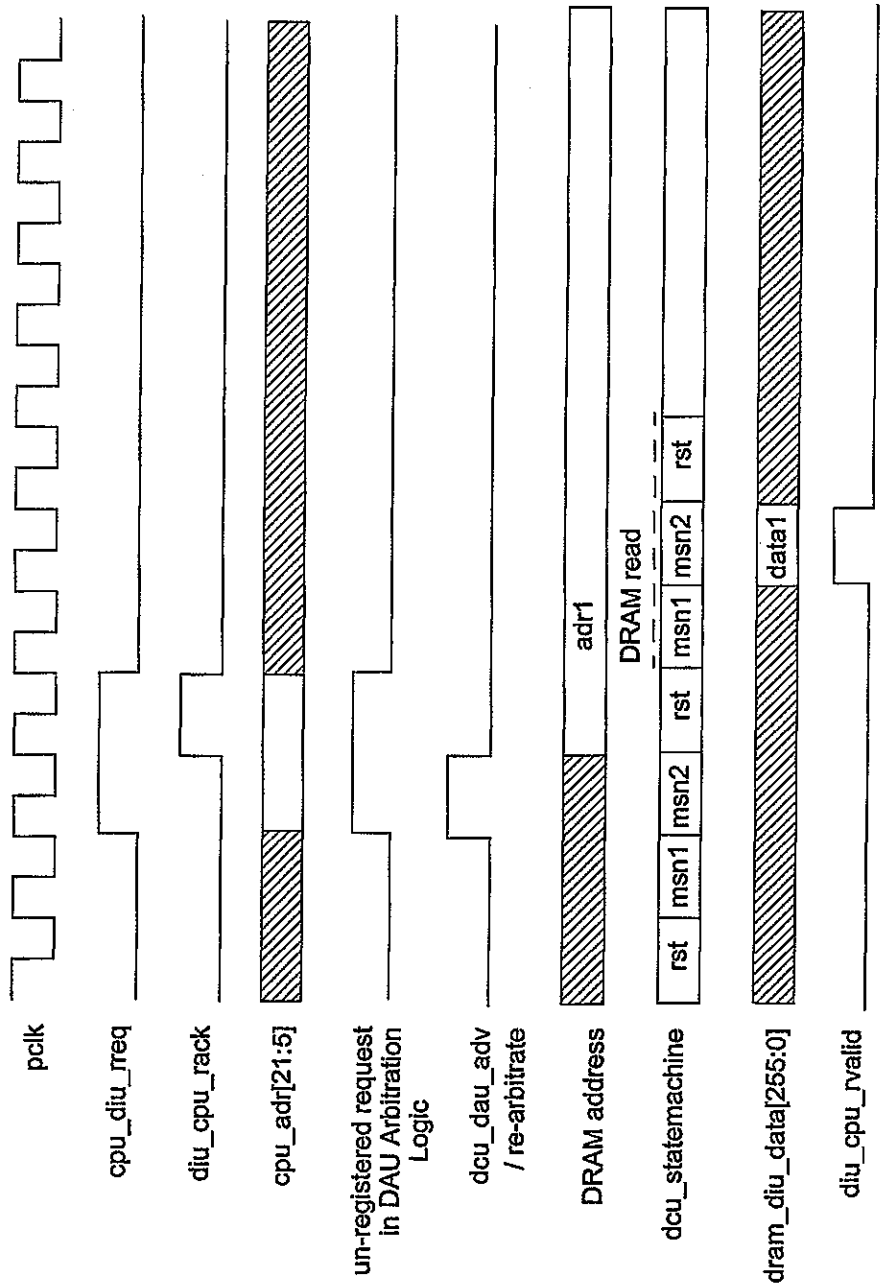

pclk
cpu_diu_rreq
diu_cpu_rack
cpu_adr[21:5]
un-registered request in DAU Arbitration Logic
dcu_dau_adv / re-arbitrate
DRAM address
adr1
DRAM read
dcu_statemachine
rst
msn1
msn2
rst
msn1
msn2
rst
dram_diu_data[255:0]
data1
diu_cpu_rvalid

FIG. 110

CPU MMU logic delay
CPU generates request
Arbitration occurs
cycle 1
DCU Address setup cycle (RST)
cycle 2
DRAM access (MSN1)
Tacc = 8 ns
cycle 3
DRAM access (MSN2)
cycle 4
CPU captures read data in AHB bridge
cycle 5
CPU MMU logic delay
CPU generates request
Arbitration occurs
cycle 6

FIG. 111

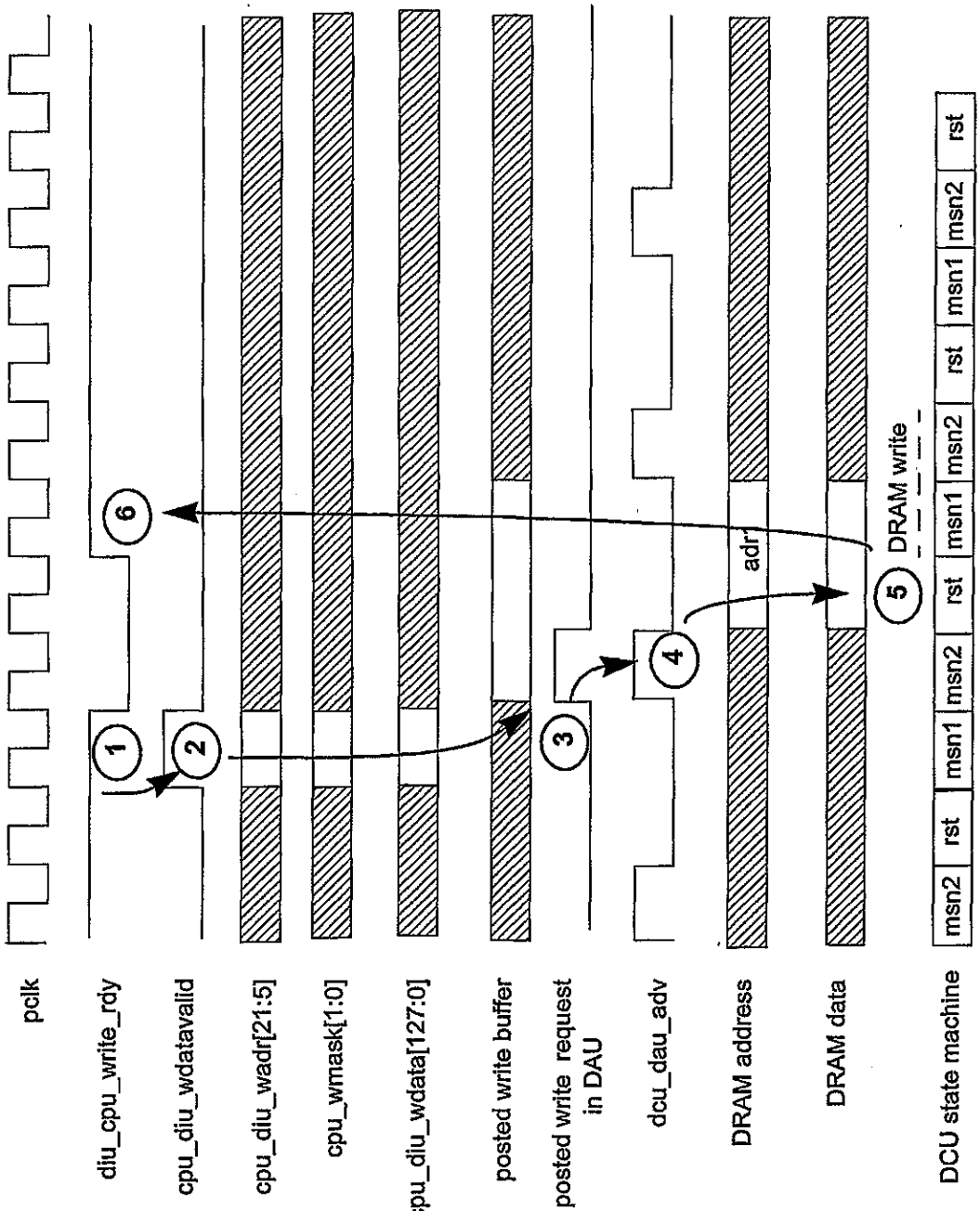

pclk
diu_cpu_write_rdy
cpu_diu_wdatavalid
cpu_diu_wadr[21:5]
cpu_wmask[1:0]
cpu_diu_wdata[127:0]
posted write buffer
posted write request in DAU
dcu_dau_adv
DRAM address
DRAM data
DCU state machine
adr1
DRAM write
msn2
rst
msn1
msn2
rst
msn1
msn2
rst
msn1
msn2
rst
1
2
3
4
5
6

FIG. 112

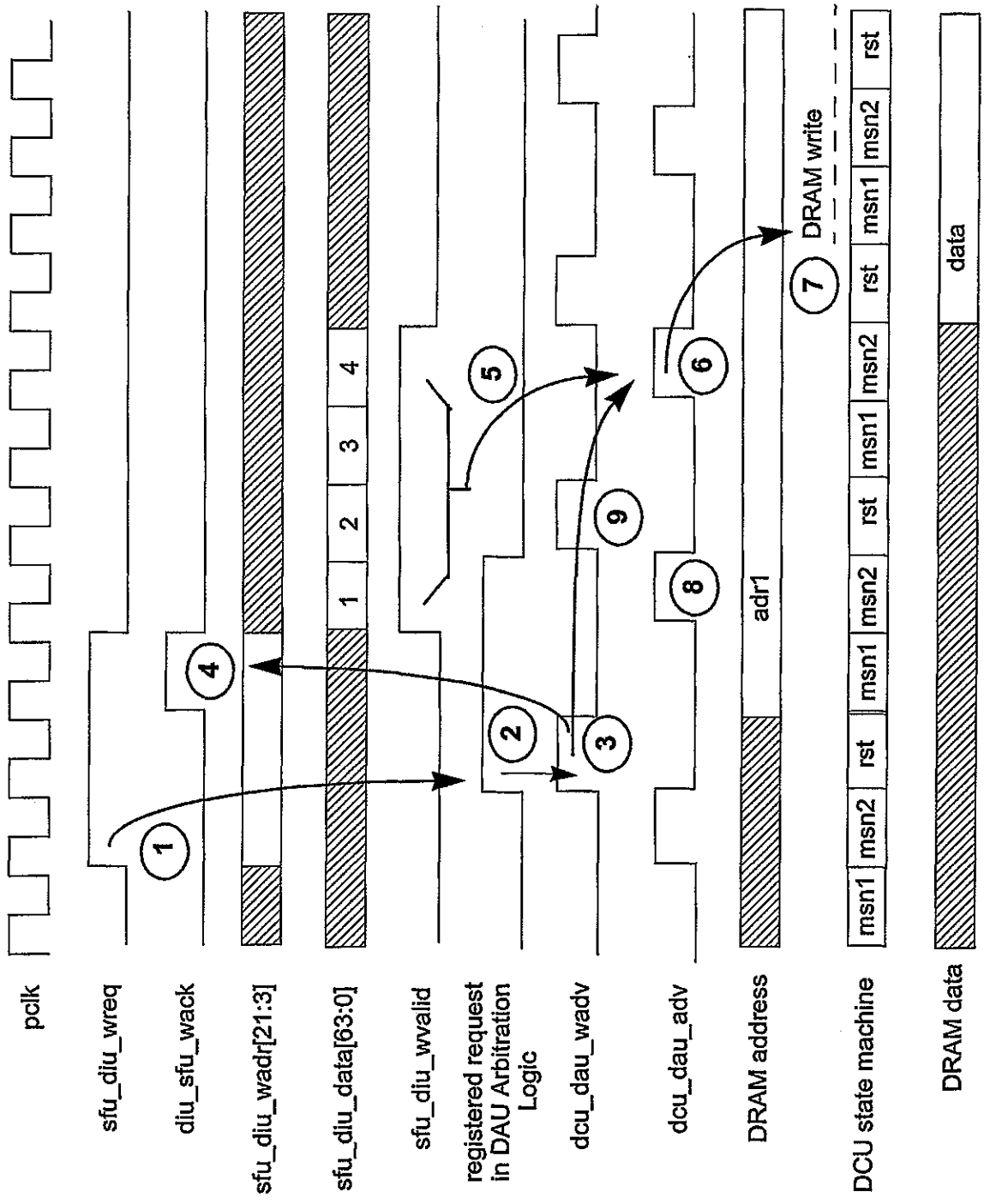

pclk
sfu_diu_wreq
diu_sfu_wack
sfu_diu_wadr[21:3]
sfu_diu_data[63:0]
sfu_diu_wvalid
registered request in DAU Arbitration Logic
dcu_dau_wadv
dcu_dau_adv
DRAM address
DCU state machine
DRAM data
1
2
3
4
5
6
7
8
9
adr1
DRAM write
msn1
msn2
rst
data

FIG. 113

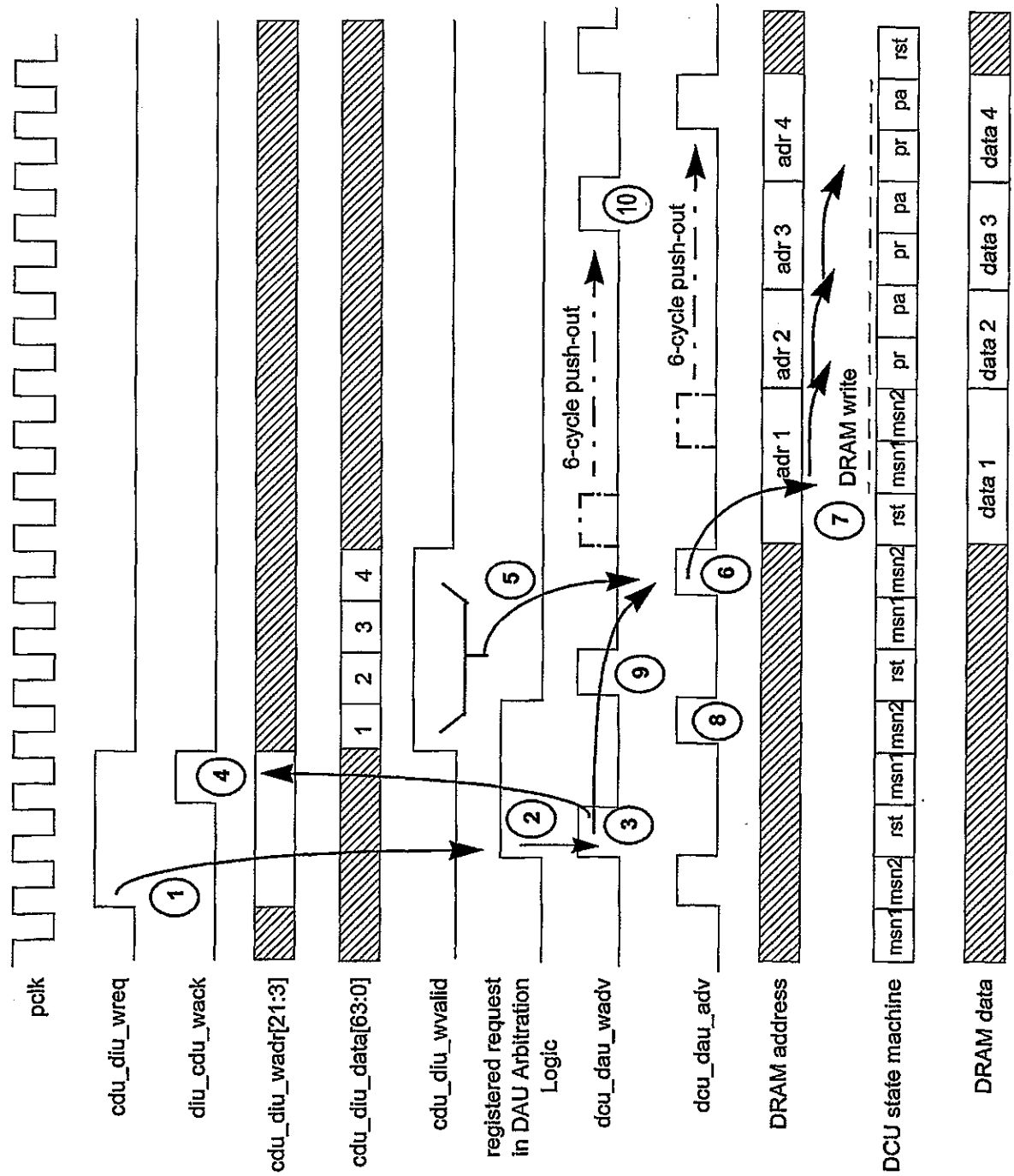

pclk
cdu_diu_wreq
diu_cdu_wack
cdu_diu_wadr[21:3]
cdu_diu_data[63:0]
cdu_diu_wvalid
registered request in DAU Arbitration Logic
dcu_dau_wadv
dcu_dau_adv
DRAM address
DCU state machine
DRAM data
1
2
3
4
5
6
7
8
9
10
6-cycle push-out
6-cycle push-out
DRAM write
rst
adr 1
adr 2
adr 3
adr 4
data 1
data 2
data 3
data 4
msn1
msn2
pr
pa

FIG. 114

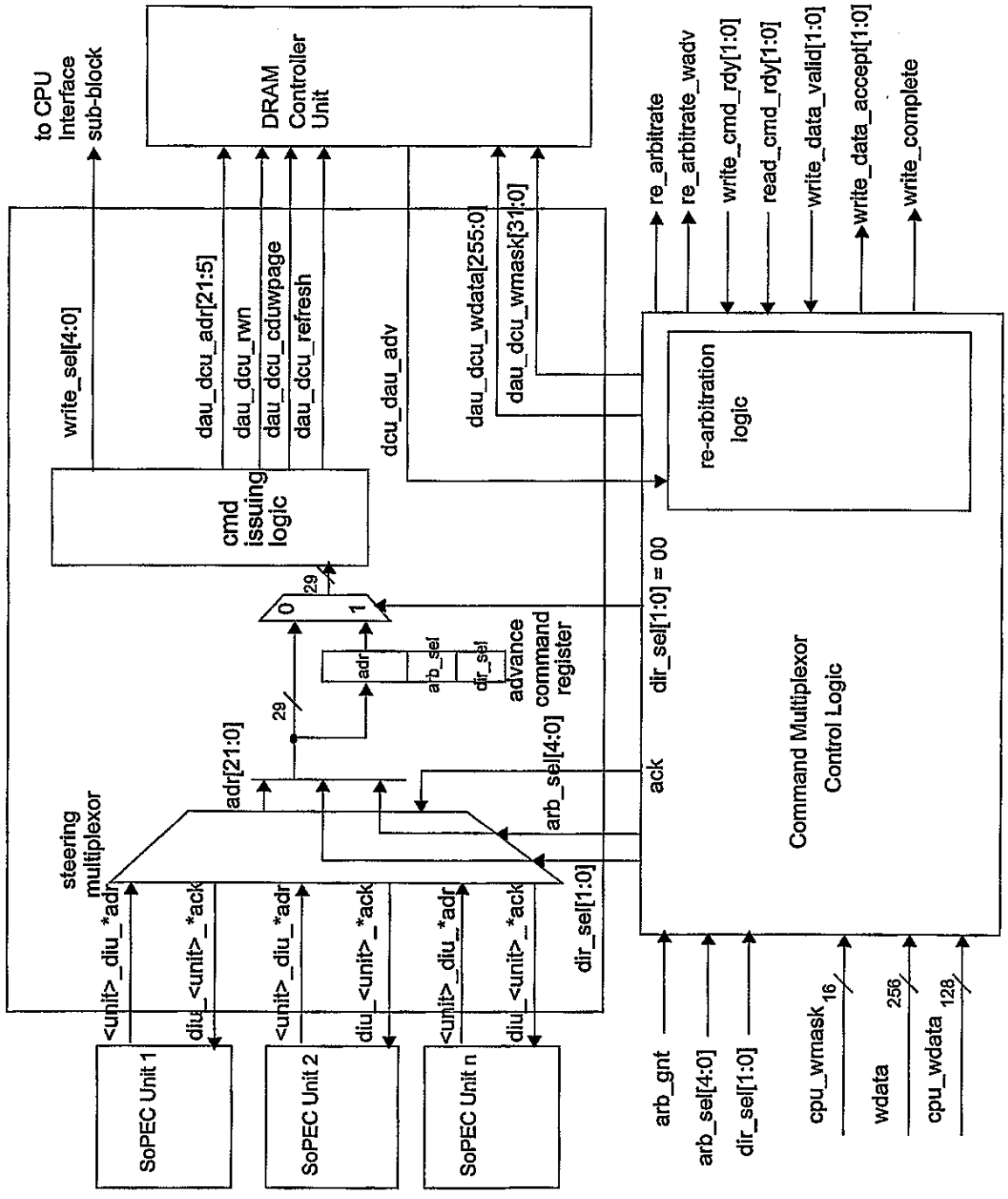

to CPU Interface sub-block
DRAM Controller Unit
write_sel[4:0]
dau_dcu_adr[21:5]
dau_dcu_rwn
dau_dcu_cduwpage
dau_dcu_refresh
dcu_dau_adv
dau_dcu_wdata[255:0]
dau_dcu_wmask[31:0]
re_arbitrate
re_arbitrate_wadv
write_cmd_rdy[1:0]
read_cmd_rdy[1:0]
write_data_valid[1:0]
write_data_accept[1:0]
write_complete
cmd issuing logic
re-arbitration logic
29
0
1
adr
arb_sel
dir_sel
advance command register
dir_sel[1:0] = 00
adr[21:0]
arb_sel[4:0]
ack
steering multiplexor
Command Multiplexor Control Logic
<unit>_diu_*adr
diu_<unit>_*ack
dir_sel[1:0]
SoPEC Unit 1
SoPEC Unit 2
SoPEC Unit n
arb_gnt
arb_sel[4:0]
dir_sel[1:0]
cpu_wmask
16
wdata
256
cpu_wdata
128

FIG. 115

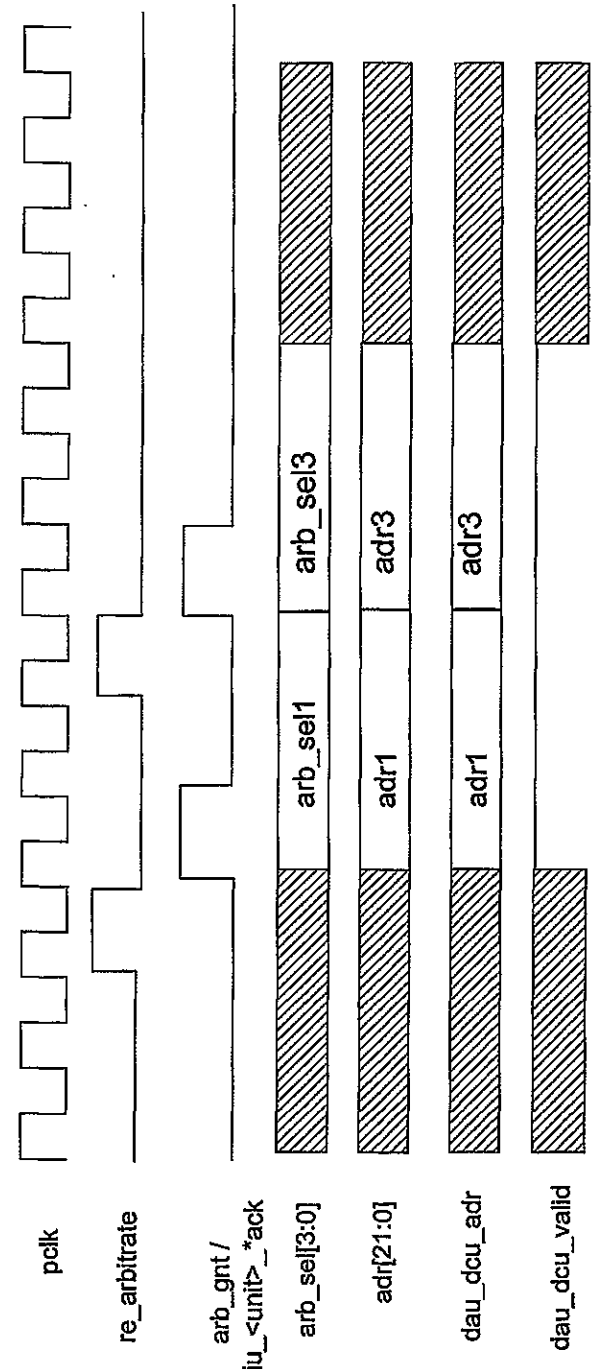

pclk
re_arbitrate
arb_gnt / diu_<unit>_*ack
arb_sel[3:0]
adr[21:0]
dau_dcu_adr
dau_dcu_valid
arb_sel1
arb_sel3
adr1
adr3
adr1
adr3

FIG. 116

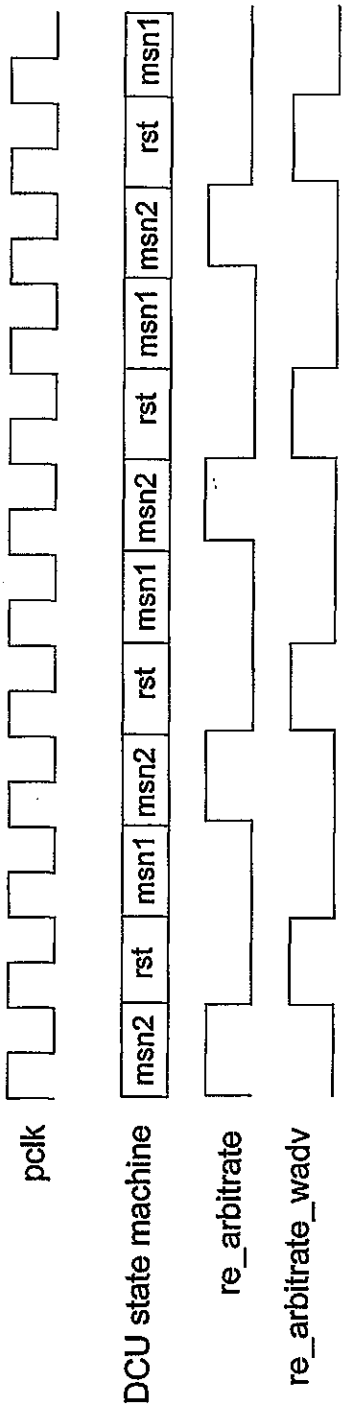

pclk
DCU state machine
re_arbitrate
re_arbitrate_wadv
msn2
rst
msn1
msn2
rst
msn1
msn2
rst
msn1
msn2
rst
msn1

FIG. 117

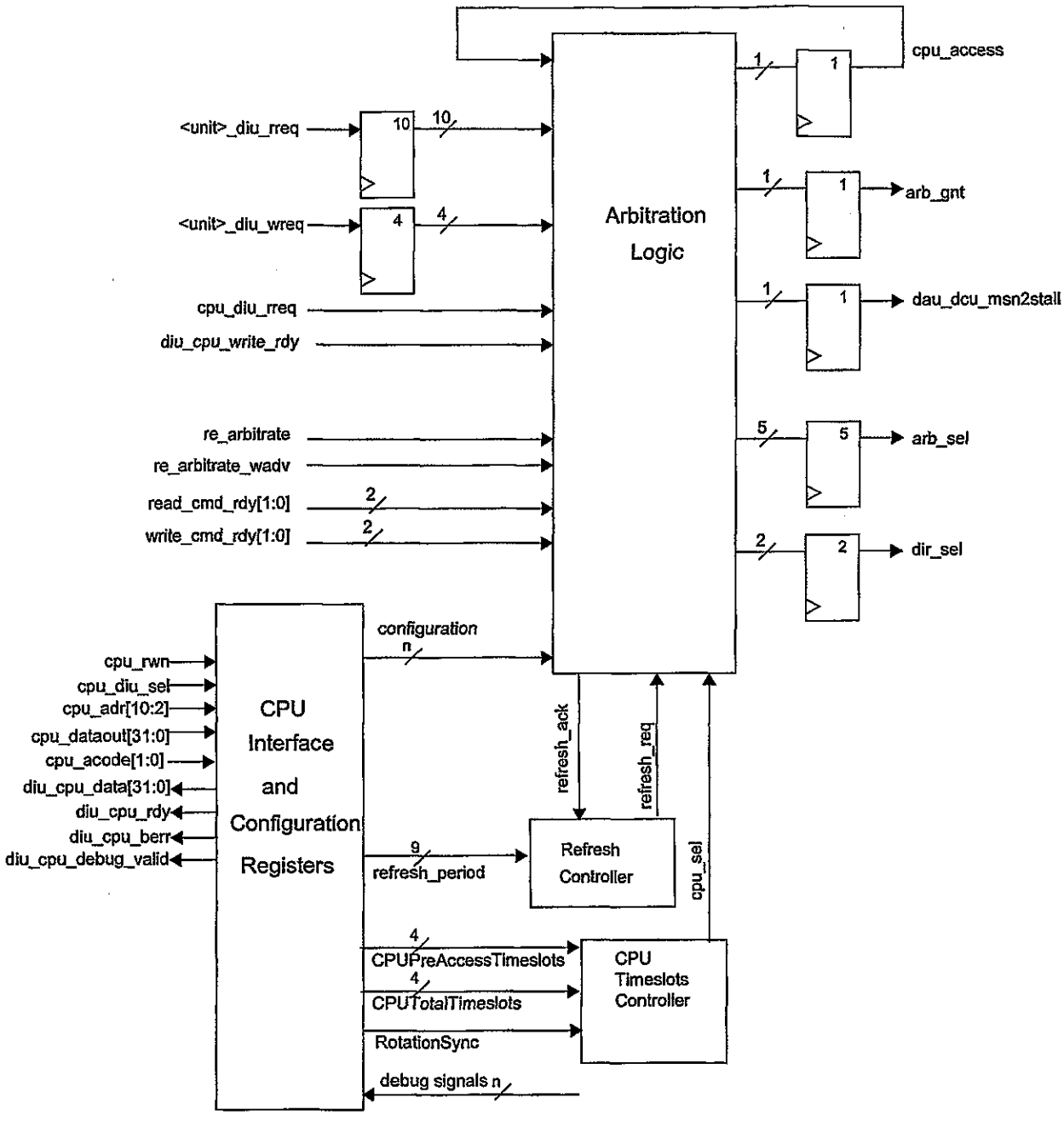

cpu_access
<unit>_diu_rreq
10
<unit>_diu_wreq
4
Arbitration Logic
arb_gnt
dau_dcu_msn2stall
cpu_diu_rreq
diu_cpu_write_rdy
re_arbitrate
re_arbitrate_wadv
read_cmd_rdy[1:0]
write_cmd_rdy[1:0]
arb_sel
dir_sel
configuration
n
cpu_rwn
cpu_diu_sel
cpu_adr[10:2]
cpu_dataout[31:0]
cpu_acode[1:0]
diu_cpu_data[31:0]
diu_cpu_rdy
diu_cpu_berr
diu_cpu_debug_valid
CPU Interface and Configuration Registers
refresh_ack
refresh_req
cpu_sel
refresh_period
Refresh Controller
CPUPreAccessTimeslots
CPUTotalTimeslots
RotationSync
CPU Timeslots Controller
debug signals n

FIG. 118

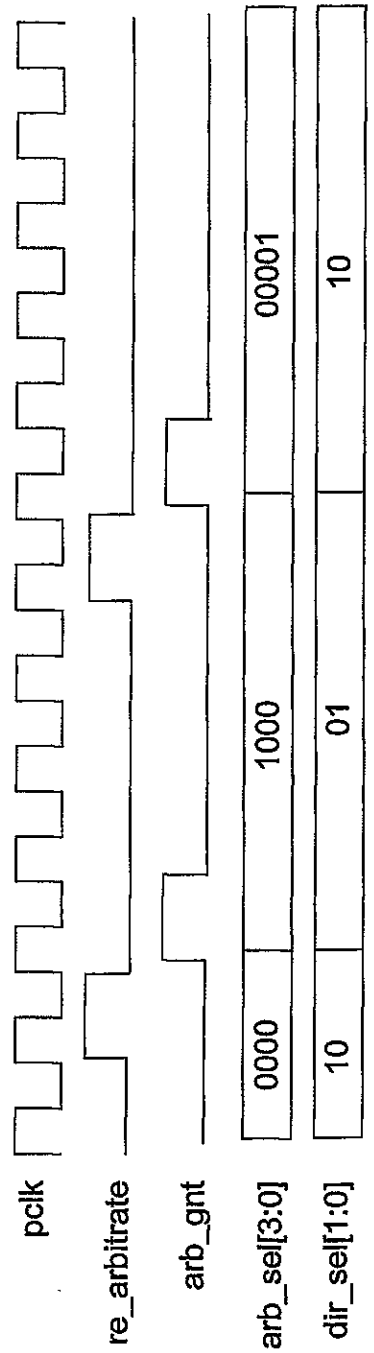

pclk
re_arbitrate
arb_gnt
arb_sel[3:0]
dir_sel[1:0]
0000
1000
00001
10
01
10

FIG. 119

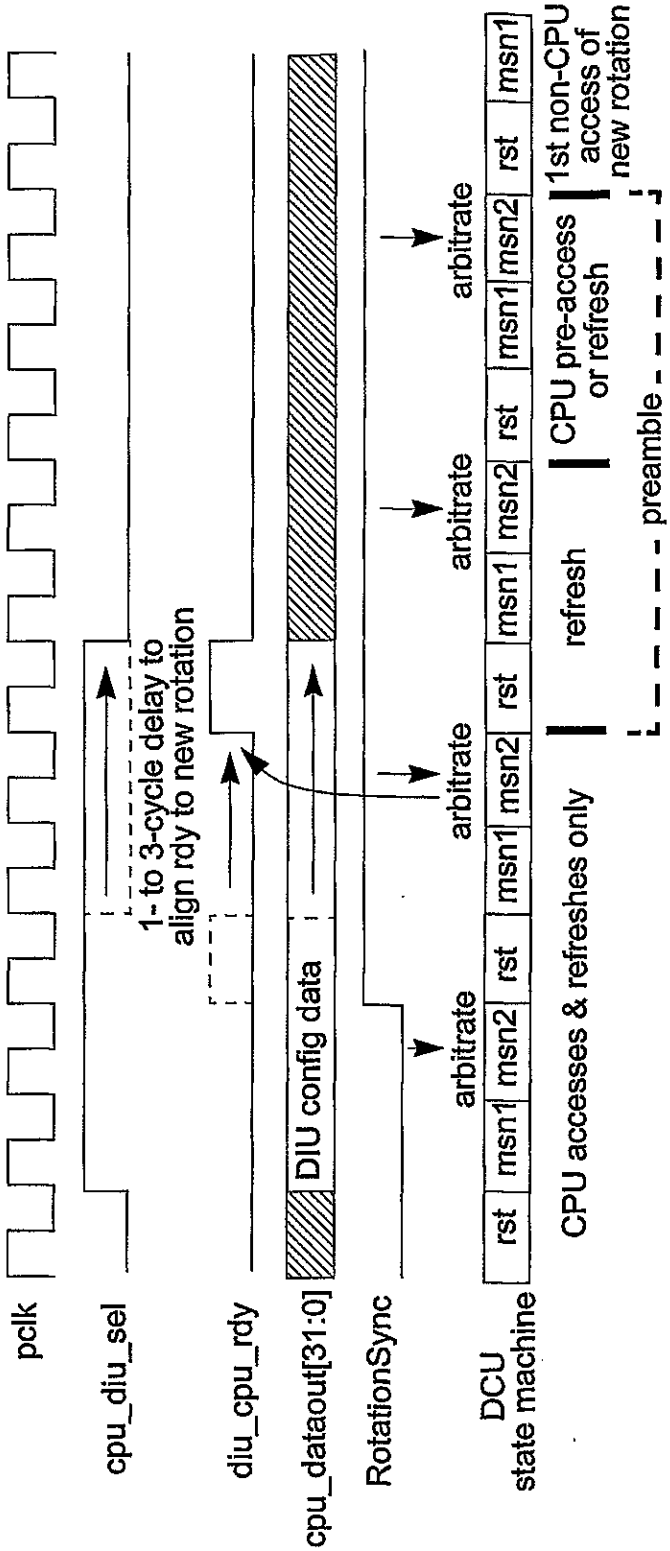

pclk
cpu_diu_sel
1- to 3-cycle delay to
align rdy to new rotation
diu_cpu_rdy
cpu_dataout[31:0]
DIU config data
RotationSync
arbitrate
arbitrate
arbitrate
DCU
state machine
rst
msn1
msn2
rst
msn1
msn2
rst
msn1
msn2
rst
msn1
msn2
rst
msn1
CPU accesses & refreshes only
refresh
CPU pre-access
or refresh
1st non-CPU
access of
new rotation
preamble

FIG. 120

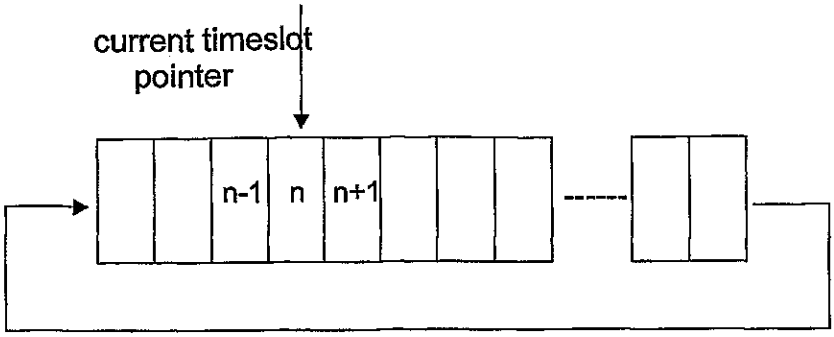


FIG. 121

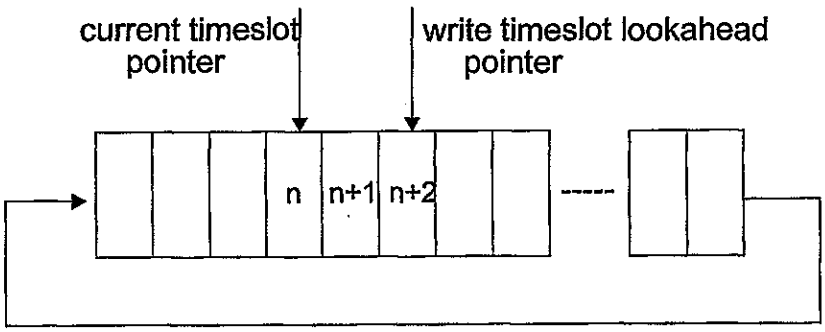


FIG. 122

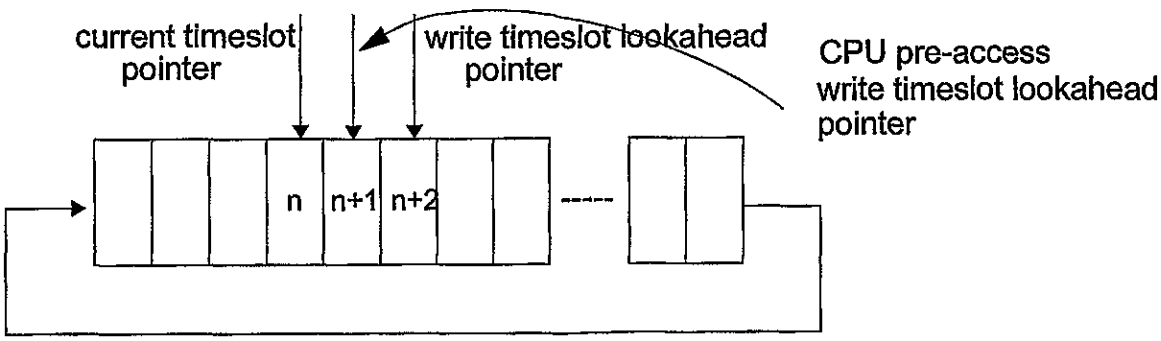


FIG. 123

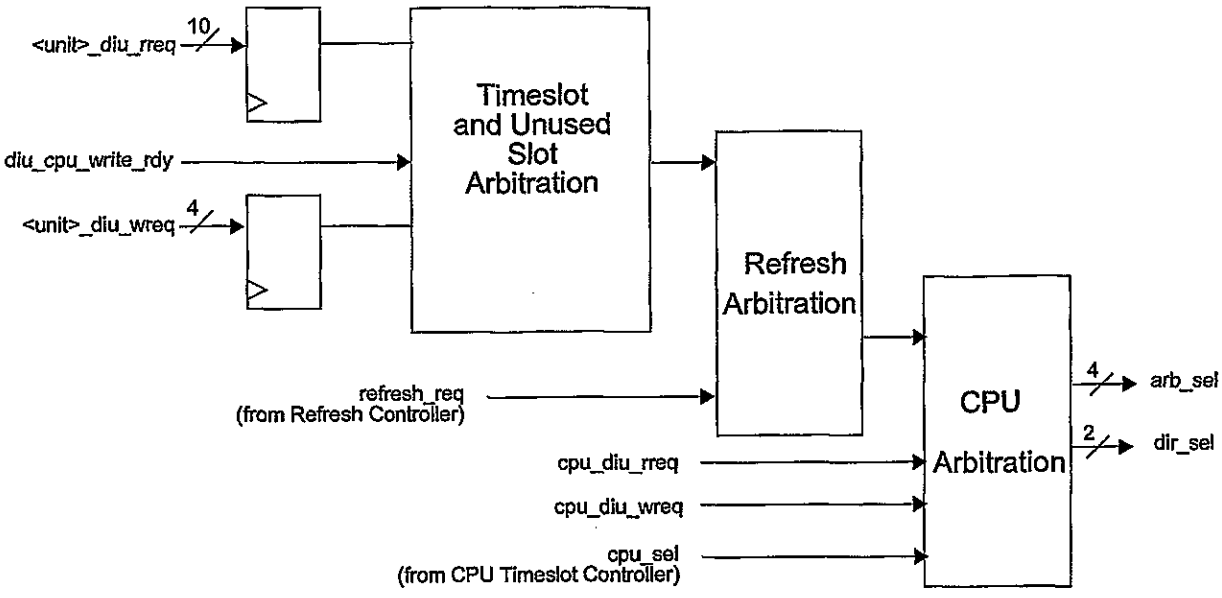


FIG. 124

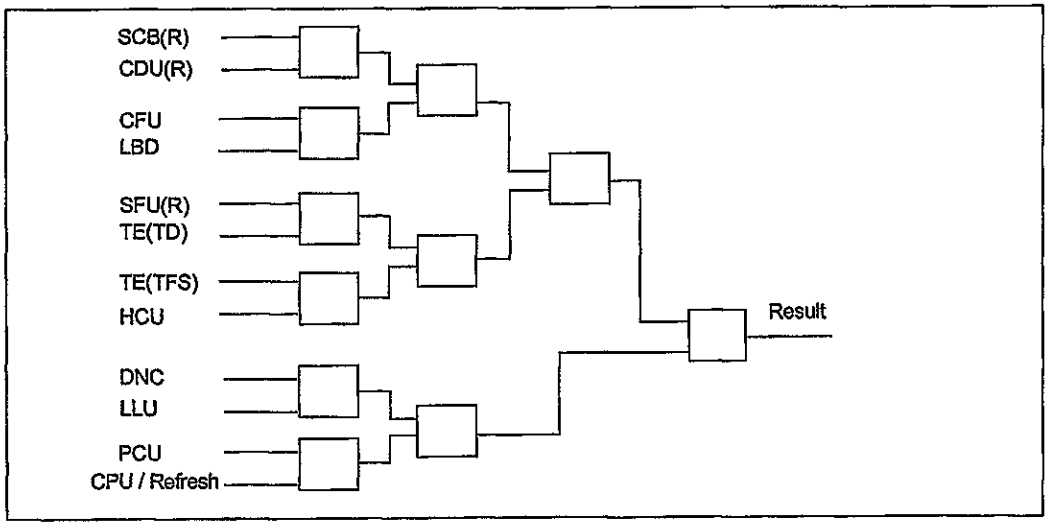


FIG. 125

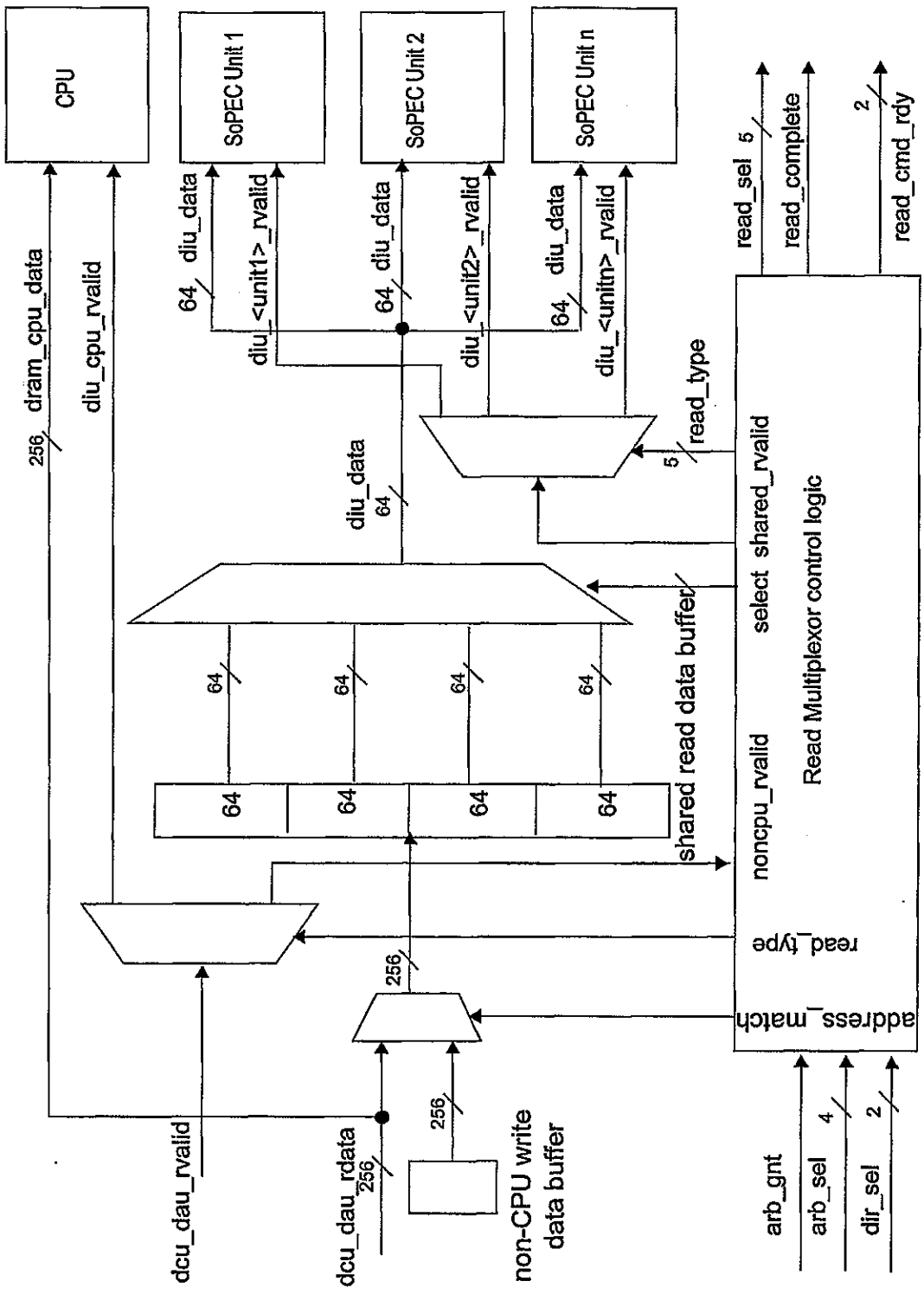

CPU
SoPEC Unit 1
SoPEC Unit 2
SoPEC Unit n
dram_cpu_data
256
diu_cpu_rvalid
64
diu_data
diu_<unit1>_rvalid
diu_<unit2>_rvalid
diu_<unitn>_rvalid
read_sel
5
read_complete
read_cmd_rdy
2
read_type
shared read data buffer
noncpu_rvalid
select
shared_rvalid
Read Multiplexor control logic
read_type
address_match
dcu_dau_rvalid
dcu_dau_rdata
non-CPU write
data buffer
arb_gnt
arb_sel
4
dir_sel

FIG. 126

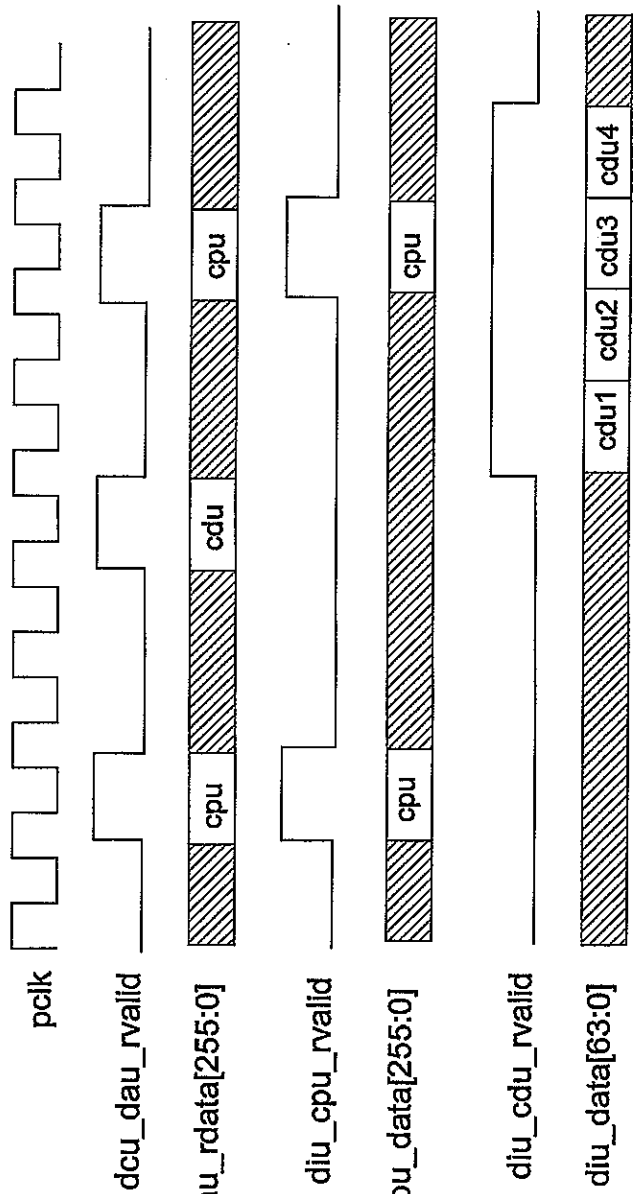

pclk
dcu_dau_rvalid
dcu_dau_rdata[255:0]
cpu
cdu
cpu
diu_cpu_rvalid
dram_cpu_data[255:0]
cpu
cpu
diu_cdu_rvalid
diu_data[63:0]
cdu1
cdu2
cdu3
cdu4

FIG. 127

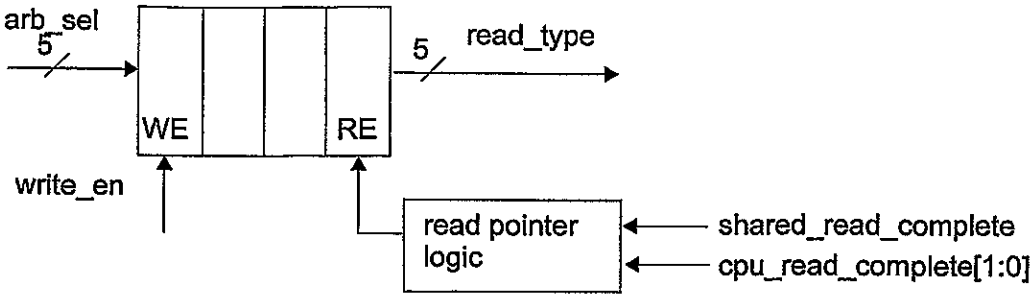


FIG. 128

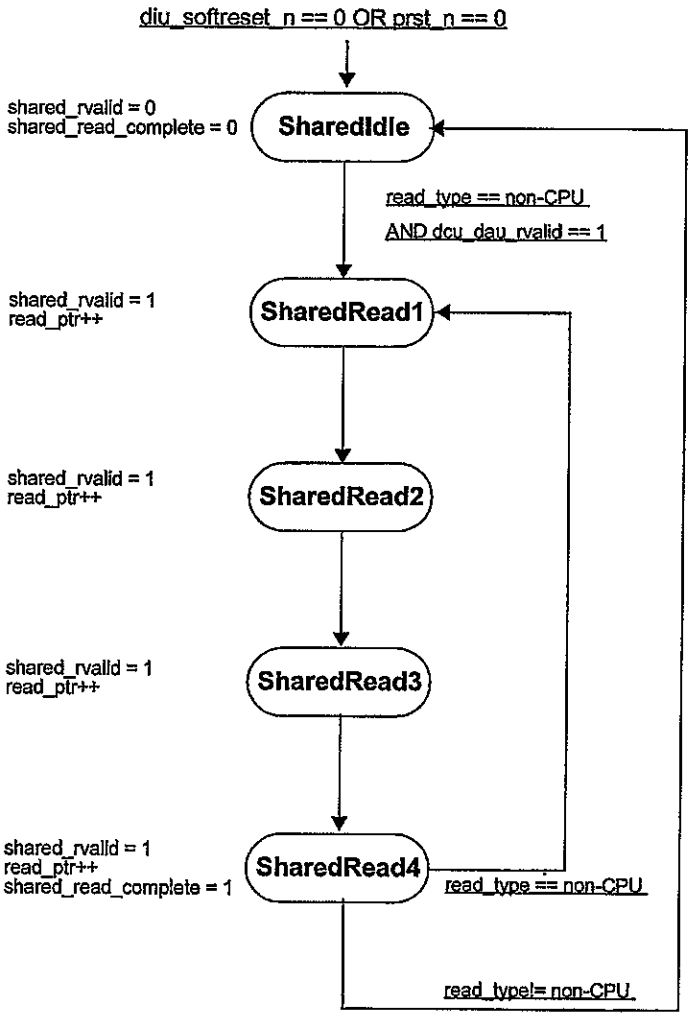


FIG. 129

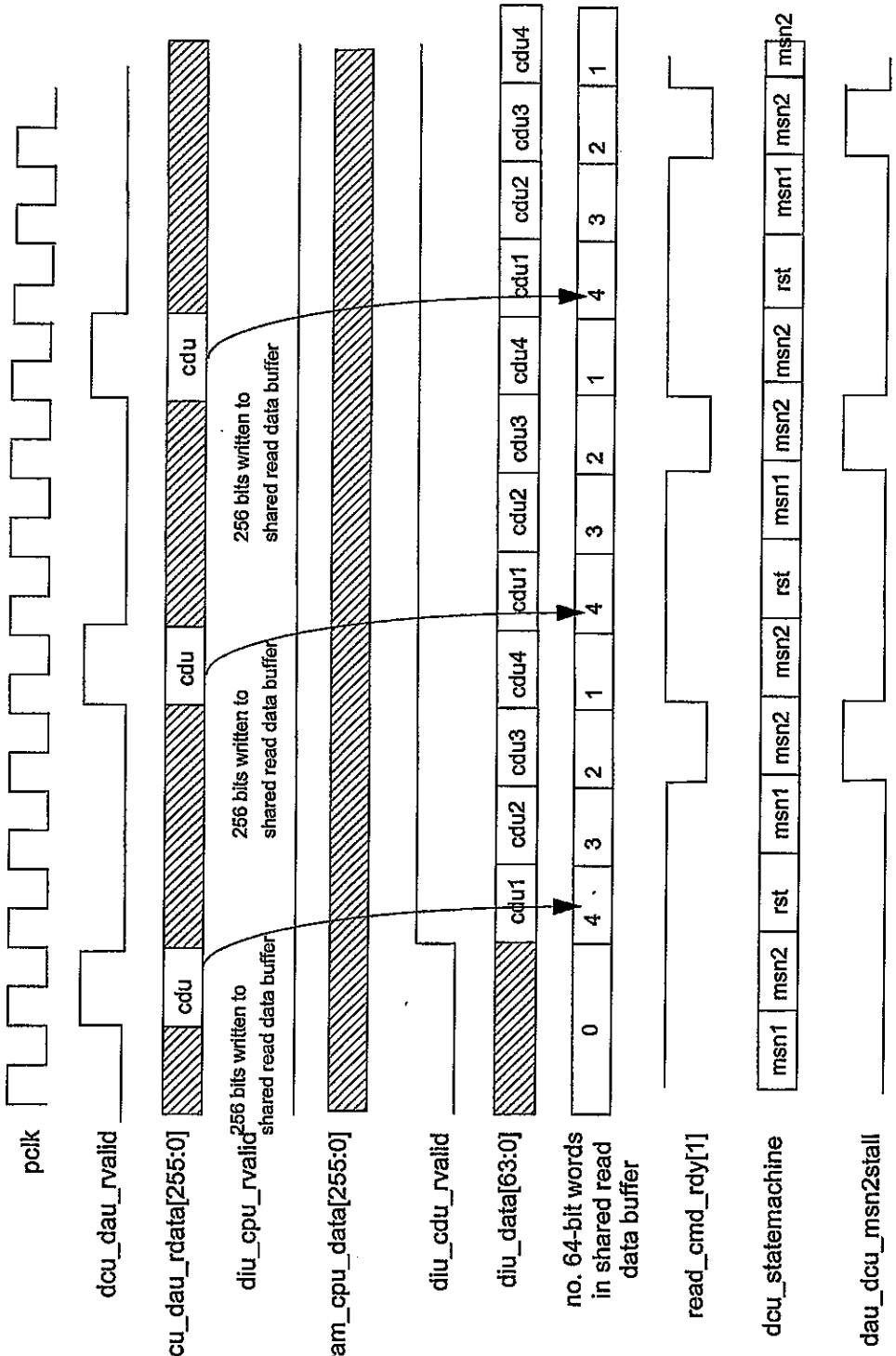

pclk
dcu_dau_rvalid
dcu_dau_rdata[255:0]
cdu
cdu
cdu
diu_cpu_rvalid
256 bits written to shared read data buffer
256 bits written to shared read data buffer
256 bits written to shared read data buffer
dram_cpu_data[255:0]
diu_cdu_rvalid
diu_data[63:0]
cdu1
cdu2
cdu3
cdu4
cdu1
cdu2
cdu3
cdu4
cdu1
cdu2
cdu3
cdu4
no. 64-bit words in shared read data buffer
0
4
3
2
1
4
3
2
1
4
3
2
1
read_cmd_rdy[1]
dcu_statemachine
msn1
msn2
rst
msn1
msn2
rst
msn1
msn2
msn2
rst
msn1
msn2
msn2
dau_dcu_msn2stall

FIG. 130

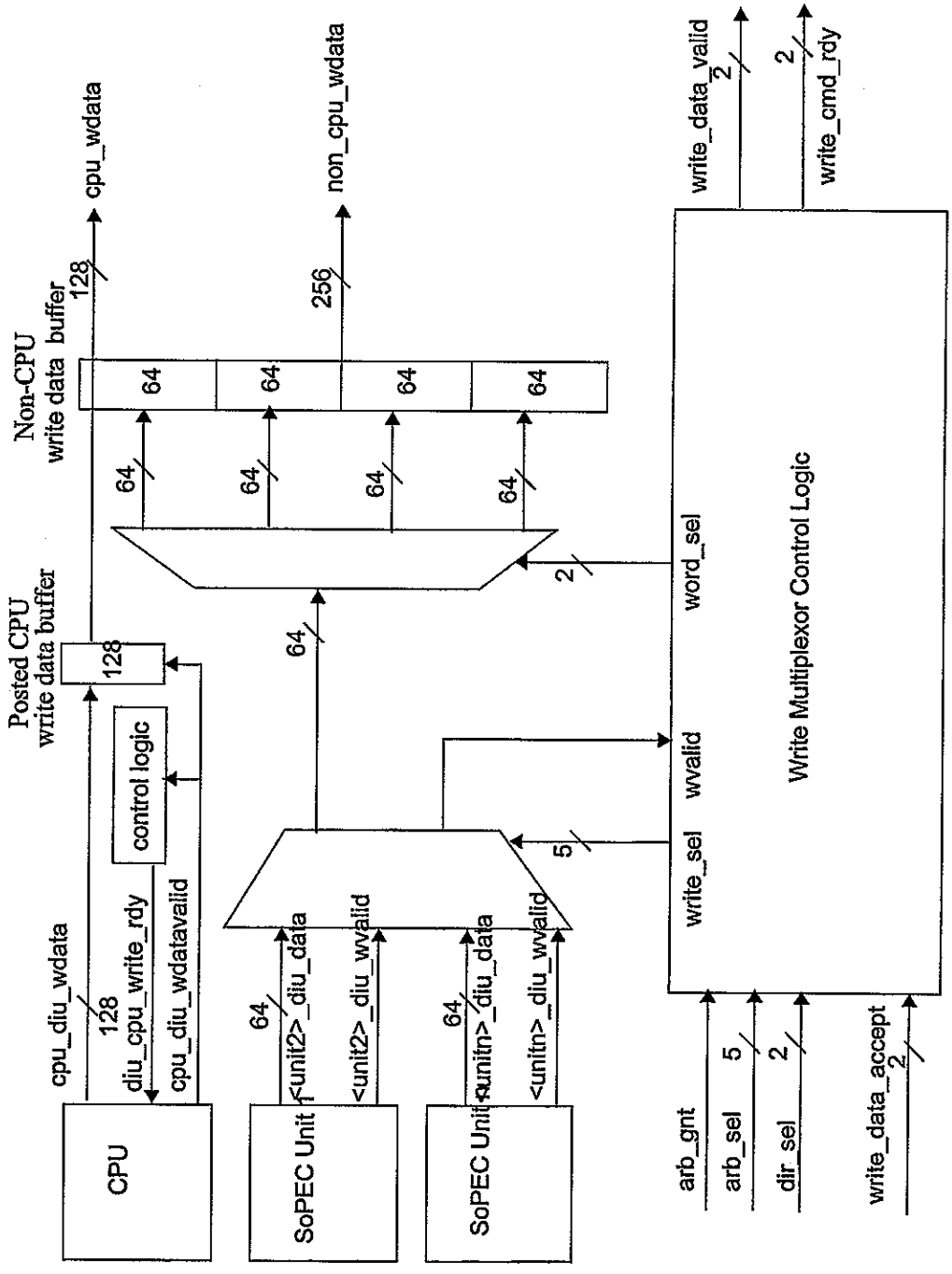

CPU
SoPEC Unit
SoPEC Unit
cpu_diu_wdata
diu_cpu_write_rdy
cpu_diu_wdatavalid
control logic
Posted CPU write data buffer
Non-CPU write data buffer
cpu_wdata
non_cpu_wdata
<unit2>_diu_data
<unit2>_diu_wvalid
<unitn>_diu_data
<unitn>_diu_wvalid
write_sel
wvalid
word_sel
Write Multiplexor Control Logic
write_data_valid
write_cmd_rdy
arb_gnt
arb_sel
dir_sel
write_data_accept

FIG. 131

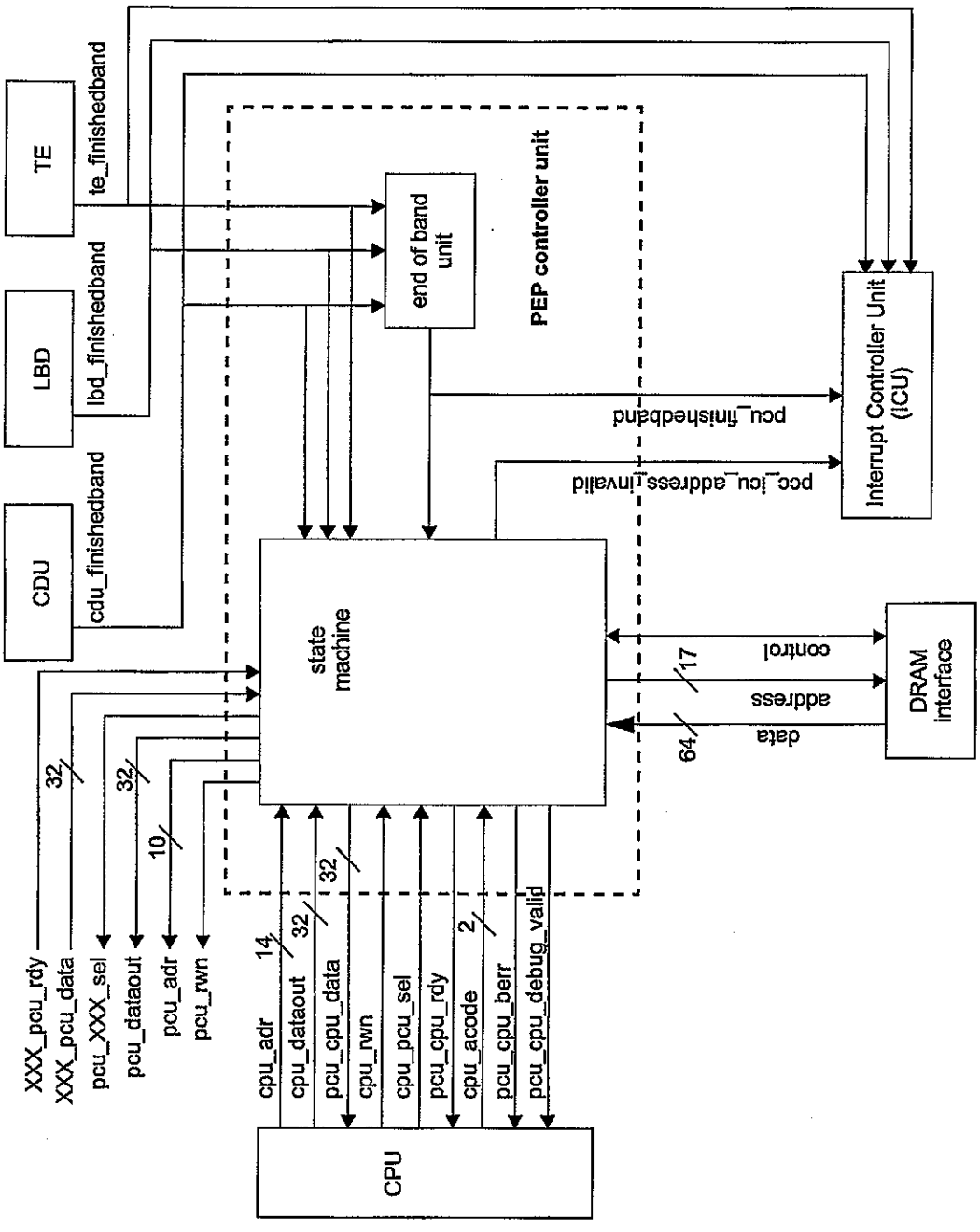

TE
LBD
CDU
te_finishedband
lbd_finishedband
cdu_finishedband
end of band unit
PEP controller unit
Interrupt Controller Unit (ICU)
pcu_finishedband
pcu_icu_addressInvalid
state machine
DRAM interface
control
address
data
17
64
32
32
10
14
32
32
2
XXX_pcu_rdy
XXX_pcu_data
pcu_XXX_sel
pcu_dataout
pcu_adr
pcu_rwn
cpu_adr
cpu_dataout
pcu_cpu_data
cpu_rwn
cpu_pcu_sel
pcu_cpu_rdy
cpu_acode
pcu_cpu_berr
pcu_cpu_debug_valid
CPU

FIG. 132

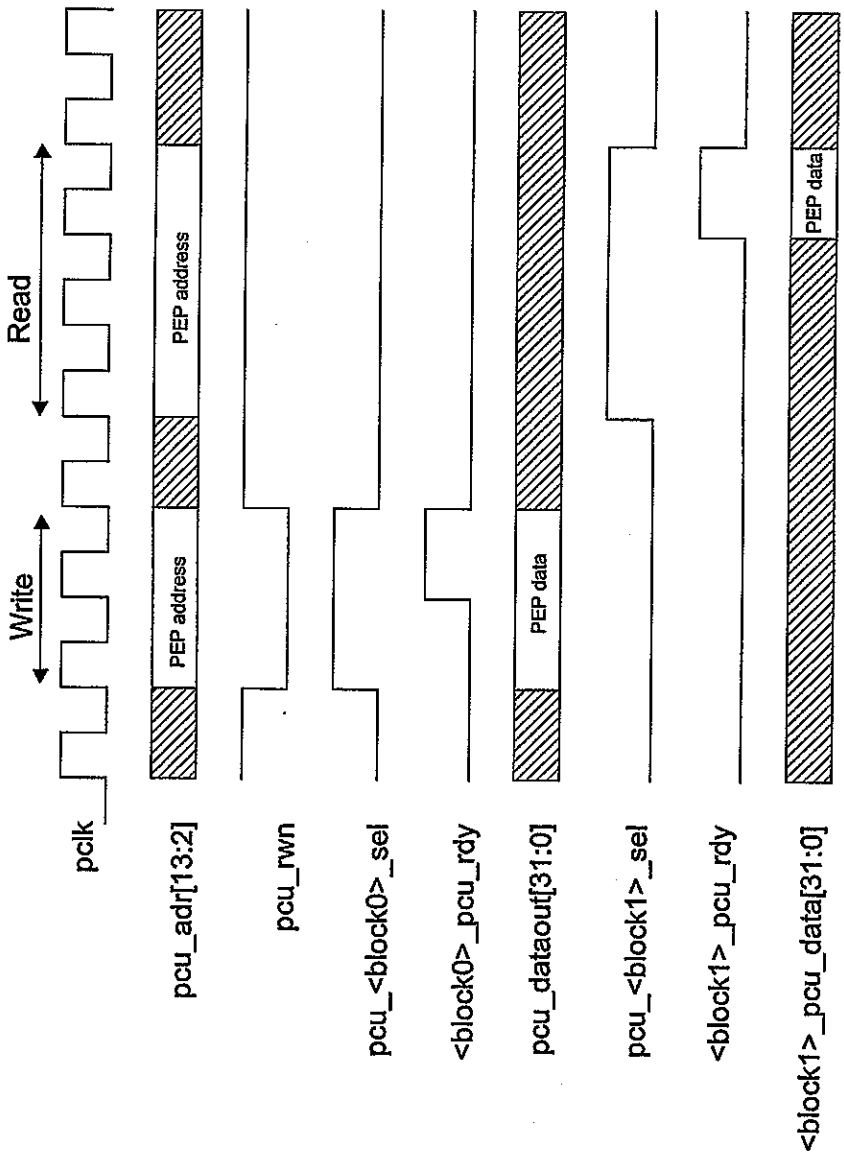

Write
Read
pclk
pcu_adr[13:2]
PEP address
PEP address
pcu_rwn
pcu_<block0>_sel
<block0>_pcu_rdy
pcu_dataout[31:0]
PEP data
pcu_<block1>_sel
<block1>_pcu_rdy
<block1>_pcu_data[31:0]
PEP data

FIG. 133

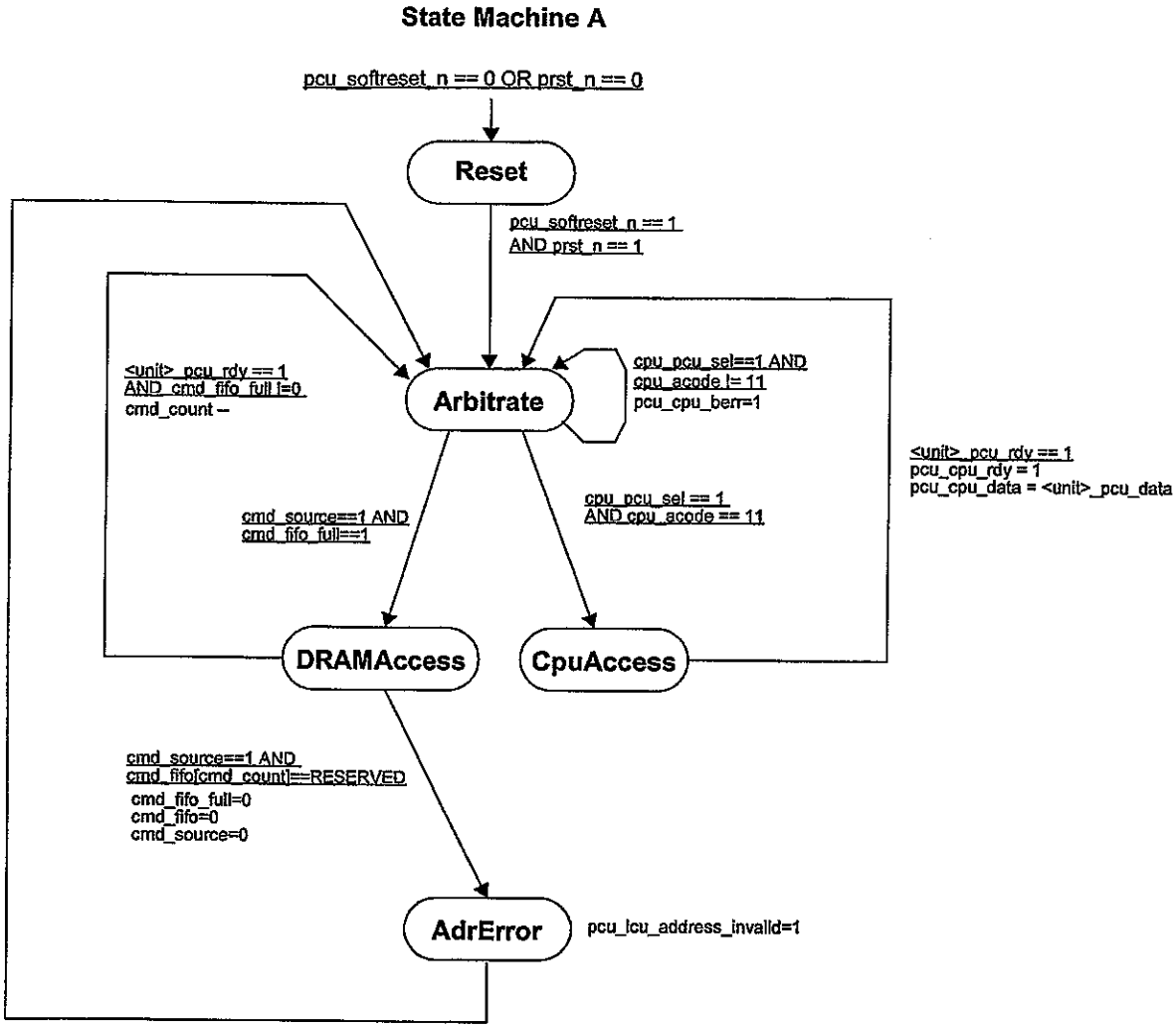

State Machine A
pcu_softreset_n == 0 OR prst_n == 0
Reset
pcu_softreset_n == 1
AND prst_n == 1
<unit>_pcu_rdy == 1
AND cmd_fifo_full !=0
cmd_count --
Arbitrate
cpu_pcu_sel==1 AND
cpu_acode != 11
pcu_cpu_berr=1
<unit>_pcu_rdy == 1
pcu_cpu_rdy = 1
pcu_cpu_data = <unit>_pcu_data
cmd_source==1 AND
cmd_fifo_full==1
cpu_pcu_sel == 1
AND cpu_acode == 11
DRAMAccess
CpuAccess
cmd_source==1 AND
cmd_fifo[cmd_count]==RESERVED
cmd_fifo_full=0
cmd_fifo=0
cmd_source=0
AdrError
pcu_lcu_address_invalid=1

FIG. 134

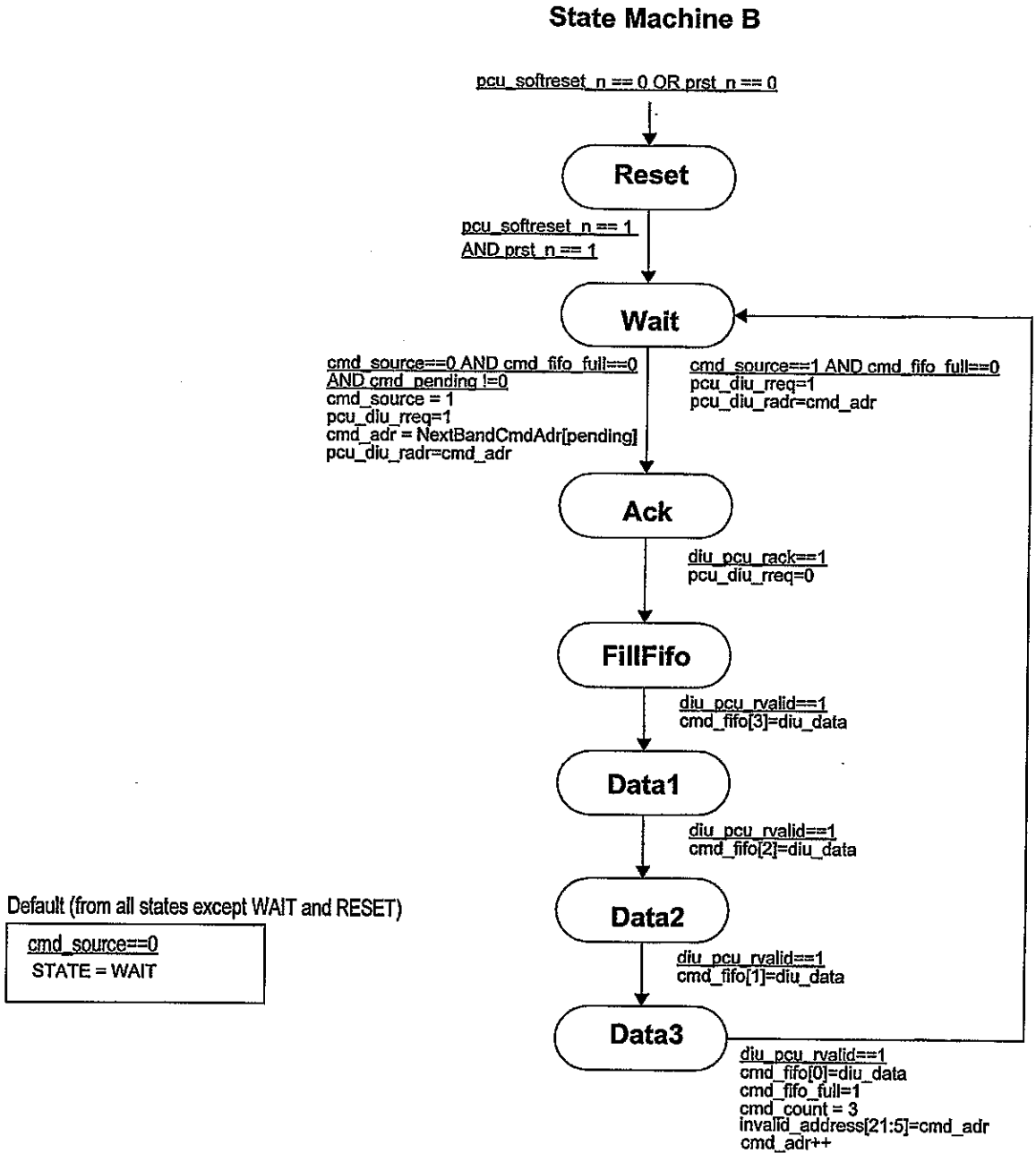

State Machine B
pcu_softreset_n == 0 OR prst_n == 0
Reset
pcu_softreset_n == 1
AND prst_n == 1
Wait
cmd_source==0 AND cmd_fifo_full==0
AND cmd_pending !=0
cmd_source = 1
pcu_diu_rreq=1
cmd_adr = NextBandCmdAdr[pending]
pcu_diu_radr=cmd_adr
cmd_source==1 AND cmd_fifo_full==0
pcu_diu_rreq=1
pcu_diu_radr=cmd_adr
Ack
diu_pcu_rack==1
pcu_diu_rreq=0
FillFifo
diu_pcu_rvalid==1
cmd_fifo[3]=diu_data
Data1
diu_pcu_rvalid==1
cmd_fifo[2]=diu_data
Data2
diu_pcu_rvalid==1
cmd_fifo[1]=diu_data
Data3
diu_pcu_rvalid==1
cmd_fifo[0]=diu_data
cmd_fifo_full=1
cmd_count = 3
invalid_address[21:5]=cmd_adr
cmd_adr++
Default (from all states except WAIT and RESET)
cmd_source==0
STATE = WAIT

FIG. 135

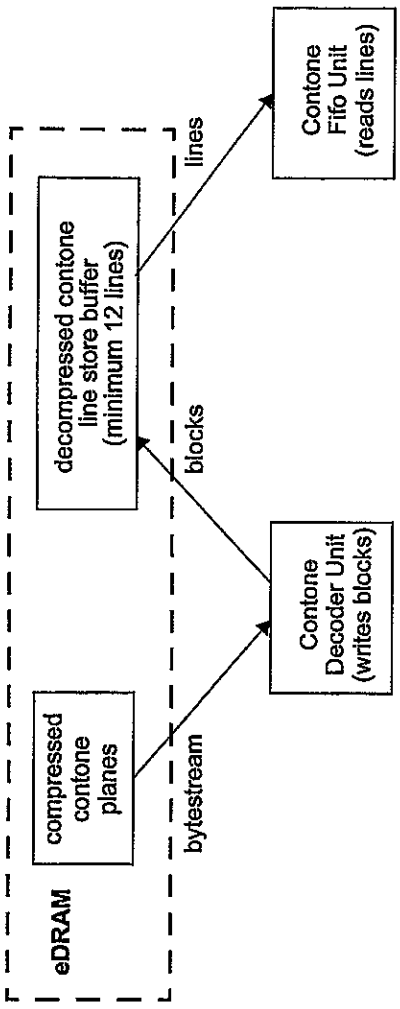

eDRAM
compressed contone planes
bytestream
Contone Decoder Unit (writes blocks)
blocks
decompressed contone line store buffer (minimum 12 lines)
lines
Contone Fifo Unit (reads lines)

FIG. 136

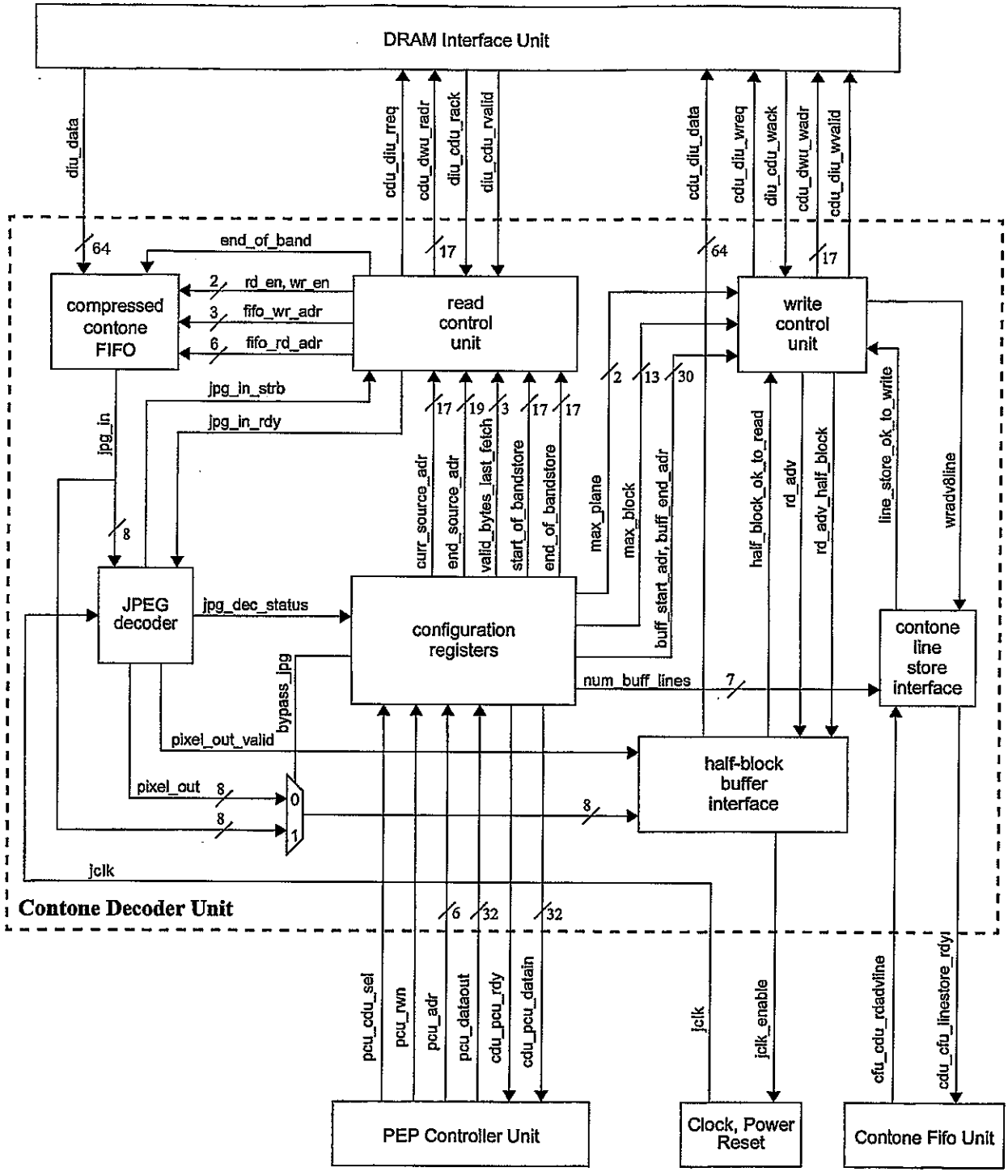

DRAM Interface Unit
diu_data
cdu_diu_rreq
cdu_dwu_radr
diu_cdu_rack
diu_cdu_rvalid
cdu_diu_data
cdu_diu_wreq
diu_cdu_wack
cdu_dwu_wadr
cdu_diu_wvalid
64
17
64
17
end_of_band
compressed contone FIFO
2
rd_en, wr_en
3
fifo_wr_adr
6
fifo_rd_adr
read control unit
write control unit
jpg_in_strb
jpg_in_rdy
jpg_in
8
17
19
3
17
17
2
13
30
curr_source_adr
end_source_adr
valid_bytes_last_fetch
start_of_bandstore
end_of_bandstore
max_plane
max_block
buff_start_adr, buff_end_adr
half_block_ok_to_read
rd_adv
rd_adv_half_block
line_store_ok_to_write
wradv8line
JPEG decoder
jpg_dec_status
configuration registers
contone line store interface
bypass_jpg
num_buff_lines
7
pixel_out_valid
half-block buffer interface
pixel_out
8
8
0
1
8
jclk
Contone Decoder Unit
6
32
32
pcu_cdu_sel
pcu_rwn
pcu_adr
pcu_dataout
cdu_pcu_rdy
cdu_pcu_datain
jclk
jclk_enable
cfu_cdu_rdadv8line
cdu_cfu_linestore_rdy
PEP Controller Unit
Clock, Power Reset
Contone Fifo Unit

FIG. 137

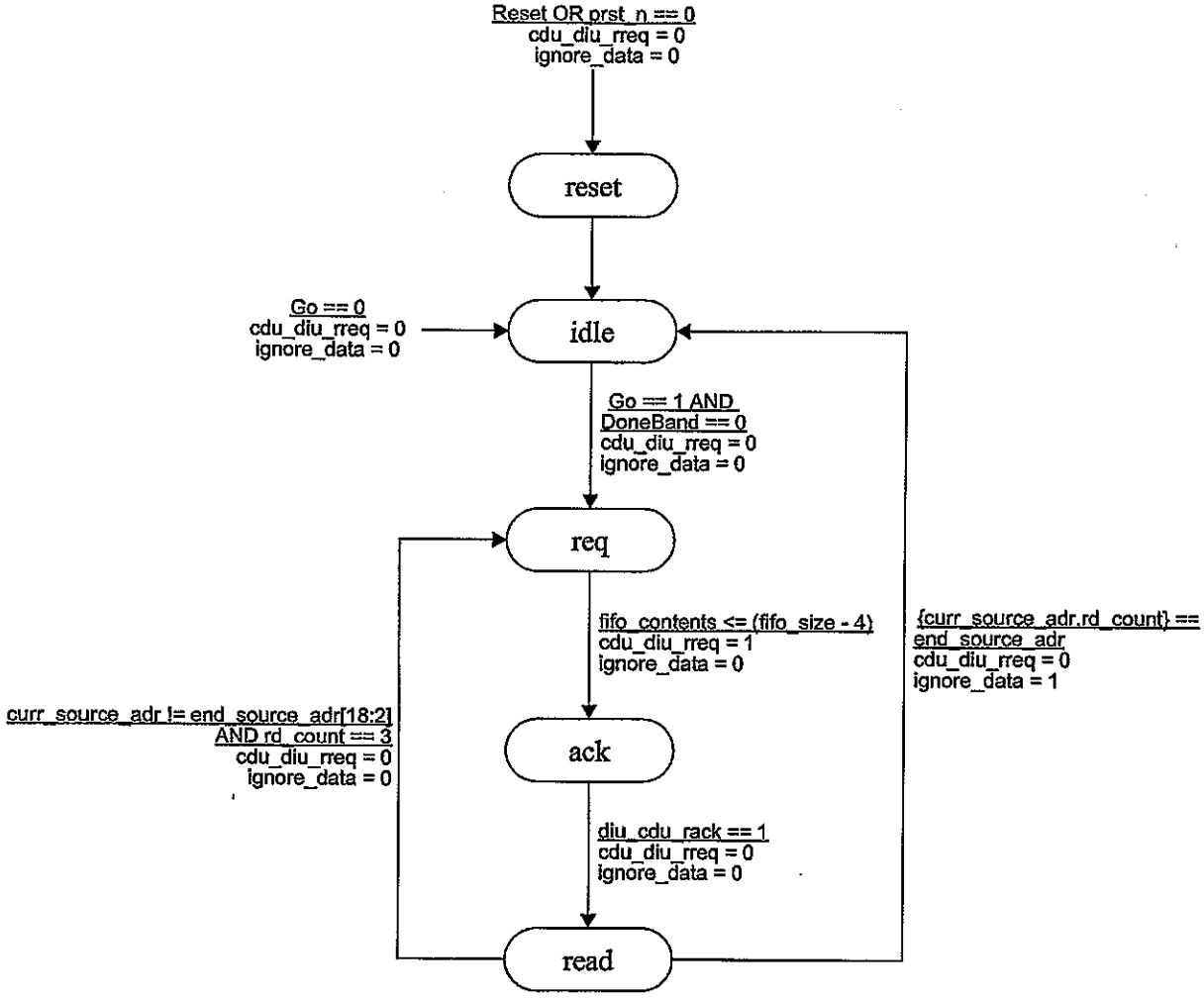

Reset OR prst_n == 0
cdu_diu_rreq = 0
ignore_data = 0
reset
Go == 0
cdu_diu_rreq = 0
ignore_data = 0
idle
Go == 1 AND
DoneBand == 0
cdu_diu_rreq = 0
ignore_data = 0
req
fifo_contents <= (fifo_size - 4)
cdu_diu_rreq = 1
ignore_data = 0
{curr_source_adr.rd_count} ==
end_source_adr
cdu_diu_rreq = 0
ignore_data = 1
curr_source_adr != end_source_adr[18:2]
AND rd_count == 3
cdu_diu_rreq = 0
ignore_data = 0
ack
diu_cdu_rack == 1
cdu_diu_rreq = 0
ignore_data = 0
read

FIG. 138

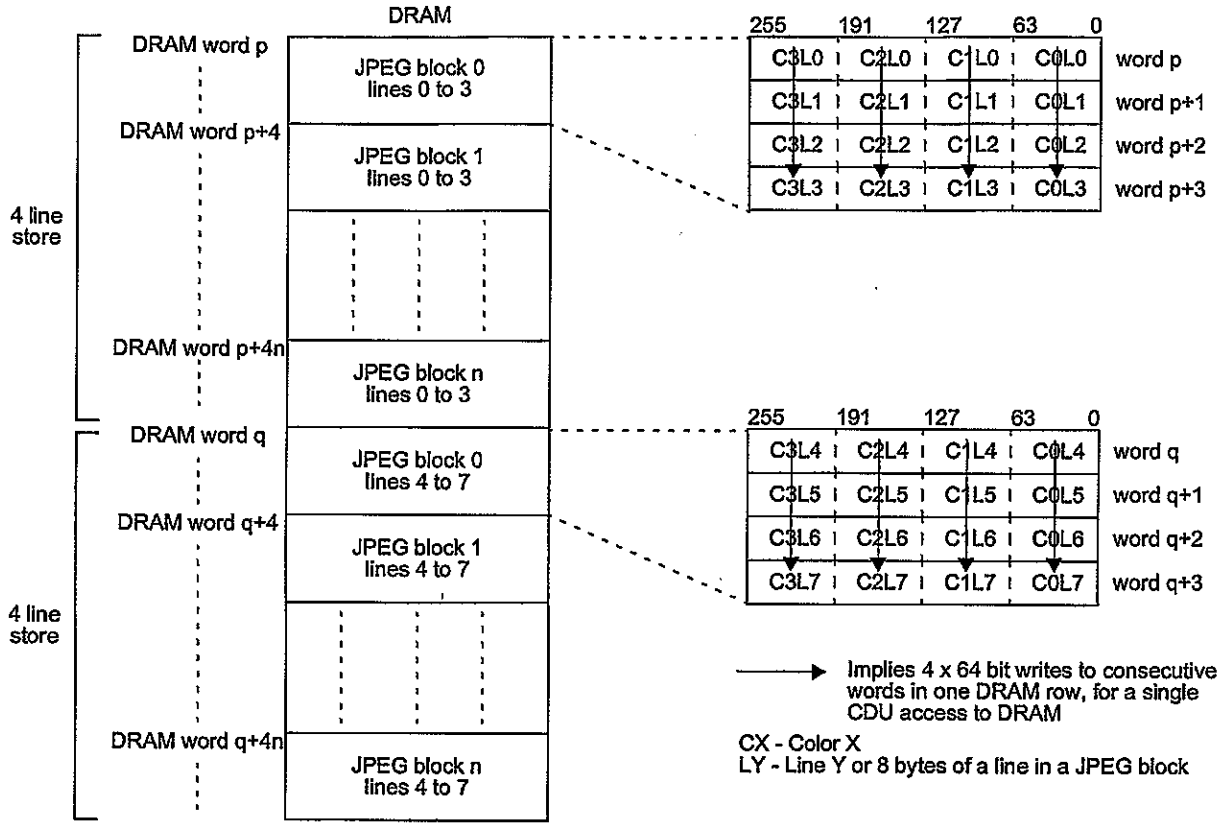

DRAM
DRAM word p
DRAM word p+4
DRAM word p+4n
4 line store
JPEG block 0 lines 0 to 3
JPEG block 1 lines 0 to 3
JPEG block n lines 0 to 3
DRAM word q
DRAM word q+4
DRAM word q+4n
4 line store
JPEG block 0 lines 4 to 7
JPEG block 1 lines 4 to 7
JPEG block n lines 4 to 7
255 191 127 63 0
C3L0 C2L0 C1L0 C0L0 word p
C3L1 C2L1 C1L1 C0L1 word p+1
C3L2 C2L2 C1L2 C0L2 word p+2
C3L3 C2L3 C1L3 C0L3 word p+3
255 191 127 63 0
C3L4 C2L4 C1L4 C0L4 word q
C3L5 C2L5 C1L5 C0L5 word q+1
C3L6 C2L6 C1L6 C0L6 word q+2
C3L7 C2L7 C1L7 C0L7 word q+3
Implies 4 x 64 bit writes to consecutive words in one DRAM row, for a single CDU access to DRAM
CX - Color X
LY - Line Y or 8 bytes of a line in a JPEG block

FIG. 139

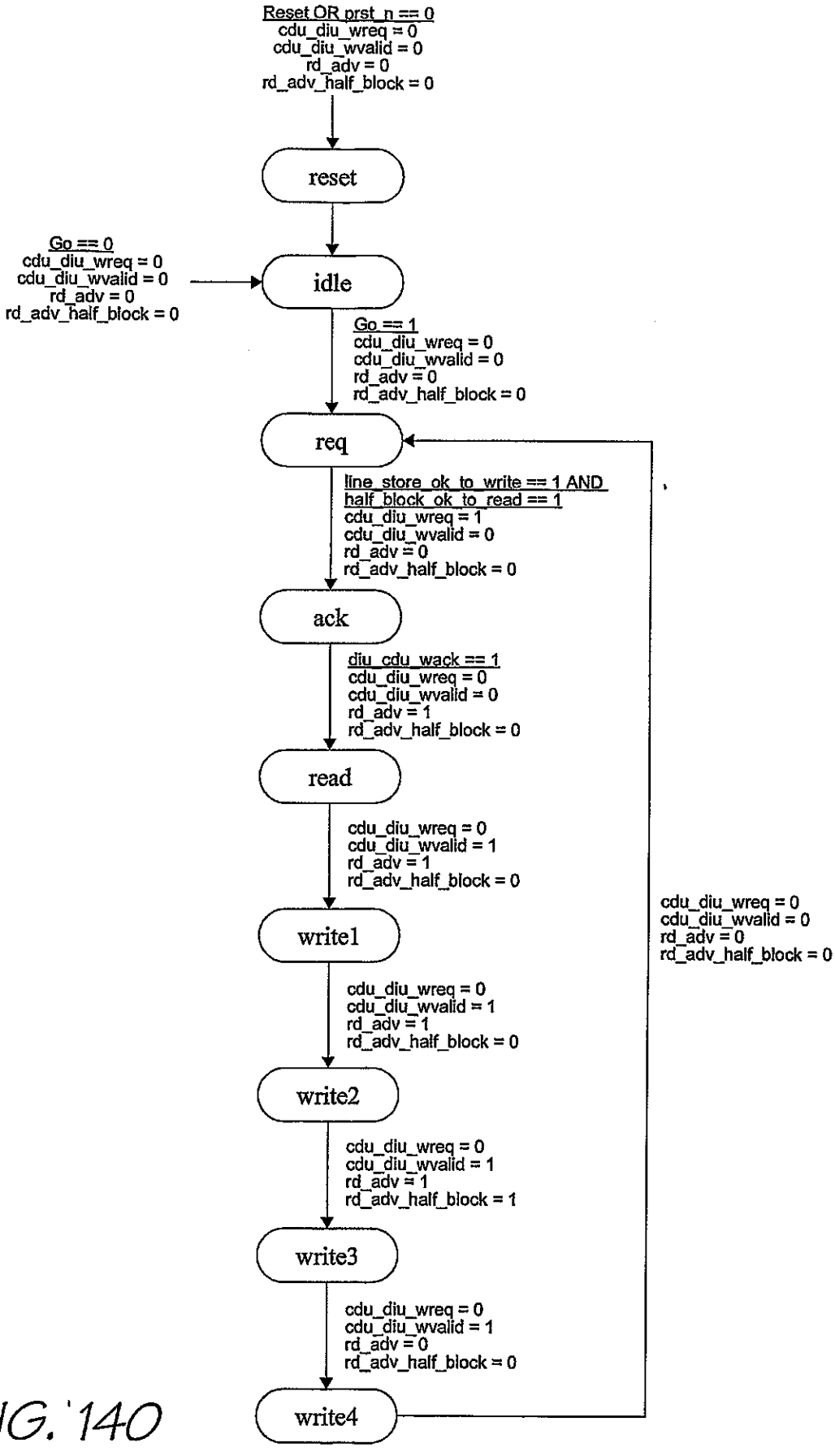

Reset OR prst_n == 0
cdu_diu_wreq = 0
cdu_diu_wvalid = 0
rd_adv = 0
rd_adv_half_block = 0
reset
Go == 0
cdu_diu_wreq = 0
cdu_diu_wvalid = 0
rd_adv = 0
rd_adv_half_block = 0
idle
Go == 1
cdu_diu_wreq = 0
cdu_diu_wvalid = 0
rd_adv = 0
rd_adv_half_block = 0
req
line_store_ok_to_write == 1 AND
half_block_ok_to_read == 1
cdu_diu_wreq = 1
cdu_diu_wvalid = 0
rd_adv = 0
rd_adv_half_block = 0
ack
diu_cdu_wack == 1
cdu_diu_wreq = 0
cdu_diu_wvalid = 0
rd_adv = 1
rd_adv_half_block = 0
read
cdu_diu_wreq = 0
cdu_diu_wvalid = 1
rd_adv = 1
rd_adv_half_block = 0
write1
cdu_diu_wreq = 0
cdu_diu_wvalid = 1
rd_adv = 1
rd_adv_half_block = 0
write2
cdu_diu_wreq = 0
cdu_diu_wvalid = 1
rd_adv = 1
rd_adv_half_block = 1
write3
cdu_diu_wreq = 0
cdu_diu_wvalid = 1
rd_adv = 0
rd_adv_half_block = 0
write4
cdu_diu_wreq = 0
cdu_diu_wvalid = 0
rd_adv = 0
rd_adv_half_block = 0

FIG. 140

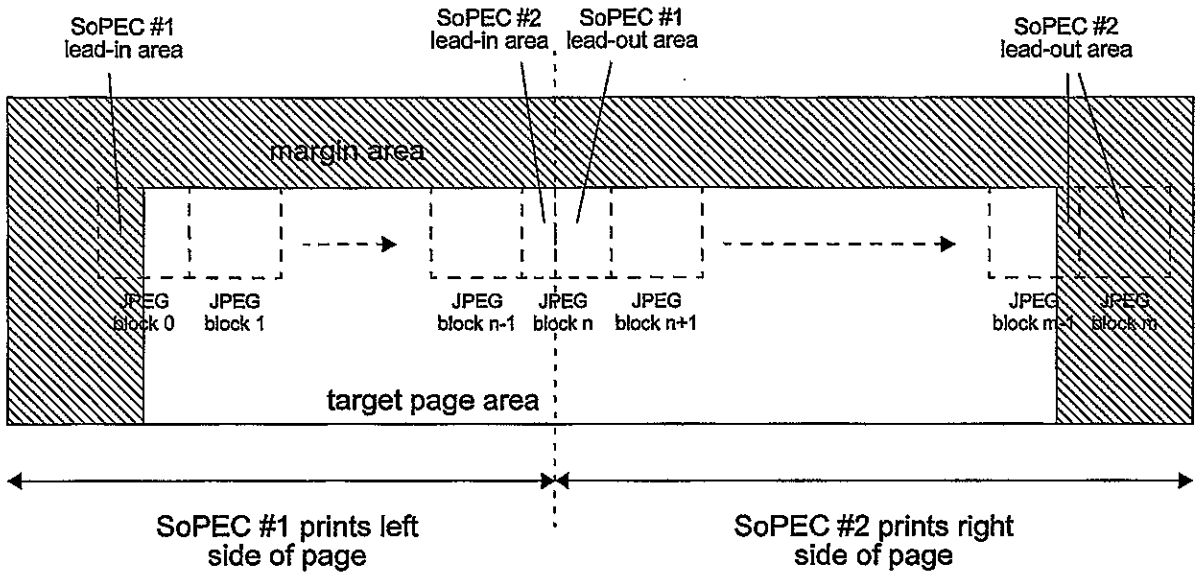

SoPEC #1
lead-in area
SoPEC #2
lead-in area
SoPEC #1
lead-out area
SoPEC #2
lead-out area
margin area
JPEG
block 0
JPEG
block 1
JPEG
block n-1
JPEG
block n
JPEG
block n+1
JPEG
block m-1
JPEG
block m
target page area
SoPEC #1 prints left
side of page
SoPEC #2 prints right
side of page

FIG. 141

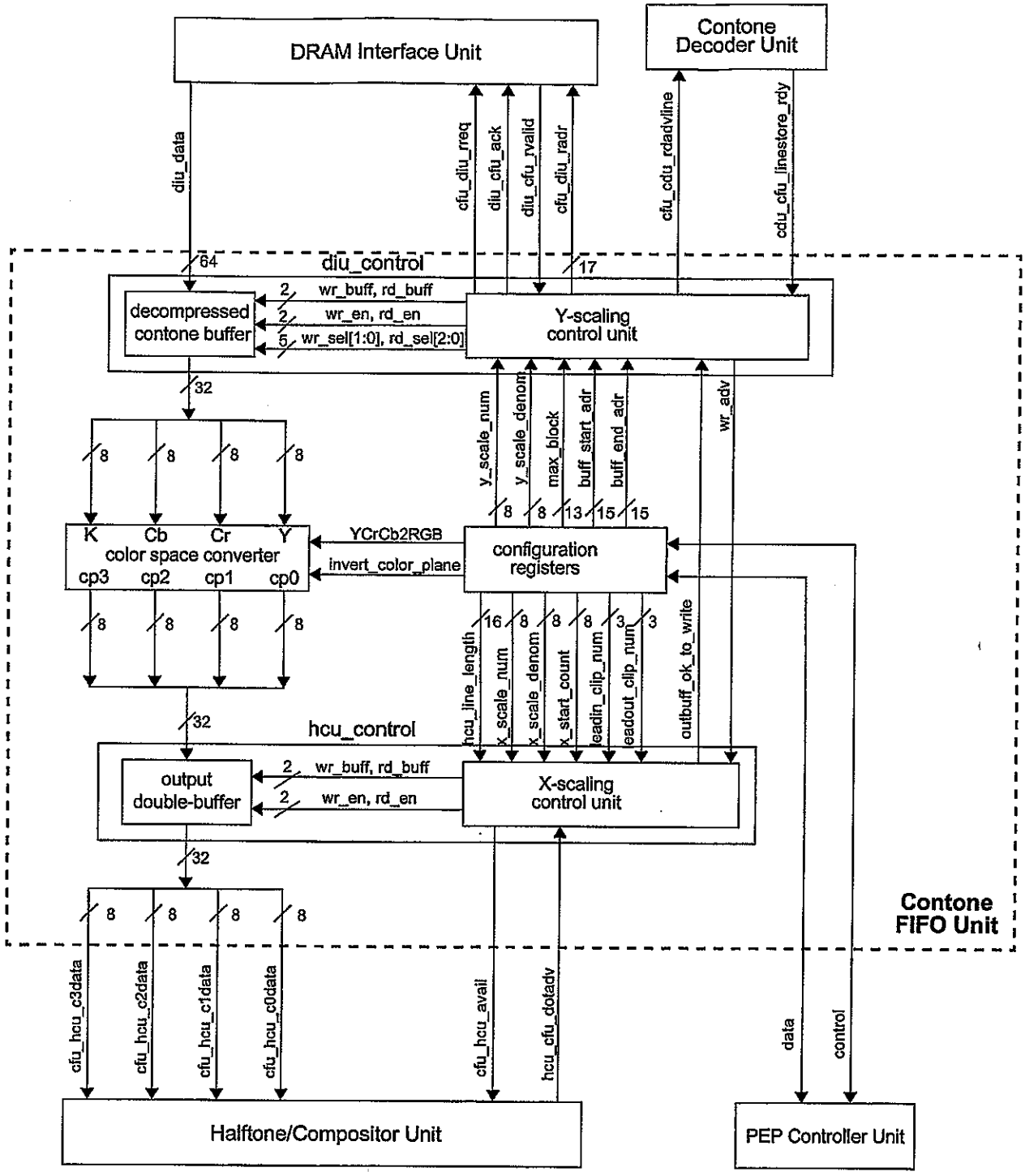

DRAM Interface Unit
Contone Decoder Unit
diu_data
cfu_diu_rreq
diu_cfu_ack
diu_cfu_rvalid
cfu_diu_radr
cfu_cdu_rdadvline
cdu_cfu_linestore_rdy
diu_control
64
17
2
wr_buff, rd_buff
2
wr_en, rd_en
5
wr_sel[1:0], rd_sel[2:0]
decompressed contone buffer
Y-scaling control unit
32
8
8
8
8
y_scale_num
y_scale_denom
max_block
buff_start_adr
buff_end_adr
wr_adv
8
8
13
15
15
K
Cb
Cr
Y
color space converter
cp3
cp2
cp1
cp0
YCrCb2RGB
invert_color_plane
configuration registers
8
8
8
8
16
8
8
8
3
3
hcu_line_length
x_scale_num
x_scale_denom
x_start_count
leadin_clip_num
leadout_clip_num
outbuff_ok_to_write
32
hcu_control
output double-buffer
2
wr_buff, rd_buff
2
wr_en, rd_en
X-scaling control unit
32
8
8
8
8
Contone FIFO Unit
cfu_hcu_c3data
cfu_hcu_c2data
cfu_hcu_c1data
cfu_hcu_c0data
cfu_hcu_avail
hcu_cfu_dotadv
data
control
Halftone/Compositor Unit
PEP Controller Unit

FIG. 142

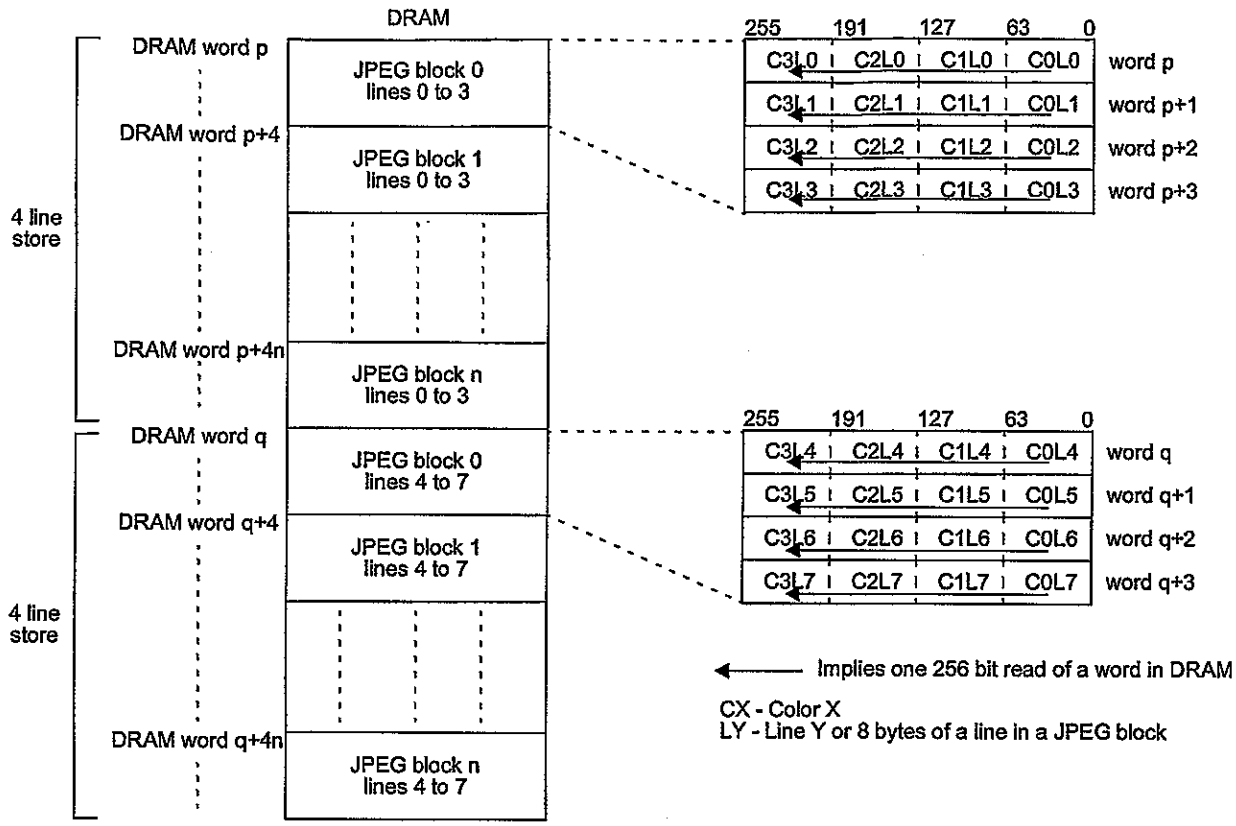

DRAM
DRAM word p
DRAM word p+4
DRAM word p+4n
DRAM word q
DRAM word q+4
DRAM word q+4n
4 line store
4 line store
JPEG block 0 lines 0 to 3
JPEG block 1 lines 0 to 3
JPEG block n lines 0 to 3
JPEG block 0 lines 4 to 7
JPEG block 1 lines 4 to 7
JPEG block n lines 4 to 7
255 191 127 63 0
C3L0 C2L0 C1L0 C0L0 word p
C3L1 C2L1 C1L1 C0L1 word p+1
C3L2 C2L2 C1L2 C0L2 word p+2
C3L3 C2L3 C1L3 C0L3 word p+3
255 191 127 63 0
C3L4 C2L4 C1L4 C0L4 word q
C3L5 C2L5 C1L5 C0L5 word q+1
C3L6 C2L6 C1L6 C0L6 word q+2
C3L7 C2L7 C1L7 C0L7 word q+3
Implies one 256 bit read of a word in DRAM
CX - Color X
LY - Line Y or 8 bytes of a line in a JPEG block

FIG. 143

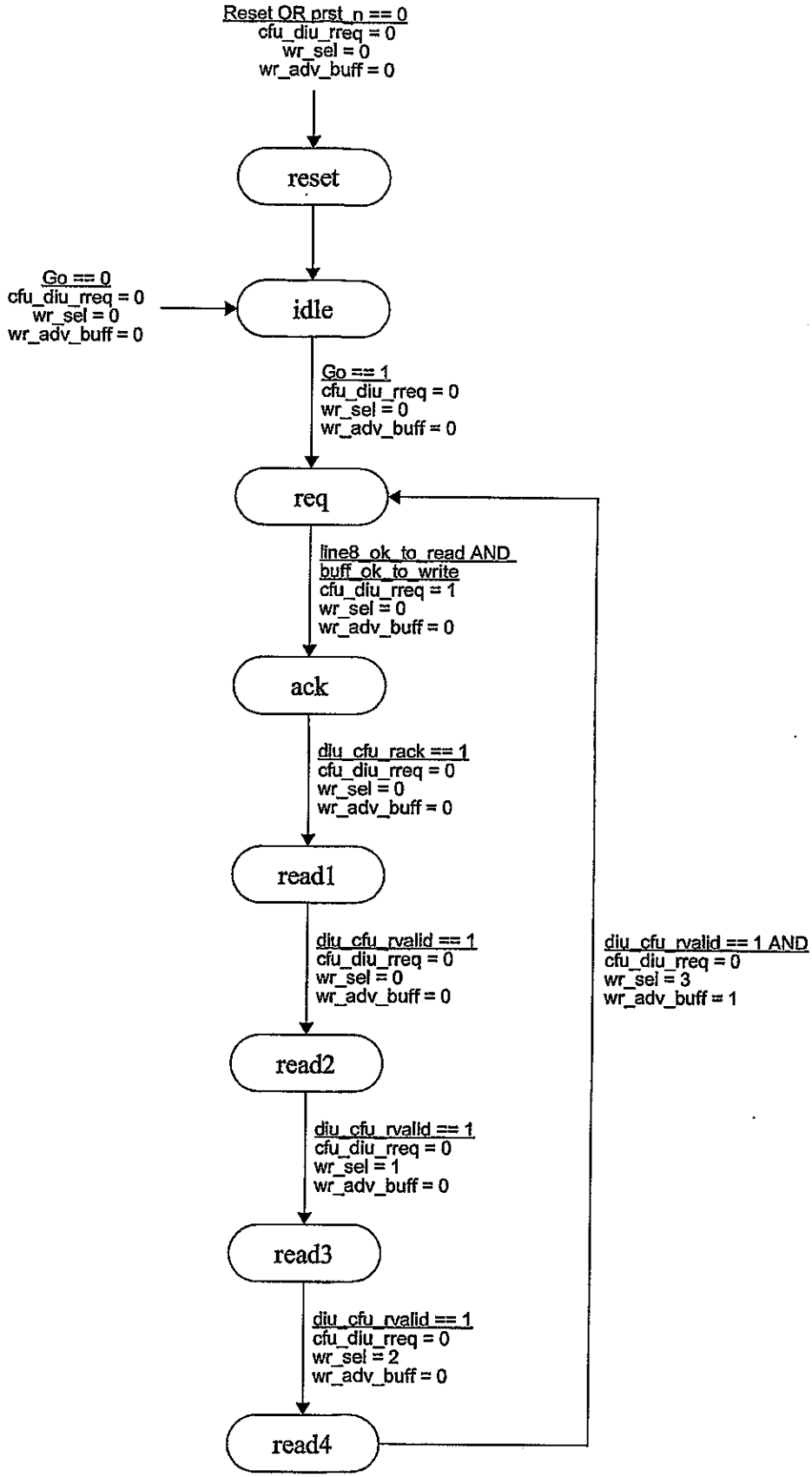

Reset OR prst_n == 0
cfu_diu_rreq = 0
wr_sel = 0
wr_adv_buff = 0
reset
Go == 0
cfu_diu_rreq = 0
wr_sel = 0
wr_adv_buff = 0
idle
Go == 1
cfu_diu_rreq = 0
wr_sel = 0
wr_adv_buff = 0
req
line8_ok_to_read AND
buff_ok_to_write
cfu_diu_rreq = 1
wr_sel = 0
wr_adv_buff = 0
ack
diu_cfu_rack == 1
cfu_diu_rreq = 0
wr_sel = 0
wr_adv_buff = 0
read1
diu_cfu_rvalid == 1
cfu_diu_rreq = 0
wr_sel = 0
wr_adv_buff = 0
read2
diu_cfu_rvalid == 1
cfu_diu_rreq = 0
wr_sel = 1
wr_adv_buff = 0
read3
diu_cfu_rvalid == 1
cfu_diu_rreq = 0
wr_sel = 2
wr_adv_buff = 0
read4
diu_cfu_rvalid == 1 AND
cfu_diu_rreq = 0
wr_sel = 3
wr_adv_buff = 1

FIG. 144

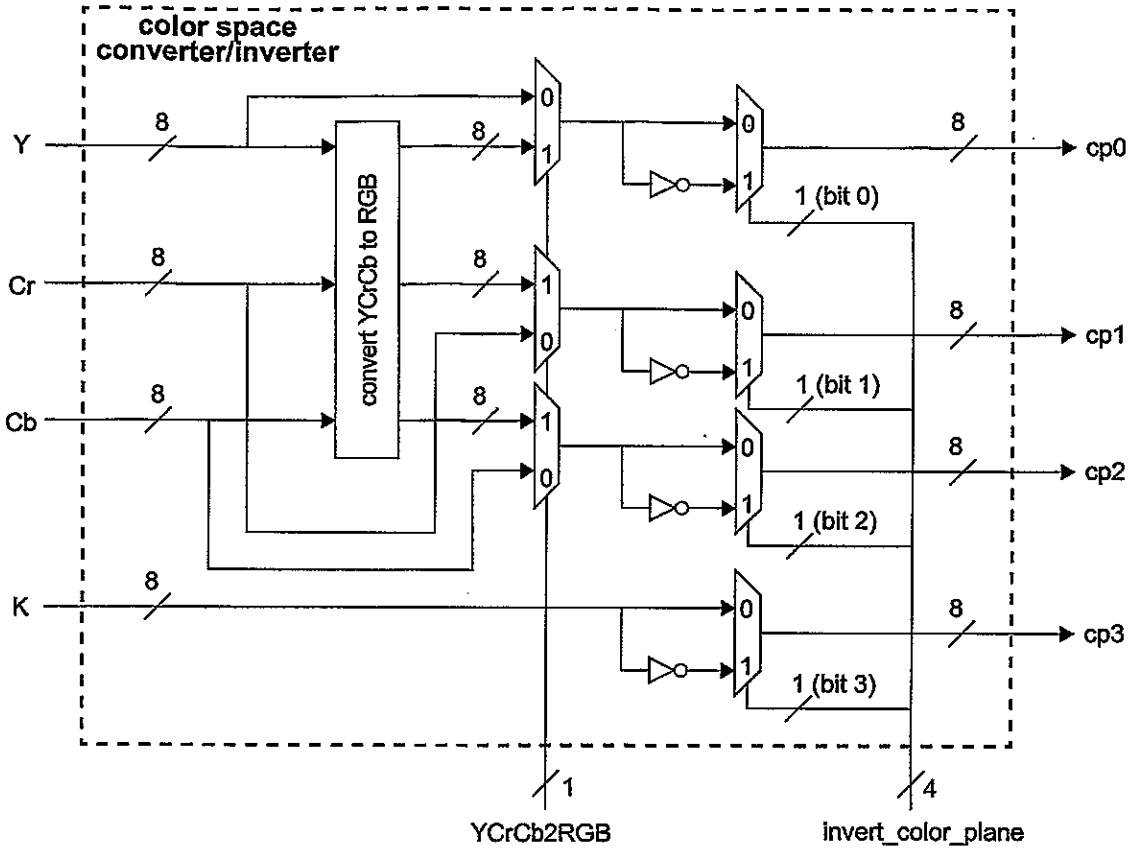


FIG. 145

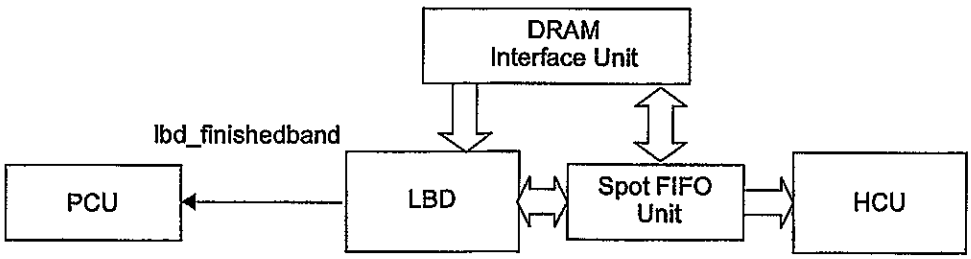


FIG. 146

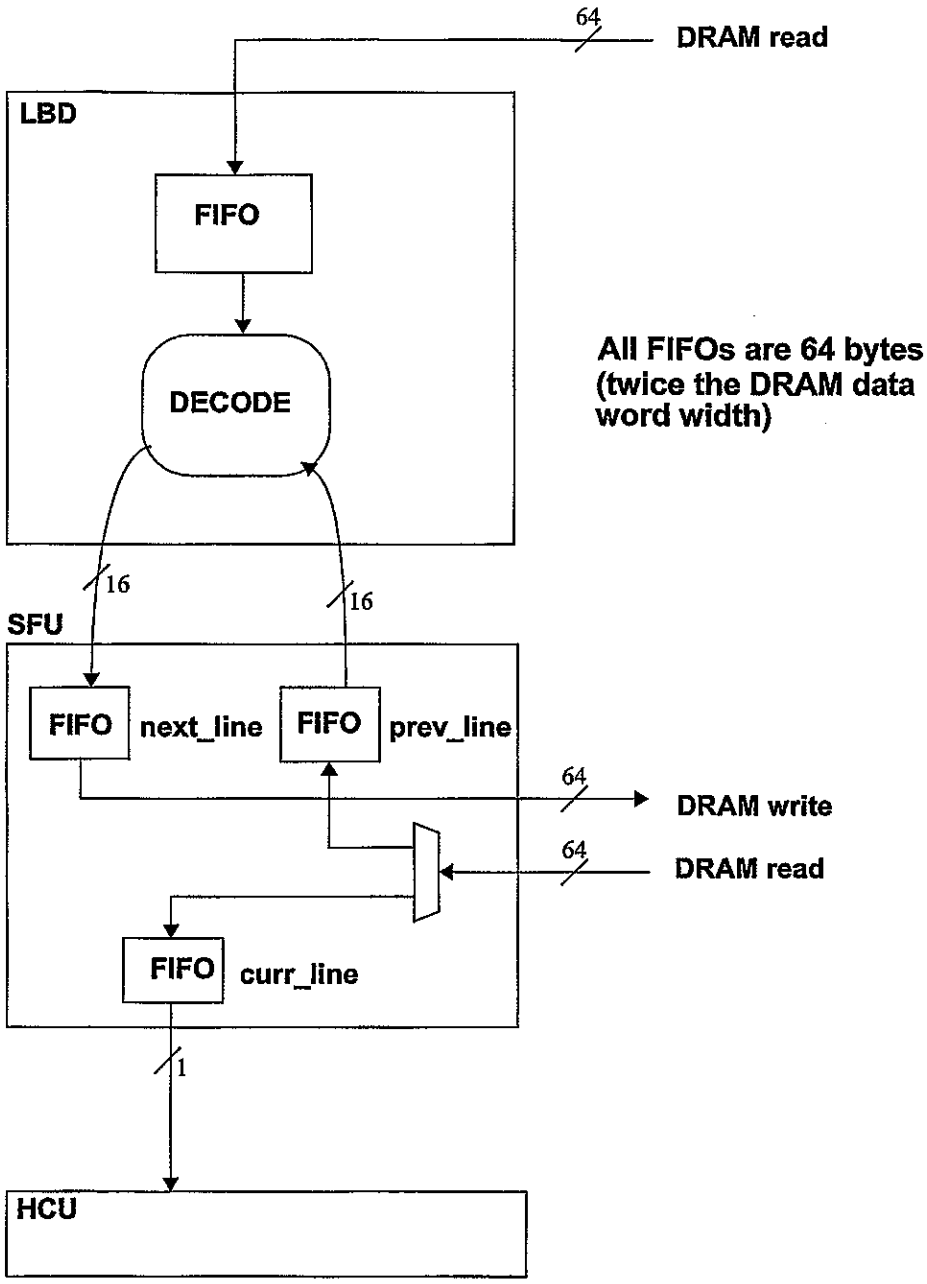

64
DRAM read
LBD
FIFO
DECODE
All FIFOs are 64 bytes (twice the DRAM data word width)
16
16
SFU
FIFO
next_line
FIFO
prev_line
64
DRAM write
64
DRAM read
FIFO
curr_line
1
HCU

FIG. 147

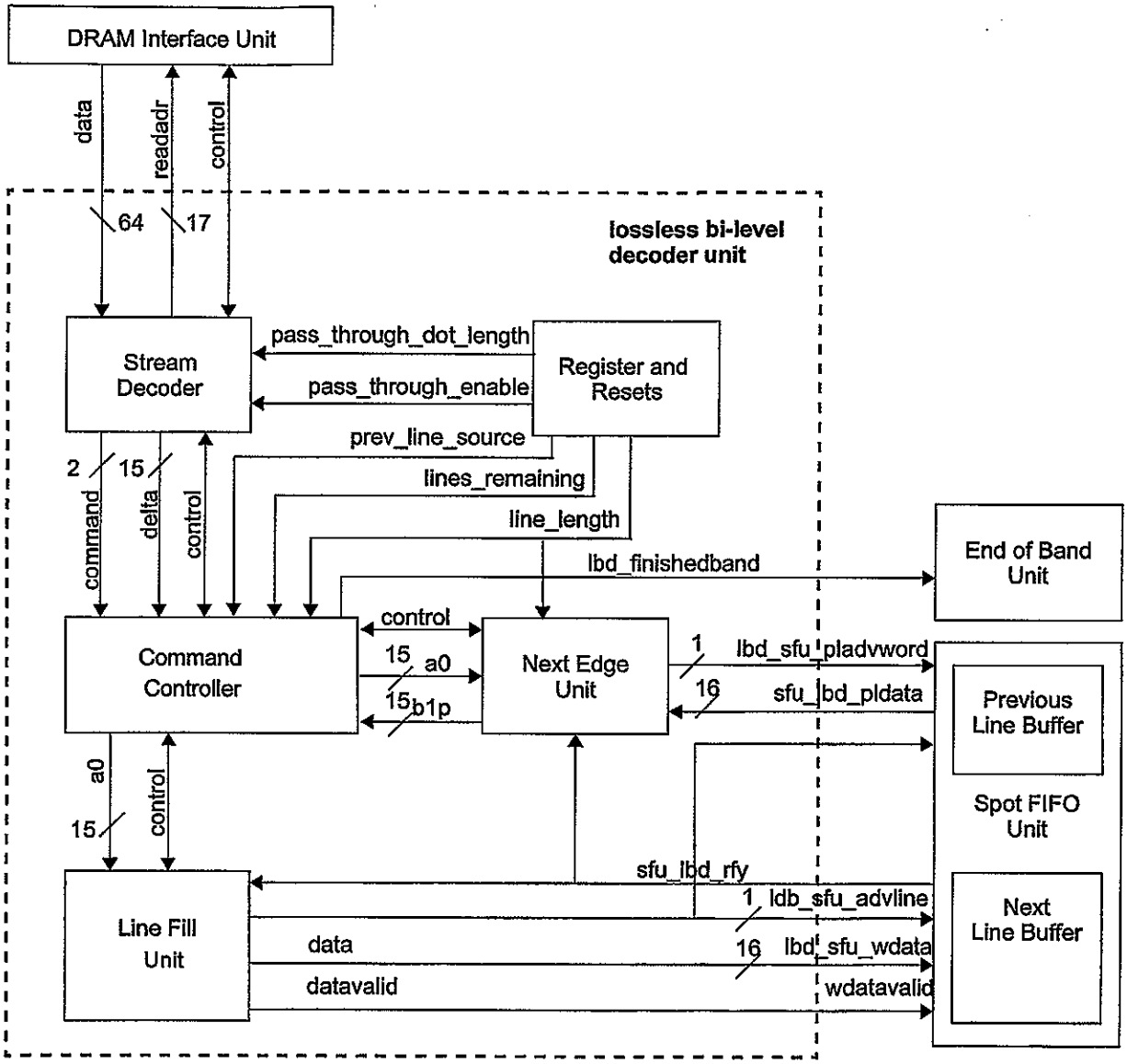

DRAM Interface Unit
data
readadr
control
64
17
lossless bi-level decoder unit
Stream Decoder
pass_through_dot_length
pass_through_enable
Register and Resets
prev_line_source
lines_remaining
line_length
2
15
command
delta
control
lbd_finishedband
End of Band Unit
control
15
a0
15
b1p
Command Controller
Next Edge Unit
1
lbd_sfu_pladvword
16
sfu_lbd_pldata
Previous Line Buffer
Spot FIFO Unit
a0
15
control
sfu_lbd_rfy
1
ldb_sfu_advline
Next Line Buffer
Line Fill Unit
data
16
lbd_sfu_wdata
datavalid
wdatavalid

FIG. 148

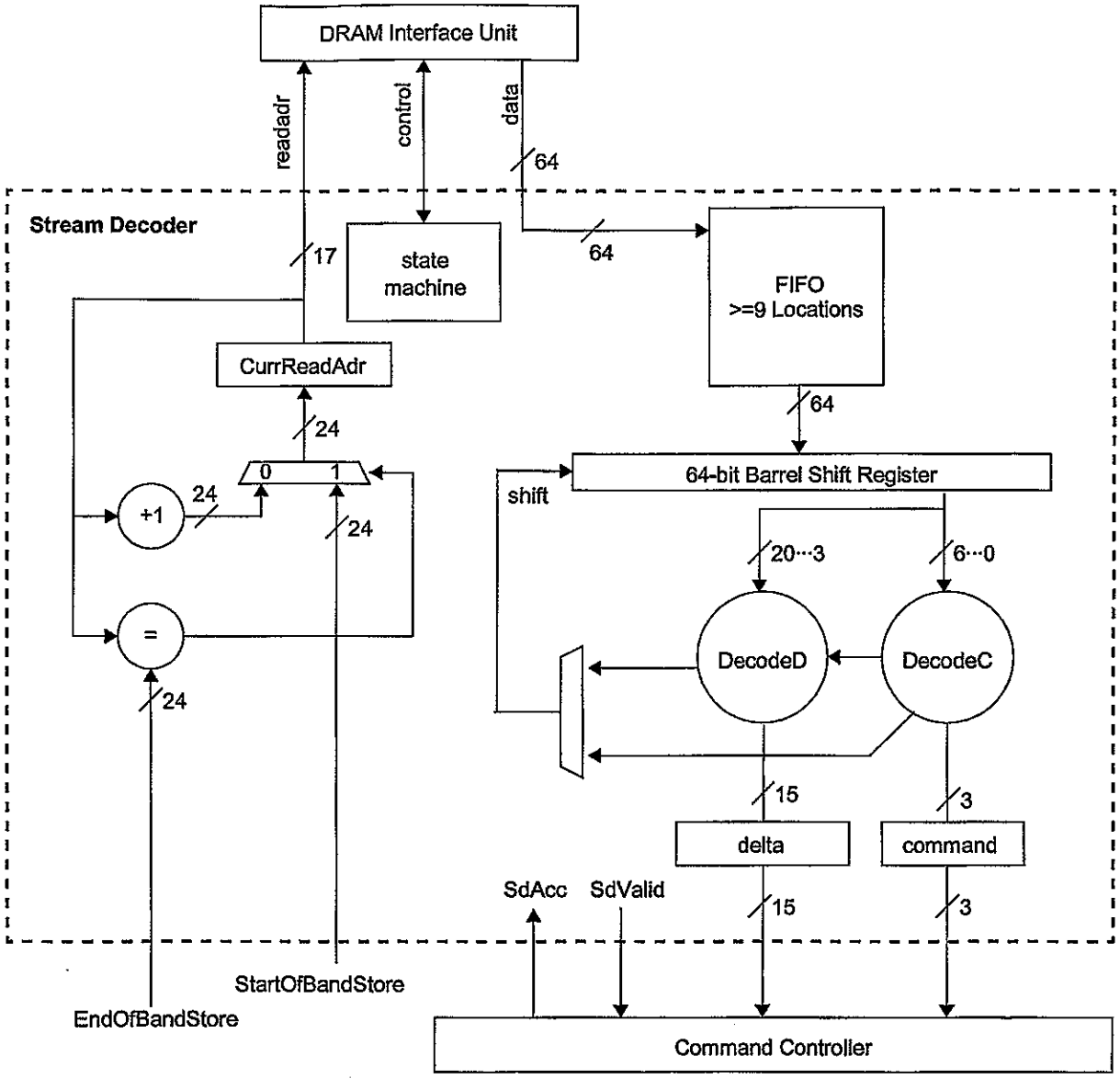

DRAM Interface Unit
readadr
control
data
64
Stream Decoder
17
state machine
64
FIFO >=9 Locations
CurrReadAdr
24
64
0
1
64-bit Barrel Shift Register
shift
+1
24
24
20···3
6···0
=
DecodeD
DecodeC
24
15
3
delta
command
SdAcc
SdValid
15
3
StartOfBandStore
EndOfBandStore
Command Controller

FIG. 149

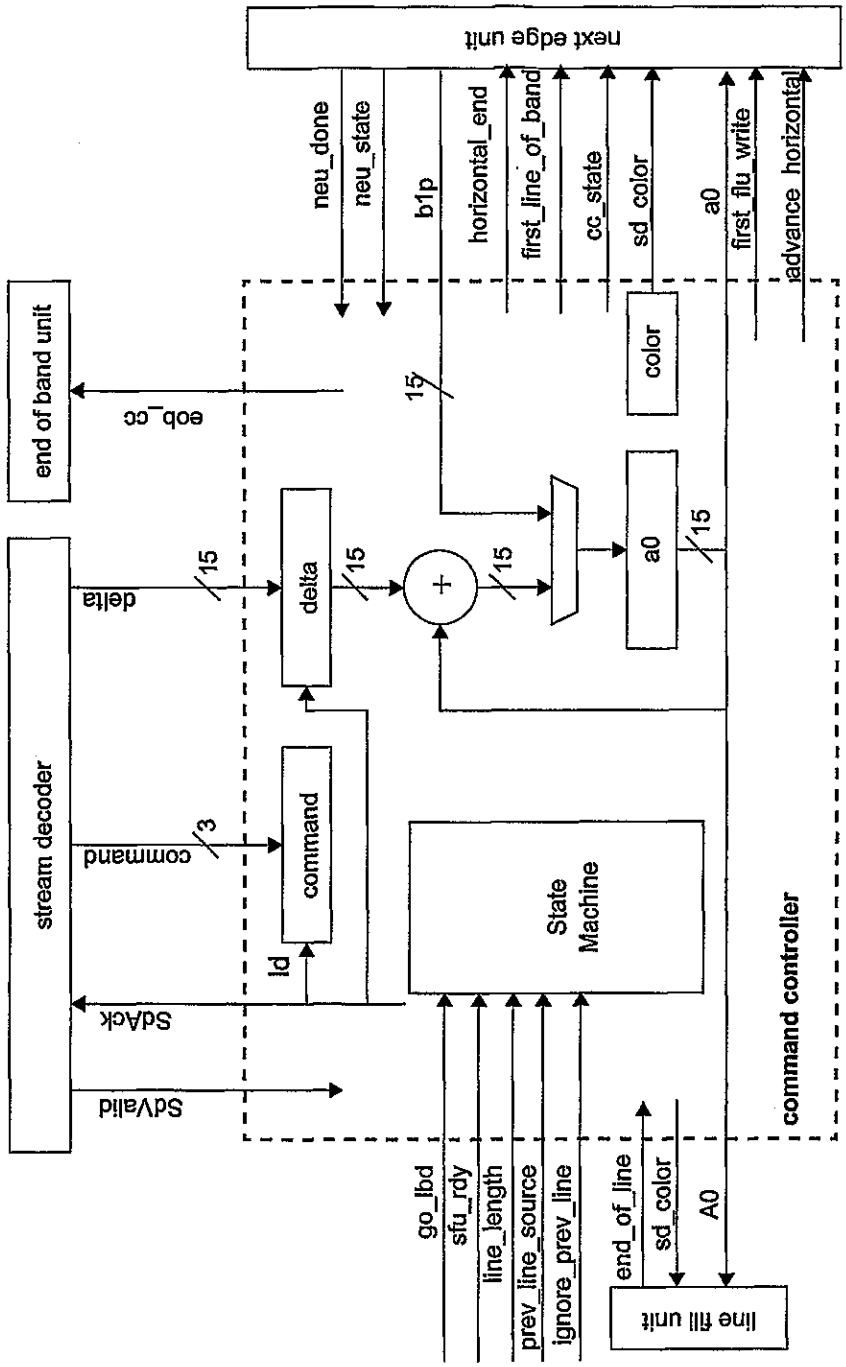

next edge unit
neu_done
neu_state
b1p
horizontal_end
first_line_of_band
cc_state
sd_color
a0
first_flu_write
advance_horizontal
end of band unit
eob_cc
color
15
delta
15
15
15
a0
stream decoder
delta
command
3
command
ld
State Machine
SdAck
SdValid
command controller
go_lbd
sfu_rdy
line_length
prev_line_source
ignore_prev_line
end_of_line
sd_color
A0
line fill unit

FIG. 150

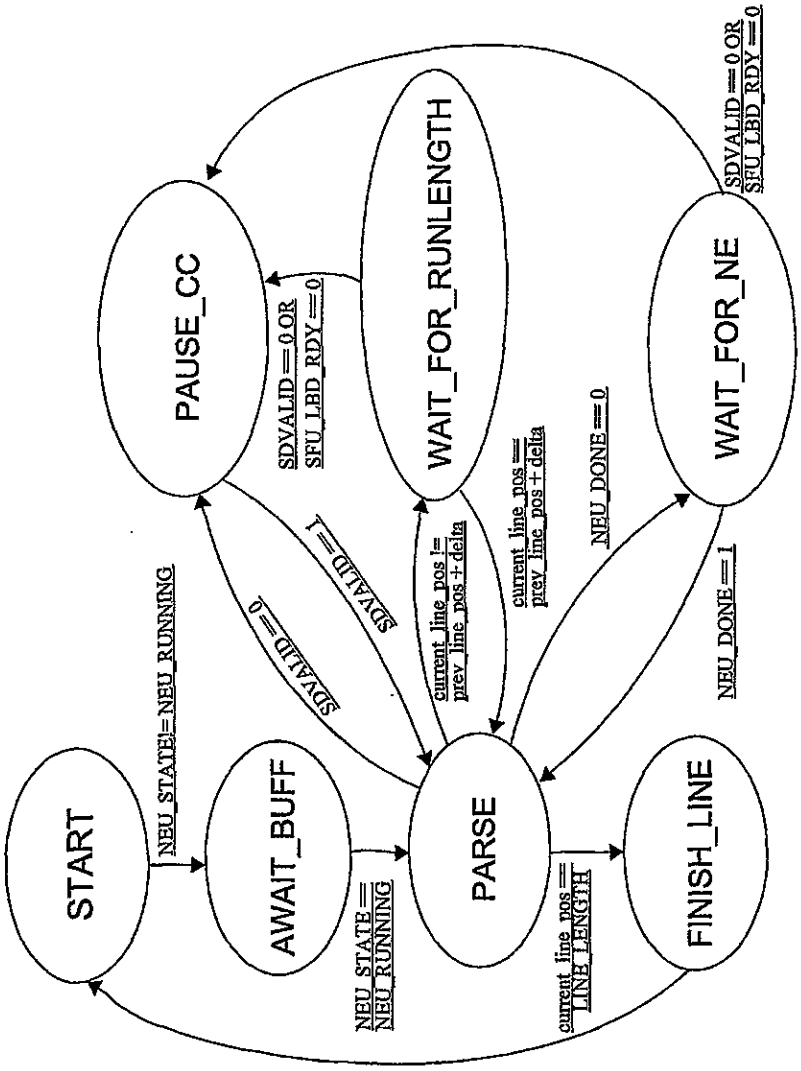

START
AWAIT_BUFF
PARSE
FINISH_LINE
PAUSE_CC
WAIT_FOR_RUNLENGTH
WAIT_FOR_NE
NEU_STATE!= NEU_RUNNING
NEU_STATE == NEU_RUNNING
SDVALID == 0
SDVALID == 1
SDVALID == 0 OR SFU_LBD_RDY == 0
current_line_pos != prev_line_pos + delta
current_line_pos == prev_line_pos + delta
NEU_DONE == 0
NEU_DONE == 1
current_line_pos == LINE_LENGTH
SDVALID == 0 OR SFU_LBD_RDY == 0

FIG. 151

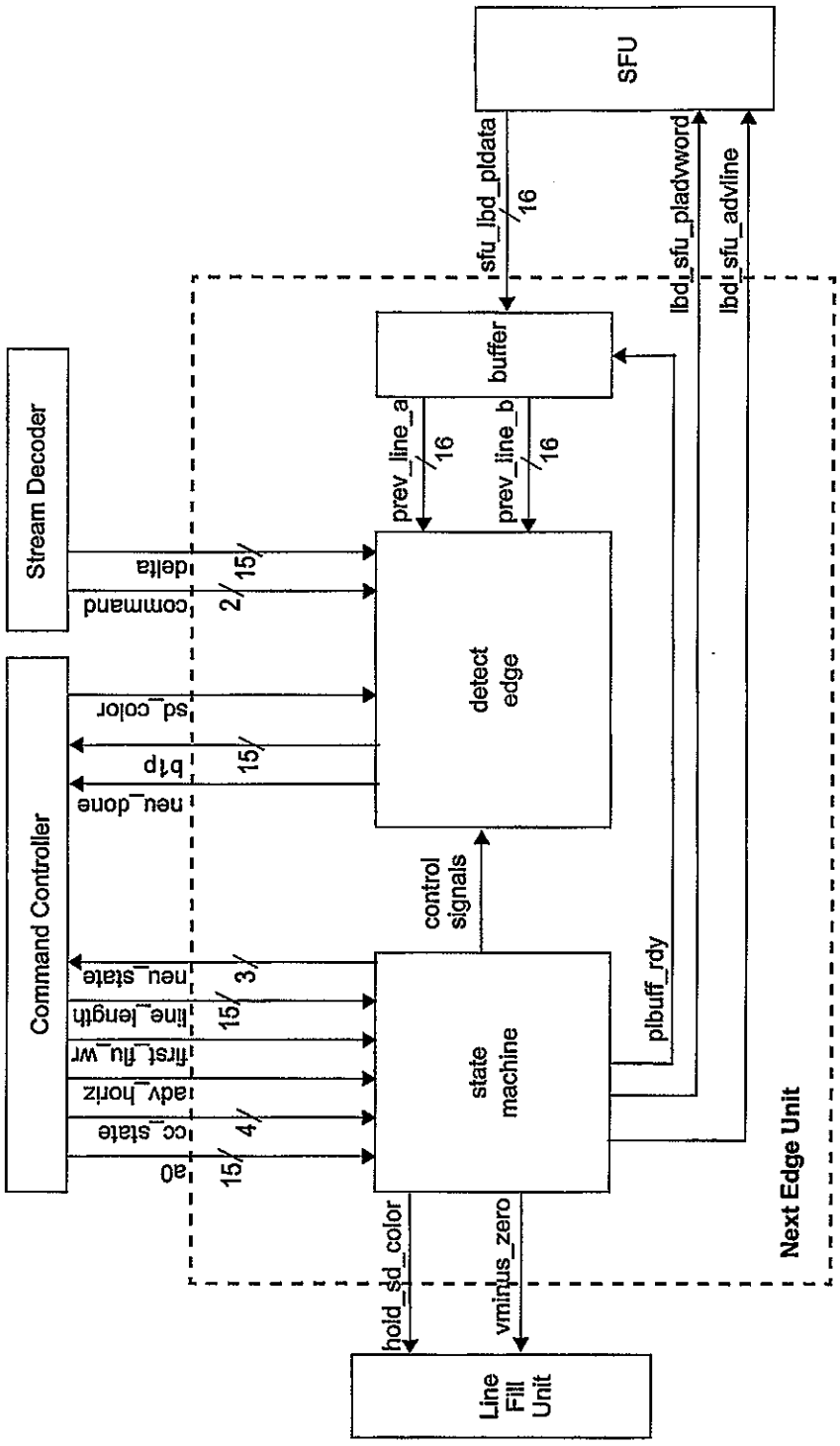

SFU
sfu_lbd_pldata
16
lbd_sfu_pladvword
lbd_sfu_advline
buffer
prev_line_a
16
prev_line_b
16
Stream Decoder
delta
15
command
2
detect edge
sd_color
b1p
15
neu_done
Command Controller
control signals
plbuff_rdy
neu_state
3
line_length
15
first_flu_wr
adv_horiz
cc_state
4
a0
15
state machine
Next Edge Unit
hold_sd_color
vminus_zero
Line Fill Unit

FIG. 152

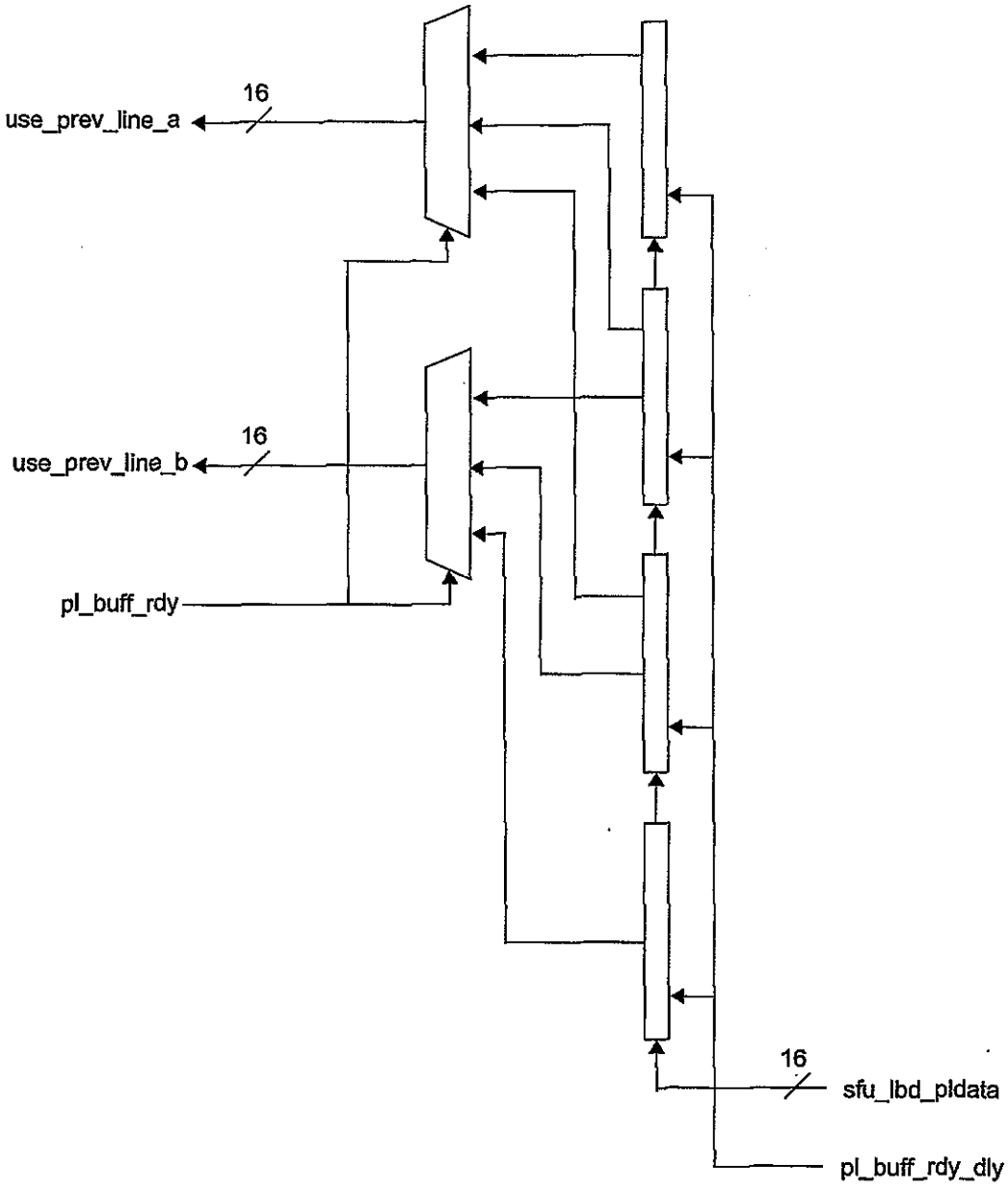

16
use_prev_line_a
16
use_prev_line_b
pl_buff_rdy
16
sfu_lbd_pldata
pl_buff_rdy_dly

FIG. 153

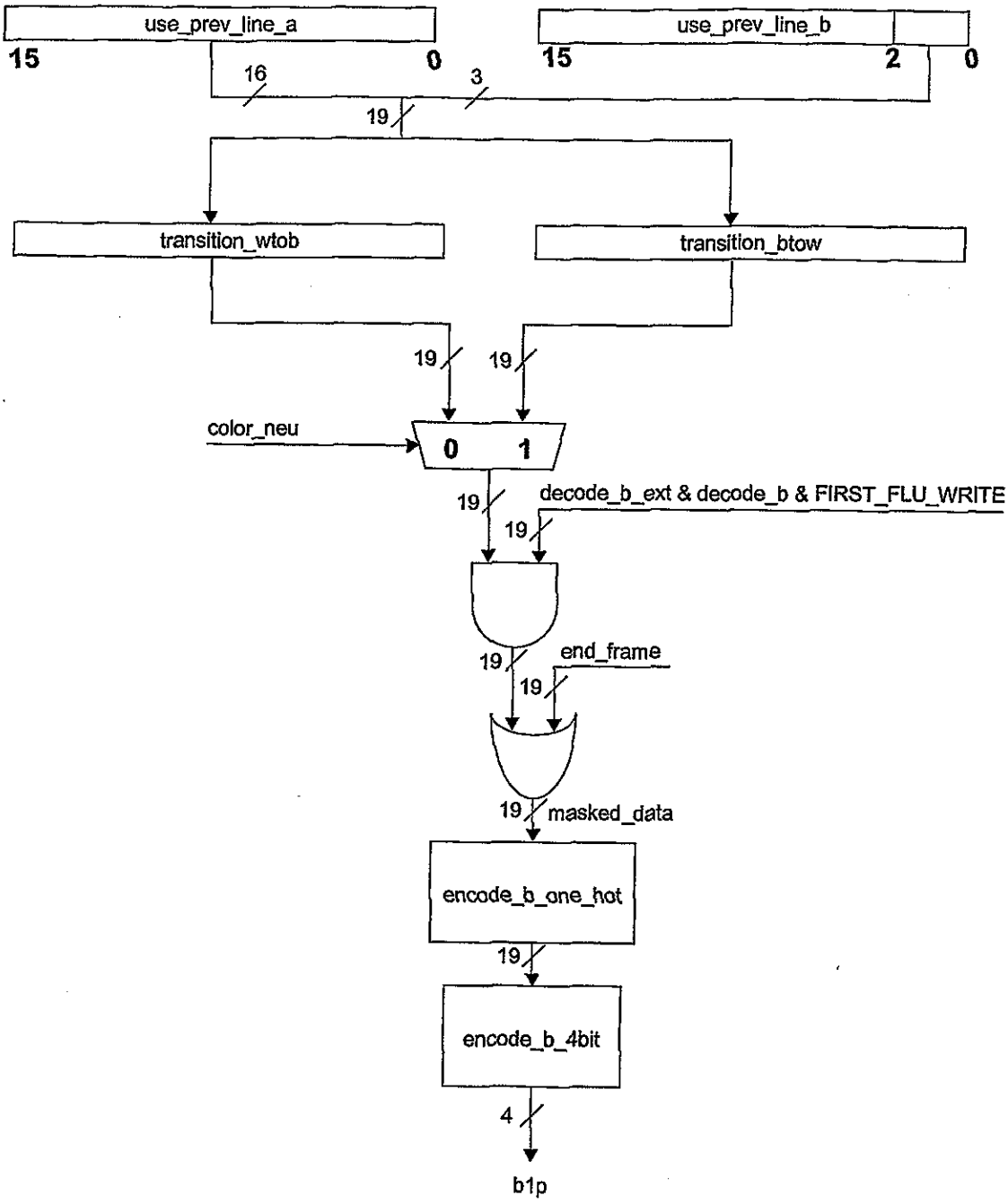

use_prev_line_a
15
0
use_prev_line_b
15
2
0
16
3
19
transition_wtob
transition_btow
19
19
color_neu
0
1
19
decode_b_ext & decode_b & FIRST_FLU_WRITE
19
19
end_frame
19
19
masked_data
encode_b_one_hot
19
encode_b_4bit
4
b1p

FIG. 154

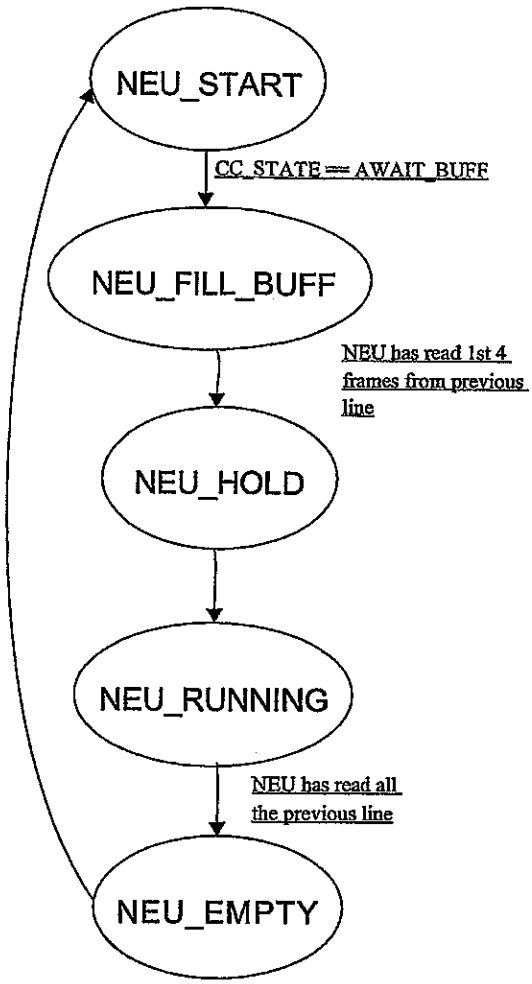

NEU_START
CC_STATE == AWAIT_BUFF
NEU_FILL_BUFF
NEU has read 1st 4 frames from previous line
NEU_HOLD
NEU_RUNNING
NEU has read all the previous line
NEU_EMPTY

FIG. 155

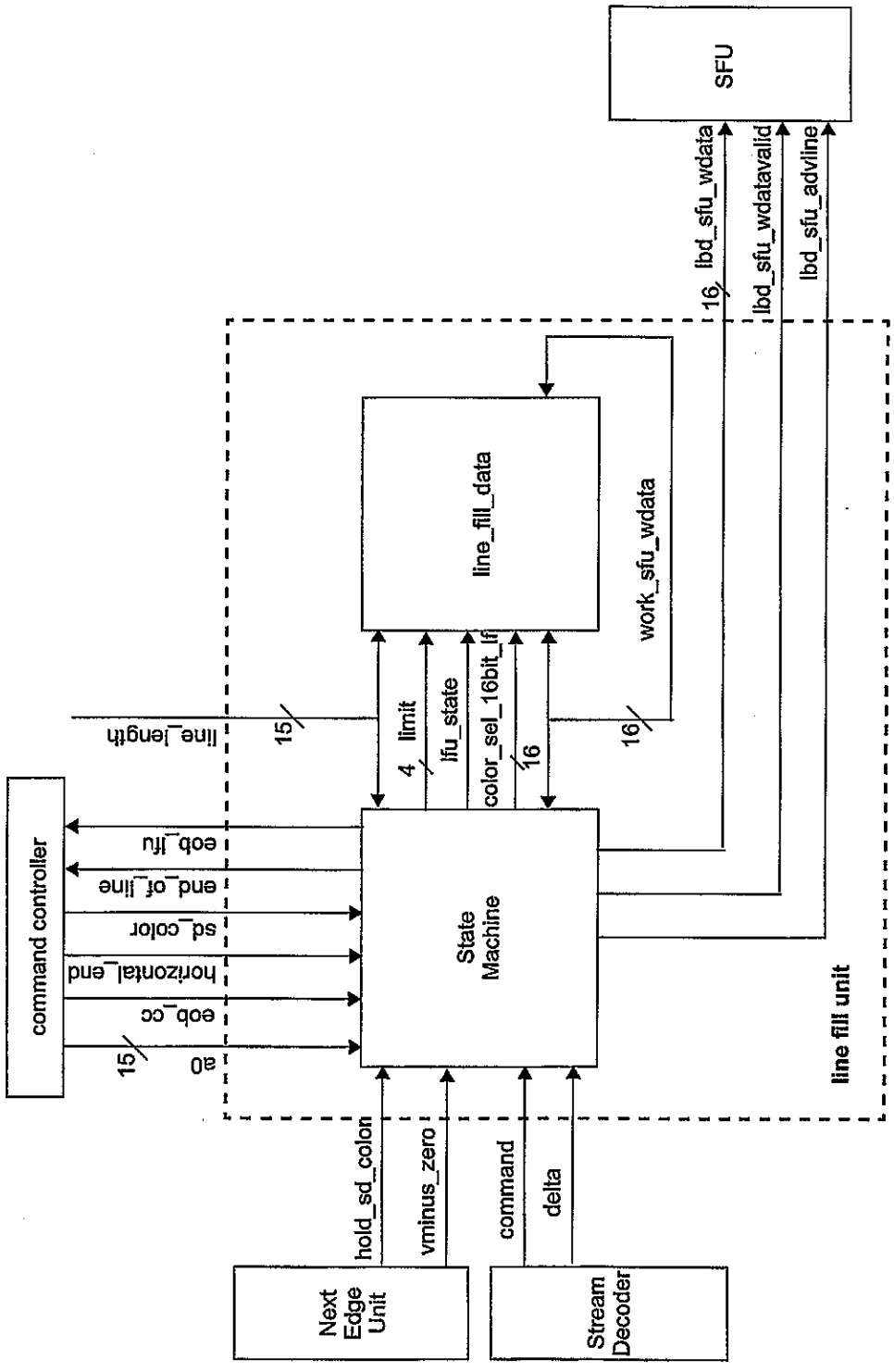

SFU
lbd_sfu_wdata
16
lbd_sfu_wdatavalid
lbd_sfu_advline
line_fill_data
work_sfu_wdata
line_length
15
4
limit
lfu_state
color_sel_16bit_lf
16
16
command controller
eob_lfu
end_of_line
sd_color
horizontal_end
eob_cc
15
a0
State Machine
line fill unit
hold_sd_colon
vminus_zero
command
delta
Next Edge Unit
Stream Decoder

FIG. 156

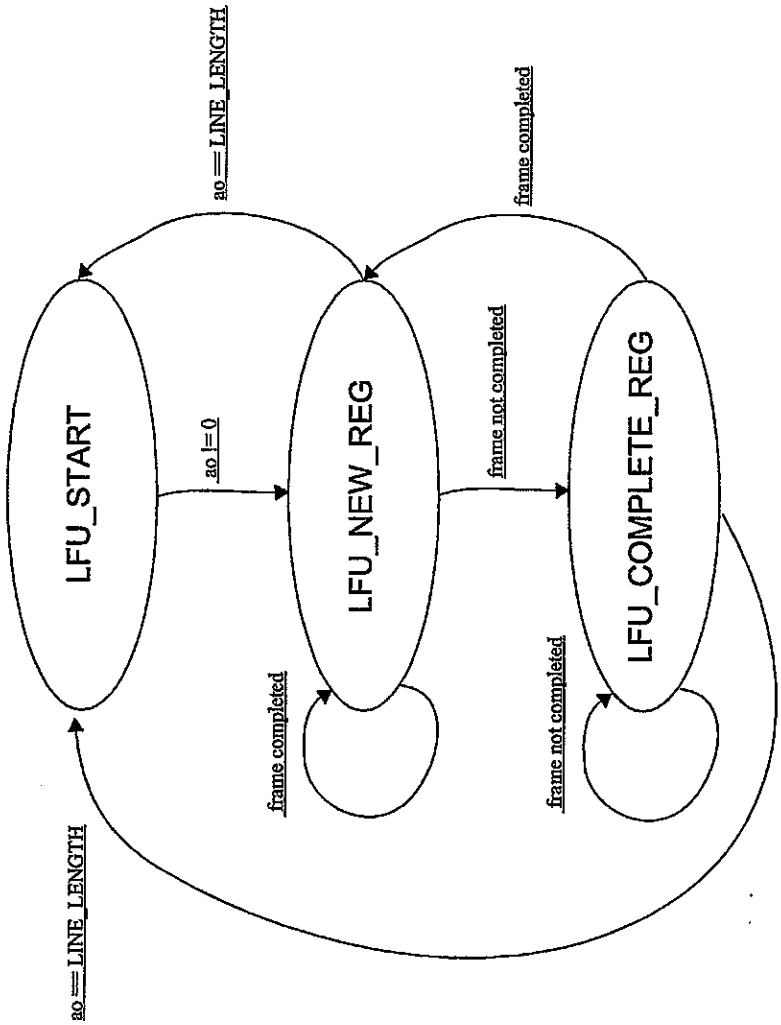

LFU_START
LFU_NEW_REG
LFU_COMPLETE_REG
ao == LINE_LENGTH
ao != 0
frame completed
frame not completed
frame completed
frame not completed
ao == LINE_LENGTH

FIG. 157

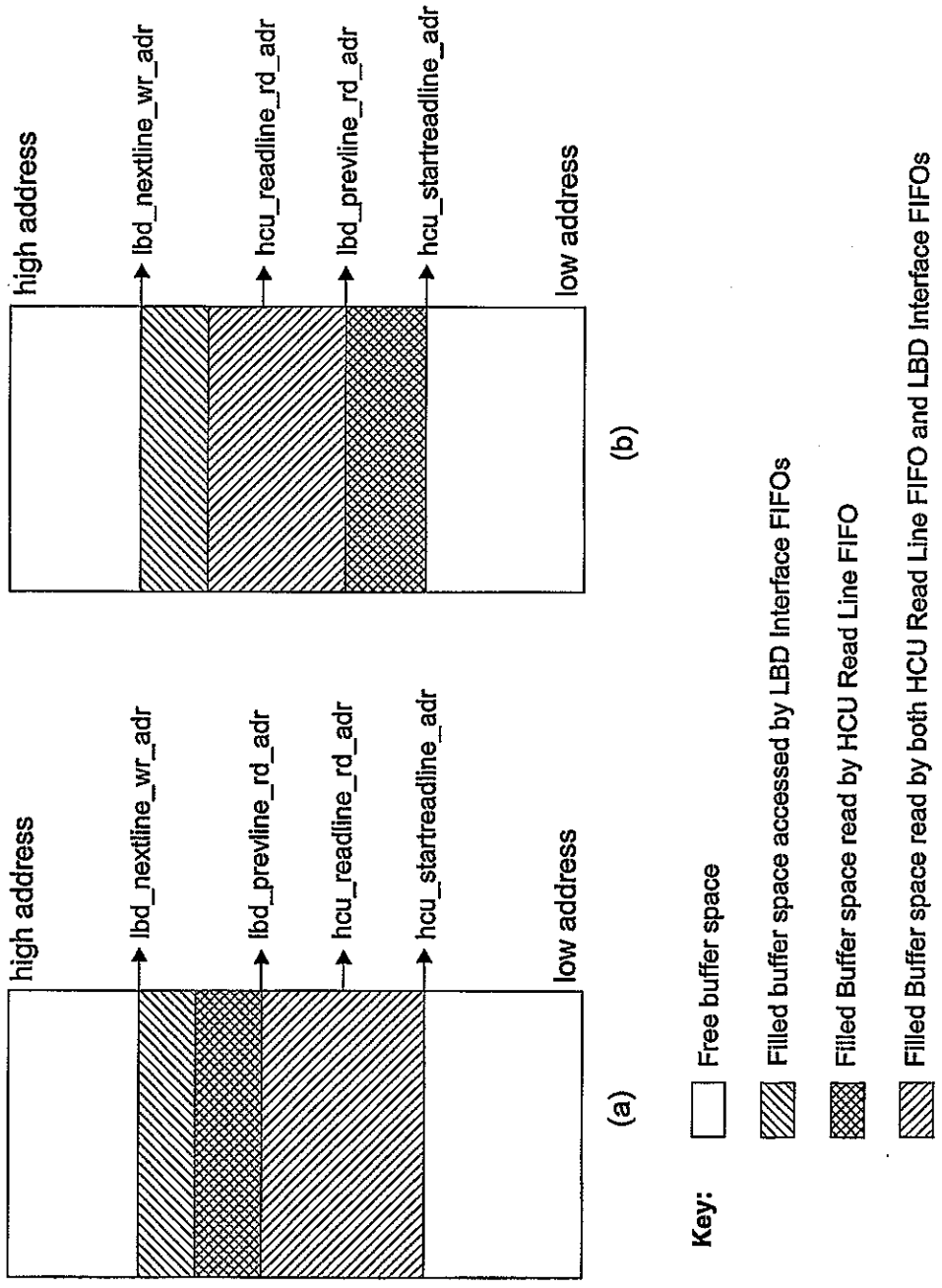

high address
lbd_nextline_wr_adr
lbd_previine_rd_adr
hcu_readline_rd_adr
hcu_startreadline_adr
low address
(a)
high address
lbd_nextline_wr_adr
hcu_readline_rd_adr
lbd_previine_rd_adr
hcu_startreadline_adr
low address
(b)
Key:
Free buffer space
Filled buffer space accessed by LBD Interface FIFOs
Filled Buffer space read by HCU Read Line FIFO
Filled Buffer space read by both HCU Read Line FIFO and LBD Interface FIFOs

FIG. 158

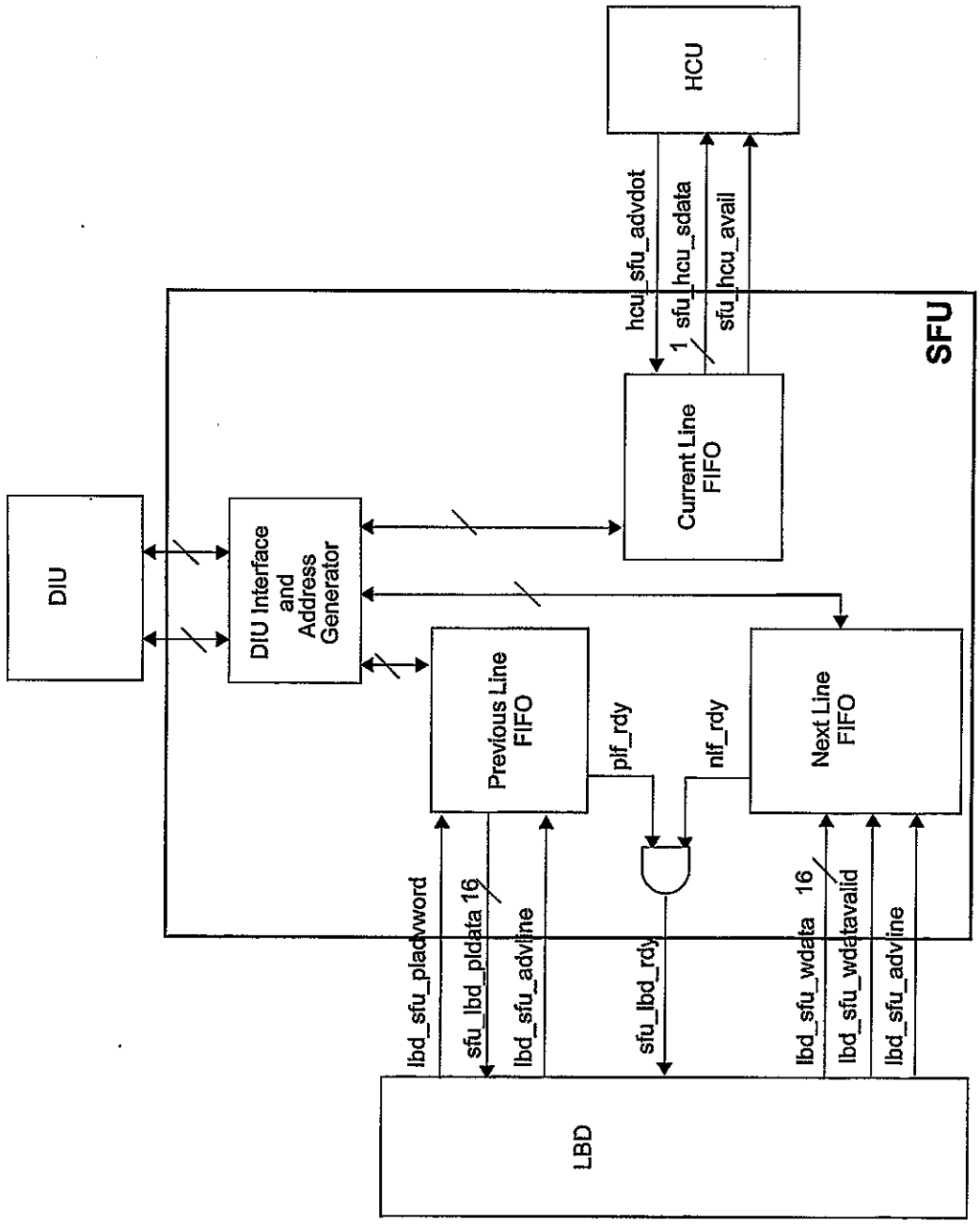

HCU
hcu_sfu_advdot
1 sfu_hcu_sdata
sfu_hcu_avail
SFU
DIU
DIU Interface and Address Generator
Current Line FIFO
Previous Line FIFO
plf_rdy
nlf_rdy
Next Line FIFO
lbd_sfu_pladvword
sfu_lbd_pldata 16
lbd_sfu_advline
sfu_lbd_rdy
lbd_sfu_wdata 16
lbd_sfu_wdatavalid
lbd_sfu_advline
LBD

FIG. 159

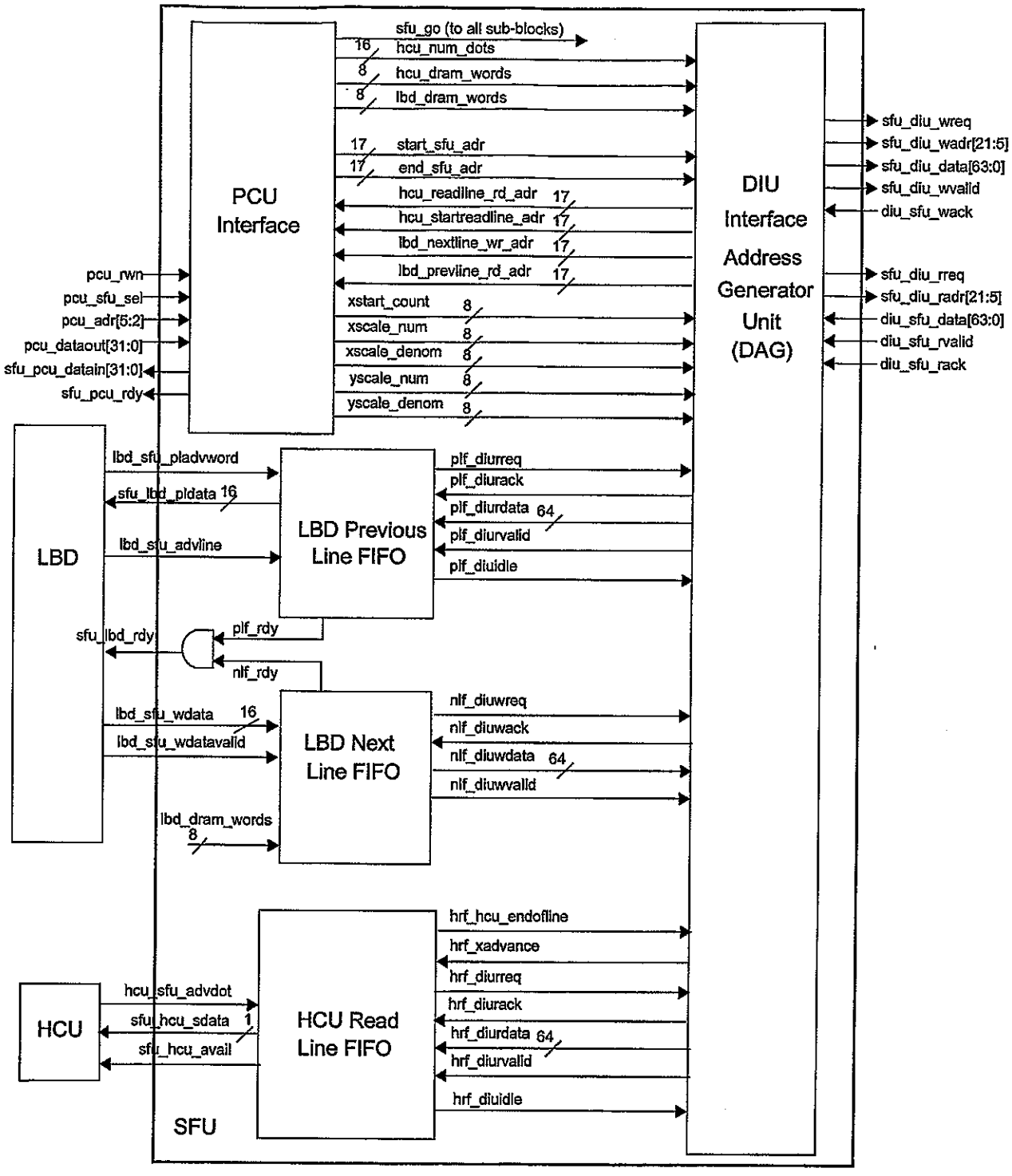

sfu_go (to all sub-blocks)
16 hcu_num_dots
8 hcu_dram_words
8 lbd_dram_words
17 start_sfu_adr
17 end_sfu_adr
hcu_readline_rd_adr 17
hcu_startreadline_adr 17
lbd_nextline_wr_adr 17
lbd_prevline_rd_adr 17
xstart_count 8
xscale_num 8
xscale_denom 8
yscale_num 8
yscale_denom 8
PCU Interface
pcu_rwn
pcu_sfu_sel
pcu_adr[5:2]
pcu_dataout[31:0]
sfu_pcu_datain[31:0]
sfu_pcu_rdy
DIU Interface Address Generator Unit (DAG)
sfu_diu_wreq
sfu_diu_wadr[21:5]
sfu_diu_data[63:0]
sfu_diu_wvalid
diu_sfu_wack
sfu_diu_rreq
sfu_diu_radr[21:5]
diu_sfu_data[63:0]
diu_sfu_rvalid
diu_sfu_rack
LBD
lbd_sfu_pladvword
sfu_lbd_pldata 16
lbd_sfu_advline
LBD Previous Line FIFO
plf_diurreq
plf_diurack
plf_diurdata 64
plf_diurvalid
plf_diuidle
sfu_lbd_rdy
plf_rdy
nlf_rdy
lbd_sfu_wdata 16
lbd_sfu_wdatavalid
LBD Next Line FIFO
nlf_diuwreq
nlf_diuwack
nlf_diuwdata 64
nlf_diuwvalid
lbd_dram_words 8
HCU
hcu_sfu_advdot
sfu_hcu_sdata 1
sfu_hcu_avail
HCU Read Line FIFO
hrf_hcu_endofline
hrf_xadvance
hrf_diurreq
hrf_diurack
hrf_diurdata 64
hrf_diurvalid
hrf_diuidle
SFU

FIG. 160

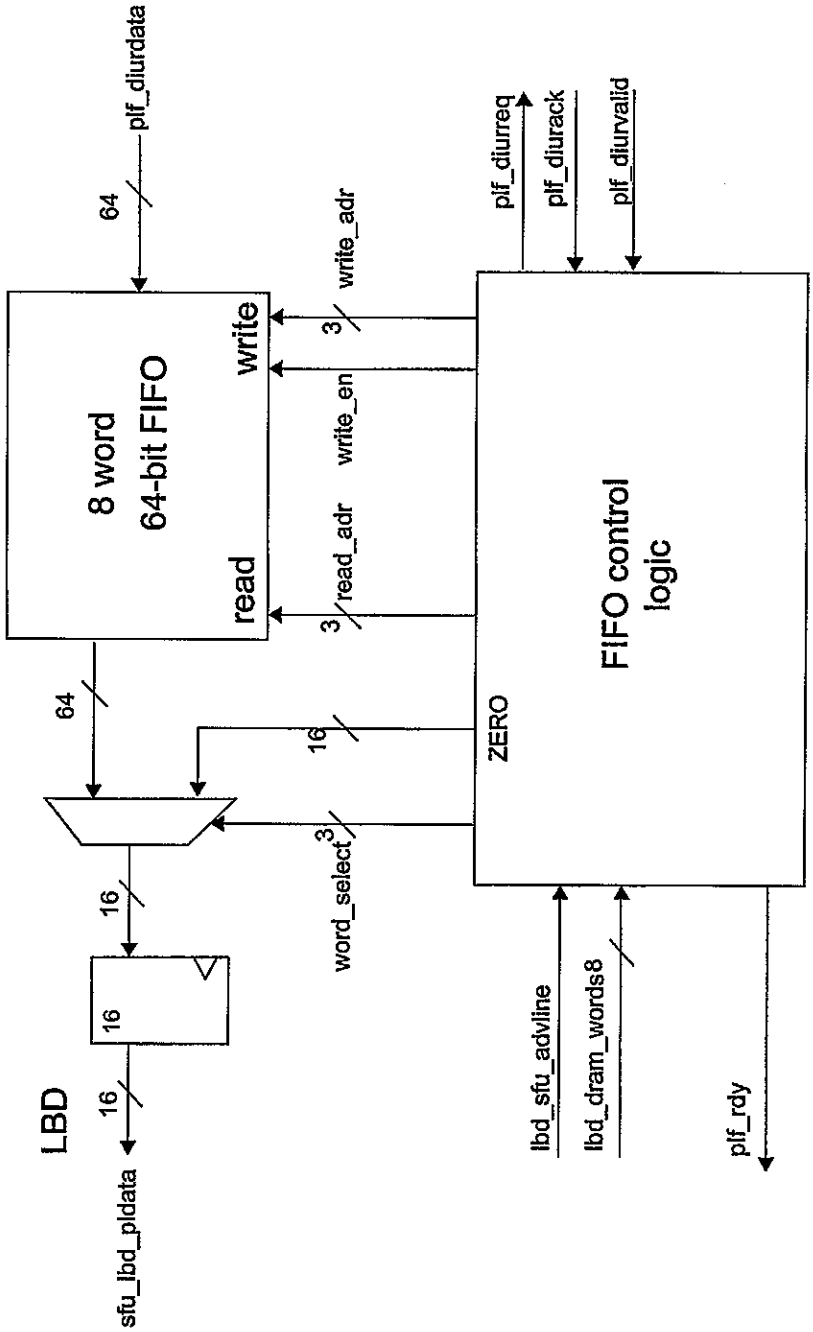

plf_diurdata
64
write_adr
3
plf_diurreq
plf_diurack
plf_diurvalid
8 word
64-bit FIFO
write
read
write_en
read_adr
3
64
16
ZERO
FIFO control
logic
3
word_select
16
16
16
LBD
sfu_lbd_pldata
lbd_sfu_advline
lbd_dram_words8
plf_rdy

FIG. 161

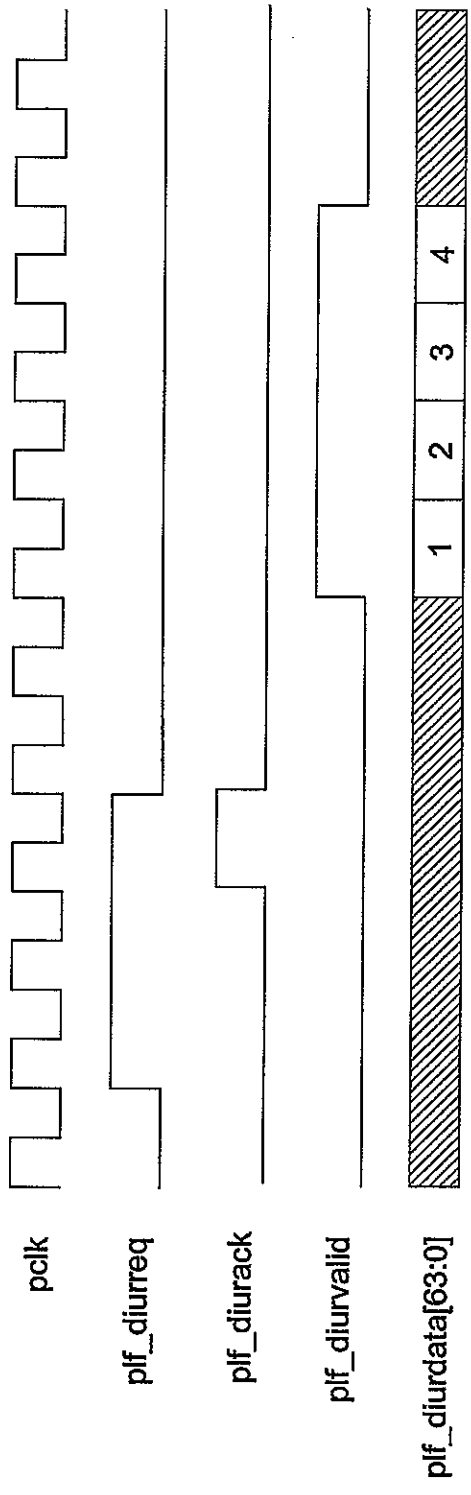

pclk
plf_diurreq
plf_diurack
plf_diurvalid
plf_diurdata[63:0]
1
2
3
4

FIG. 162

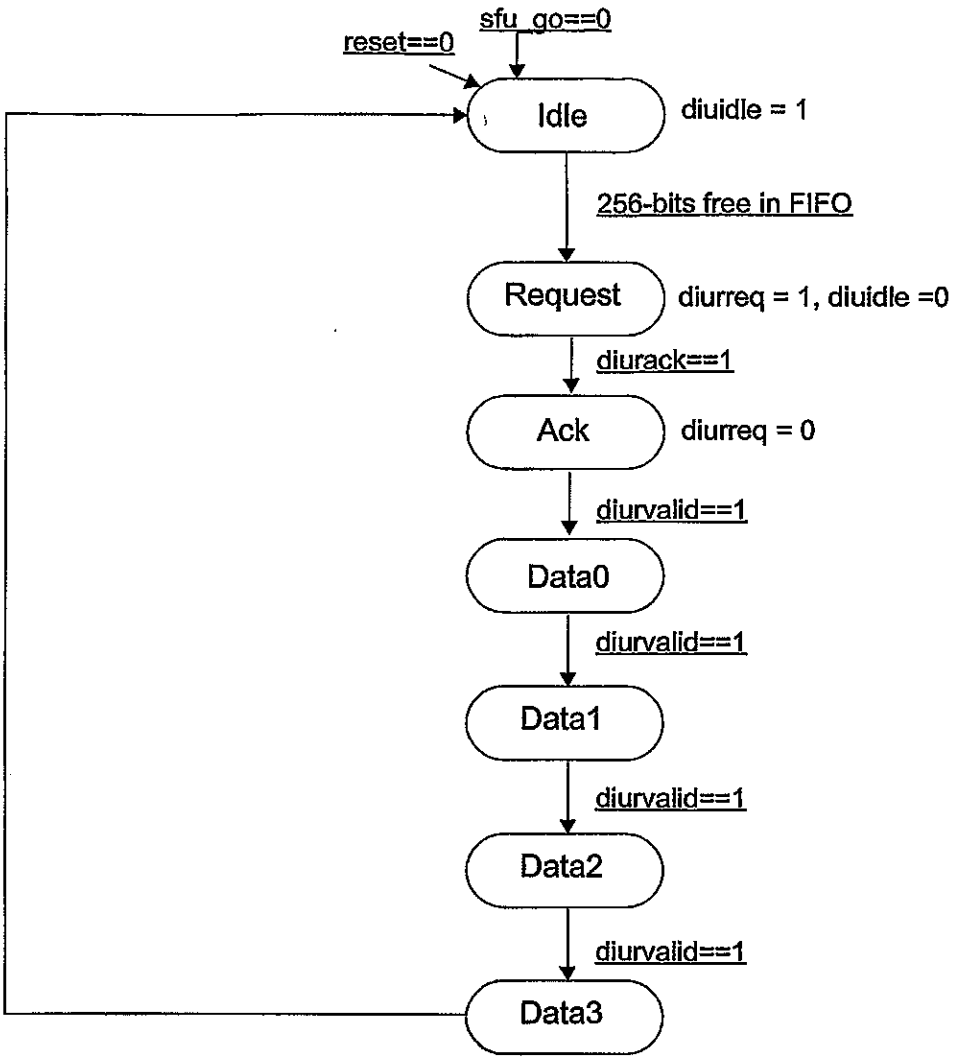

sfu_go==0
reset==0
Idle
diuidle = 1
256-bits free in FIFO
Request
diurreq = 1, diuidle =0
diurack==1
Ack
diurreq = 0
diurvalid==1
Data0
diurvalid==1
Data1
diurvalid==1
Data2
diurvalid==1
Data3

FIG. 163

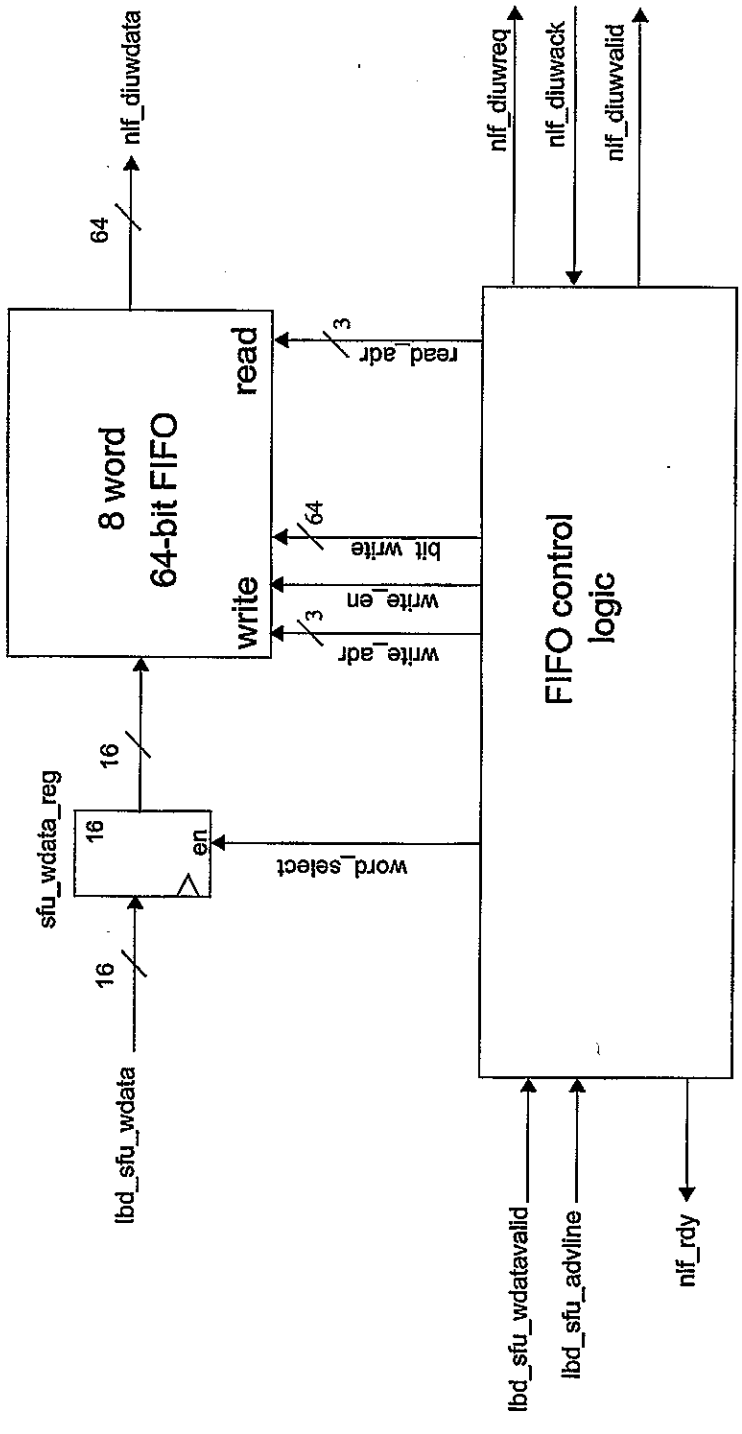

nlf_diuwdata
64
nlf_diuwreq
nlf_diuwack
nlf_diuwvalid
8 word
64-bit FIFO
read
write
read_adr
3
bit_write
64
write_en
write_adr
3
FIFO control
logic
sfu_wdata_reg
16
16
en
word_select
16
lbd_sfu_wdata
lbd_sfu_wdatavalid
lbd_sfu_advline
nlf_rdy

FIG. 164

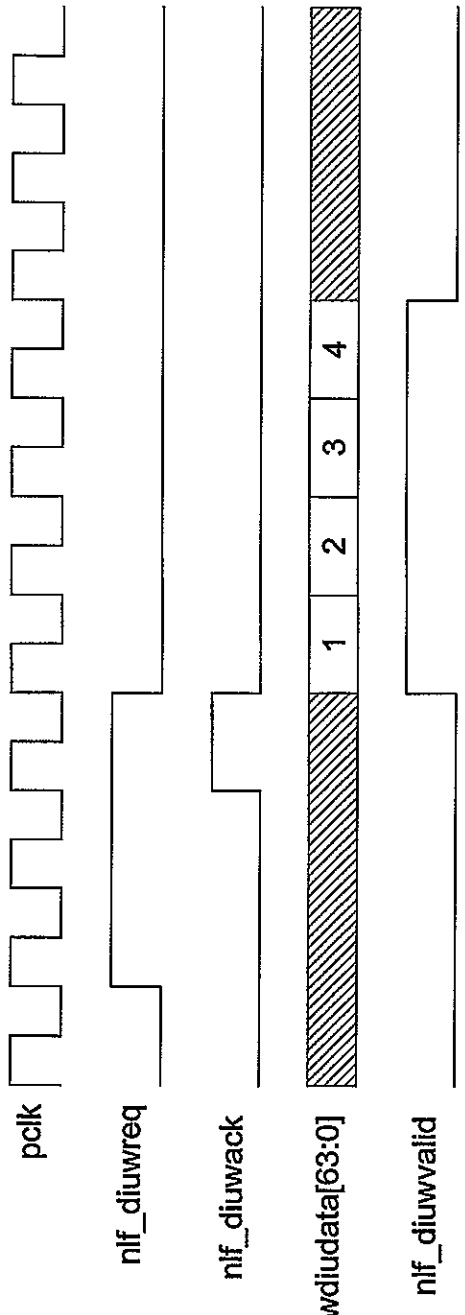

pclk
nlf_diuwreq
nlf_diuwack
nlf_wdiudata[63:0]
1
2
3
4
nlf_diuwvalid

FIG. 165

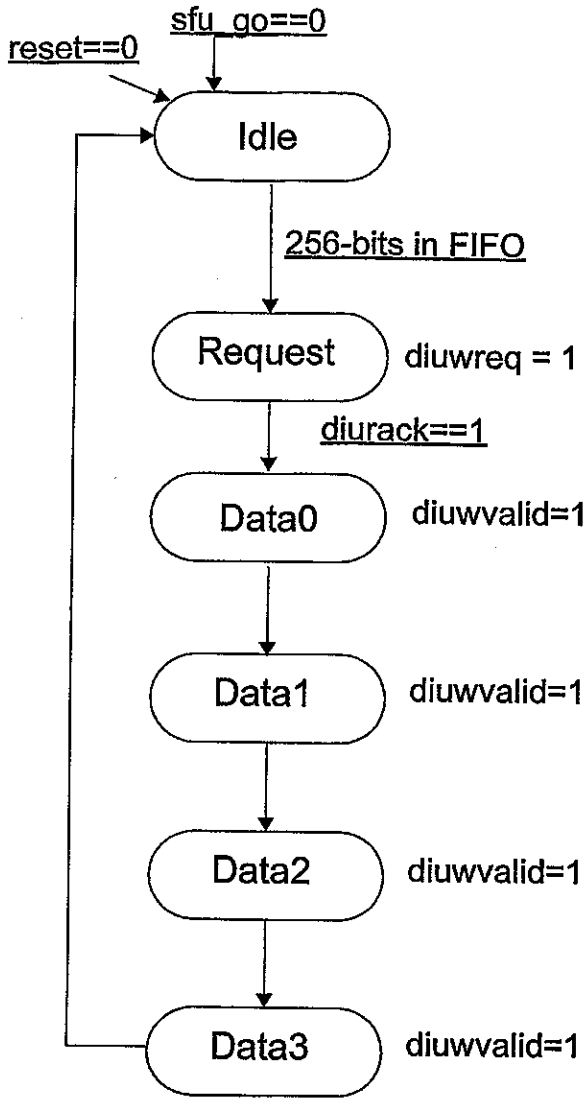

sfu_go==0
reset==0
Idle
256-bits in FIFO
Request
diuwreq = 1
diurack==1
Data0
diuwvalid=1
Data1
diuwvalid=1
Data2
diuwvalid=1
Data3
diuwvalid=1

FIG. 166

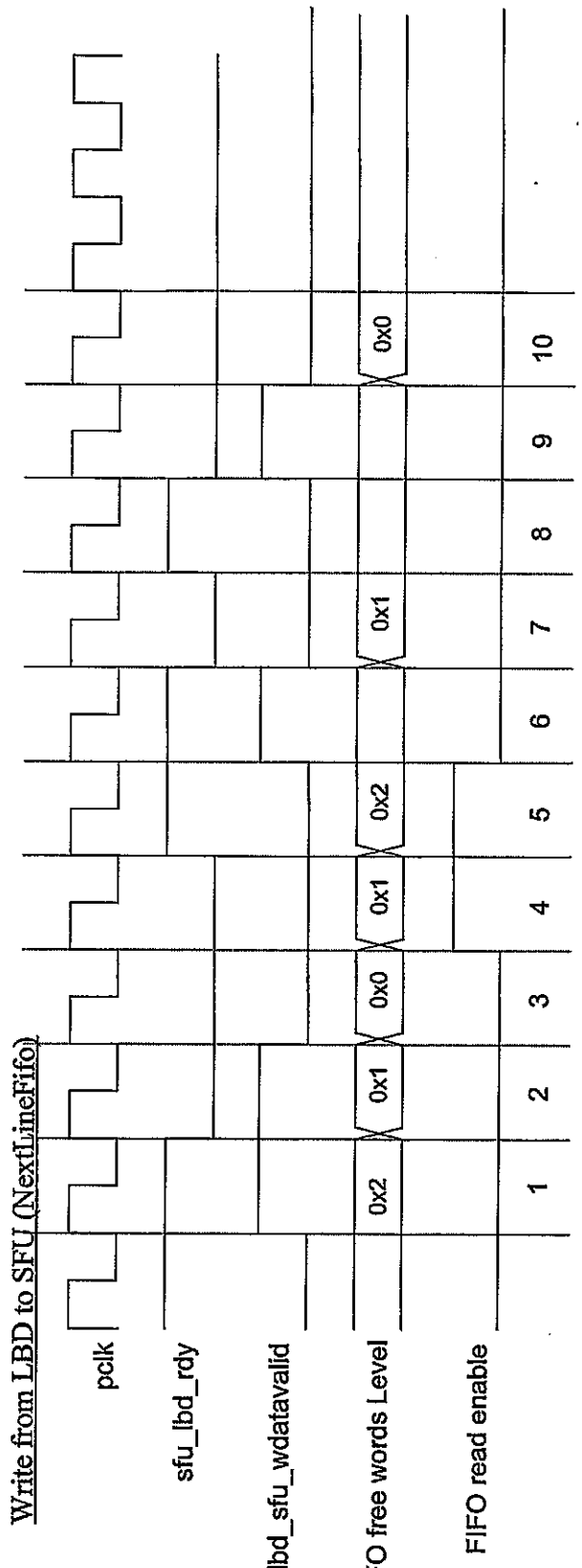

Write from LBD to SFU (NextLineFifo)
pclk
sfu_lbd_rdy
lbd_sfu_wdatavalid
FIFO free words Level
FIFO read enable
0x2
0x1
0x0
0x1
0x2
0x1
0x0
1
2
3
4
5
6
7
8
9
10

FIG. 167

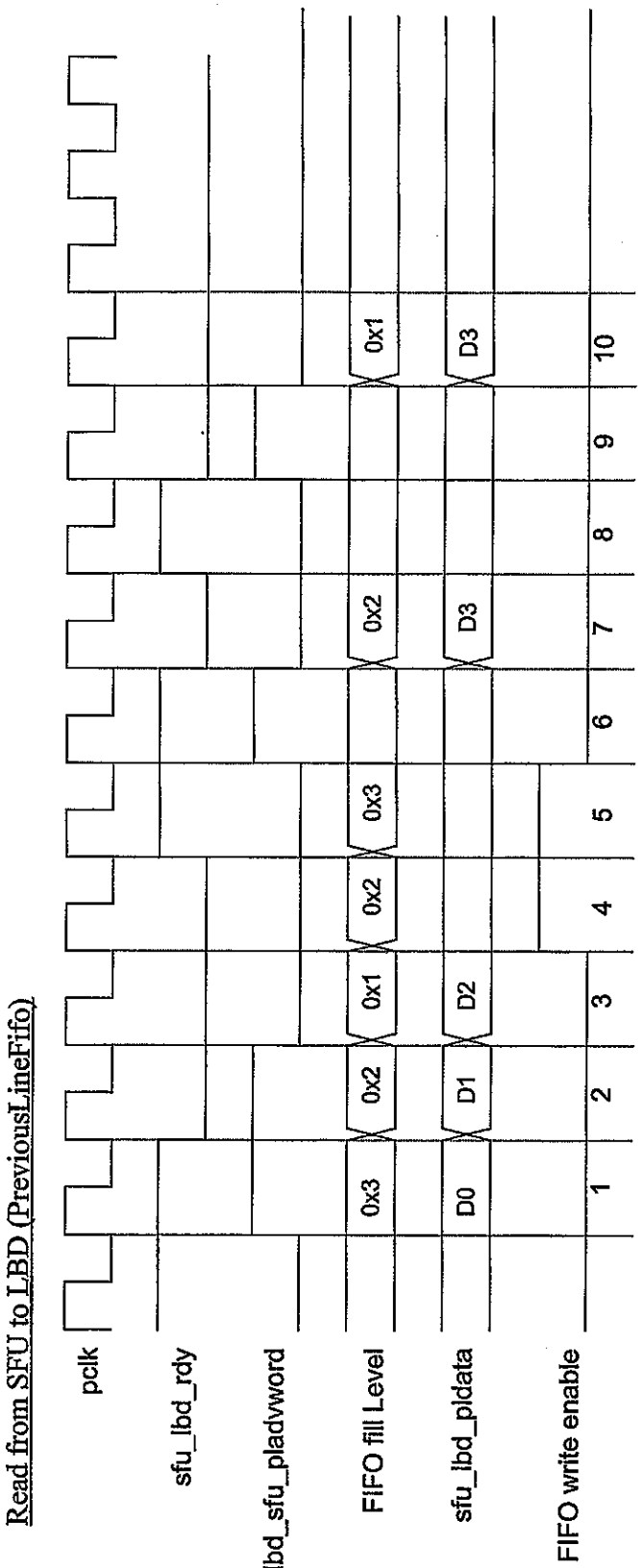

Read from SFU to LBD (PreviousLineFifo)
pclk
sfu_lbd_rdy
lbd_sfu_pladvword
FIFO fill Level
sfu_lbd_pldata
FIFO write enable
0x3
0x2
0x1
0x2
0x3
0x2
0x1
D0
D1
D2
D3
D3
1
2
3
4
5
6
7
8
9
10

FIG. 168

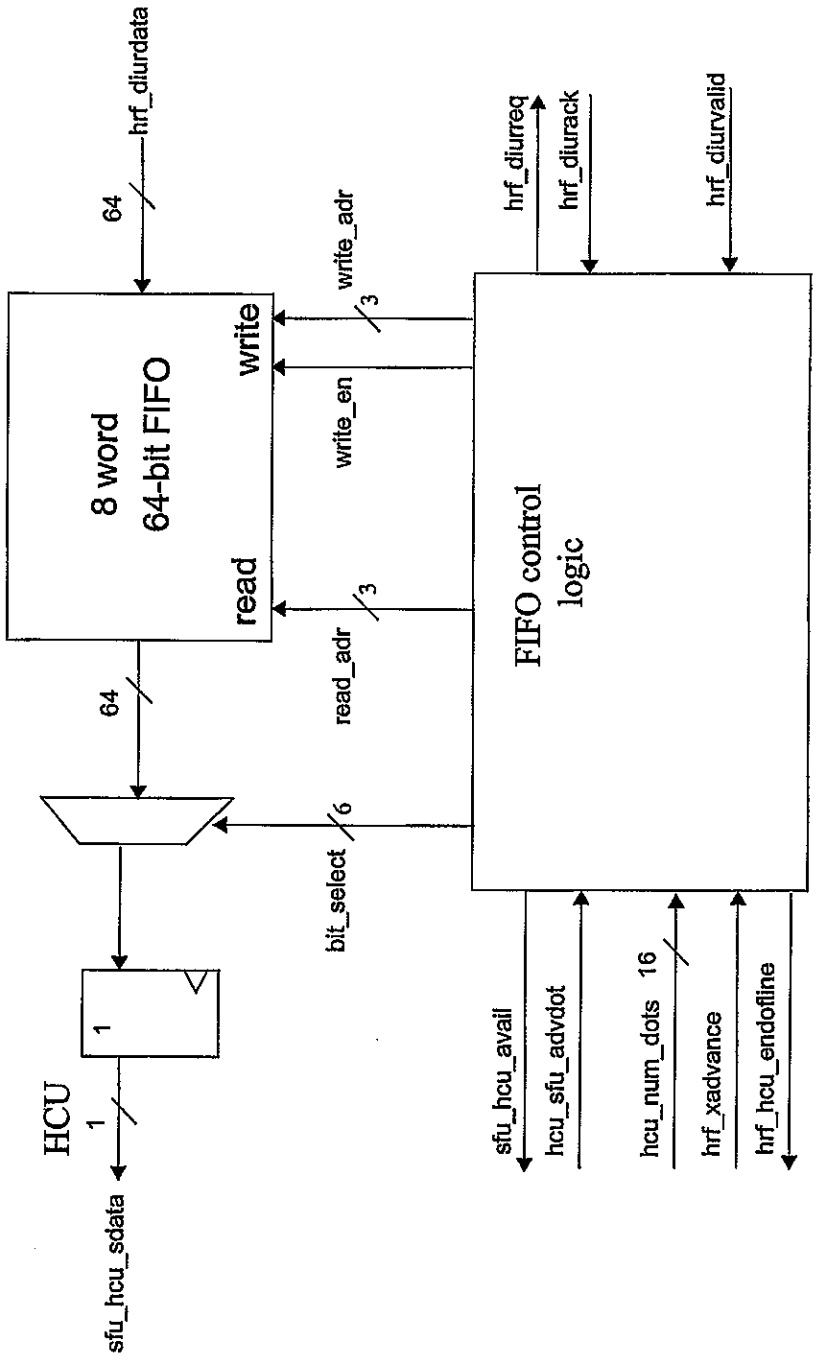

hrf_diurdata
64
write_adr
3
hrf_diurreq
hrf_diurack
hrf_diurvalid
8 word
64-bit FIFO
write
read
write_en
3
FIFO control
logic
64
read_adr
6
bit_select
1
HCU
1
sfu_hcu_sdata
sfu_hcu_avail
hcu_sfu_advdot
hcu_num_dots
16
hrf_xadvance
hrf_hcu_endofline

FIG. 169

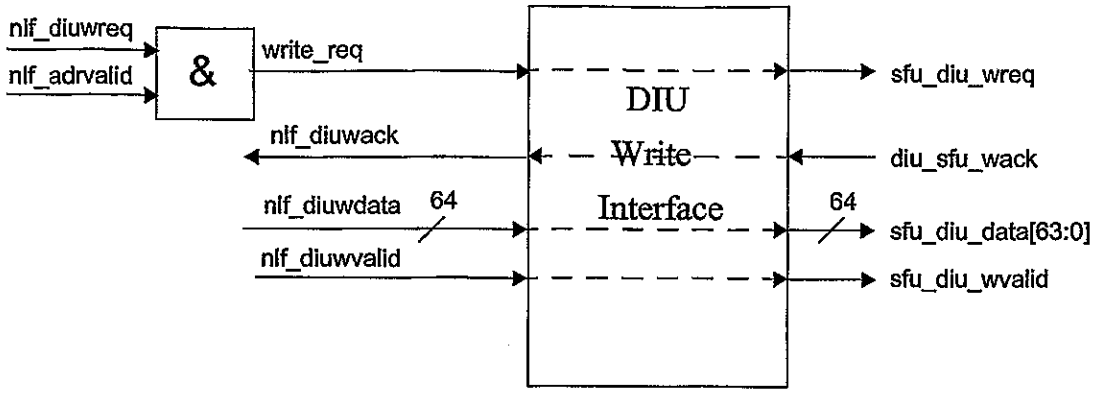


FIG. 170

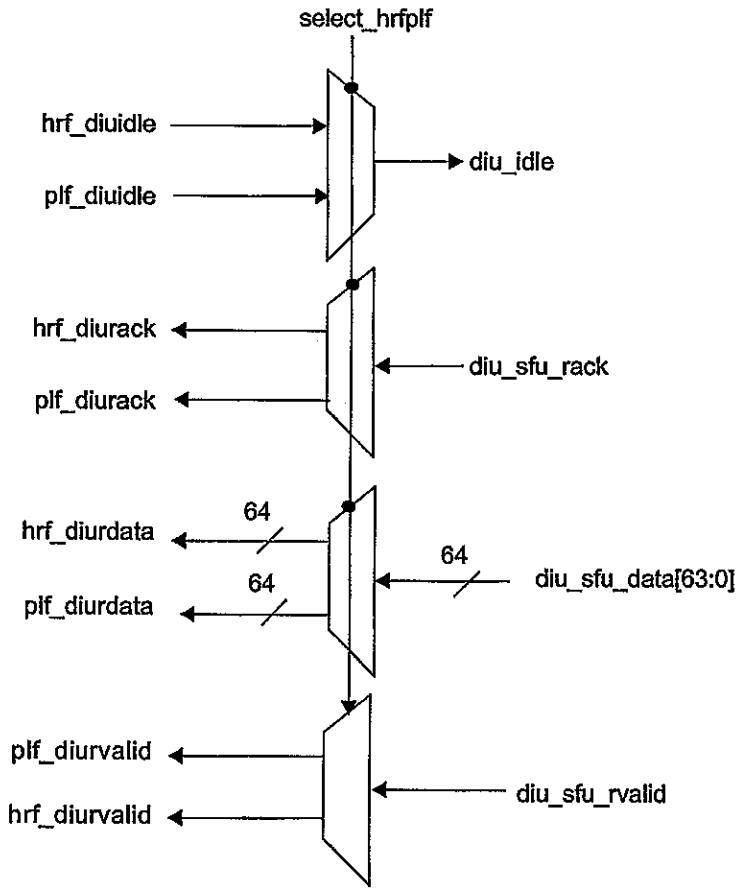


FIG. 171

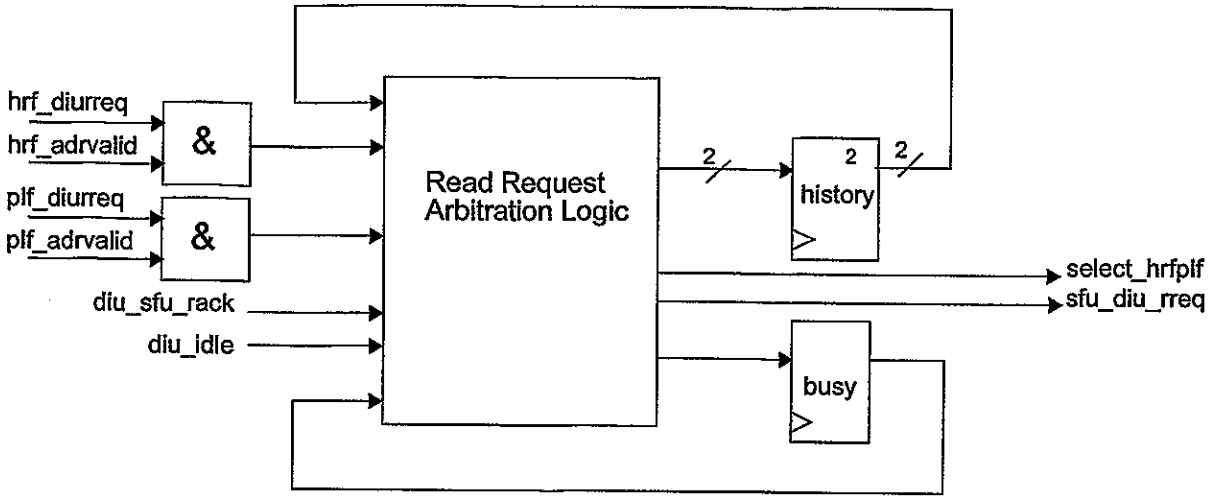


FIG. 172

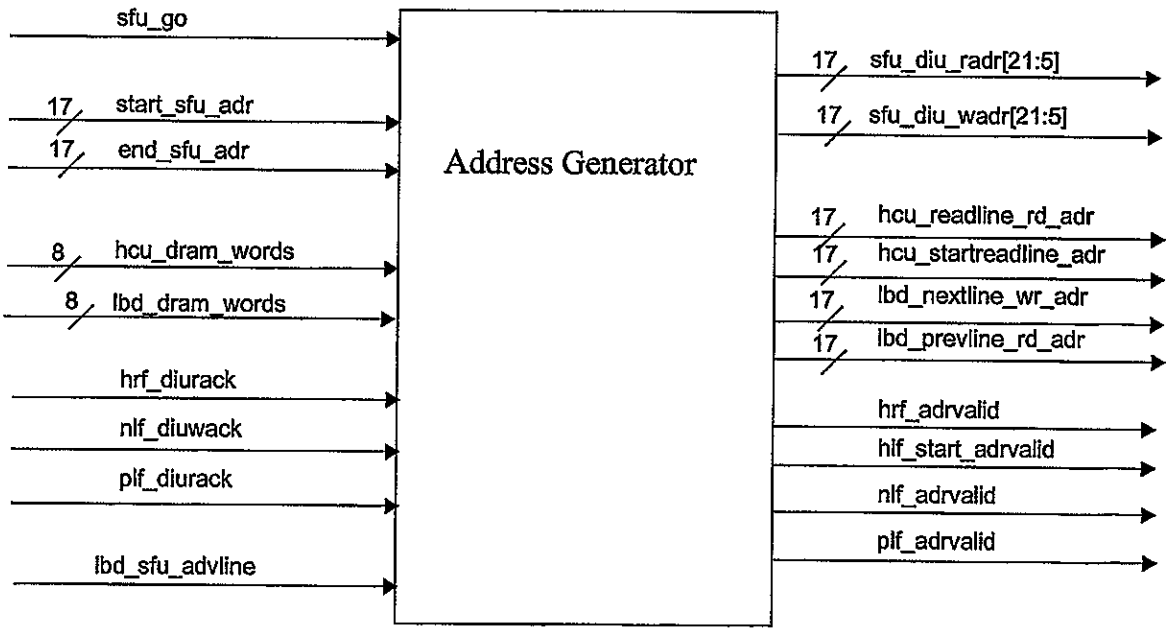


FIG. 173

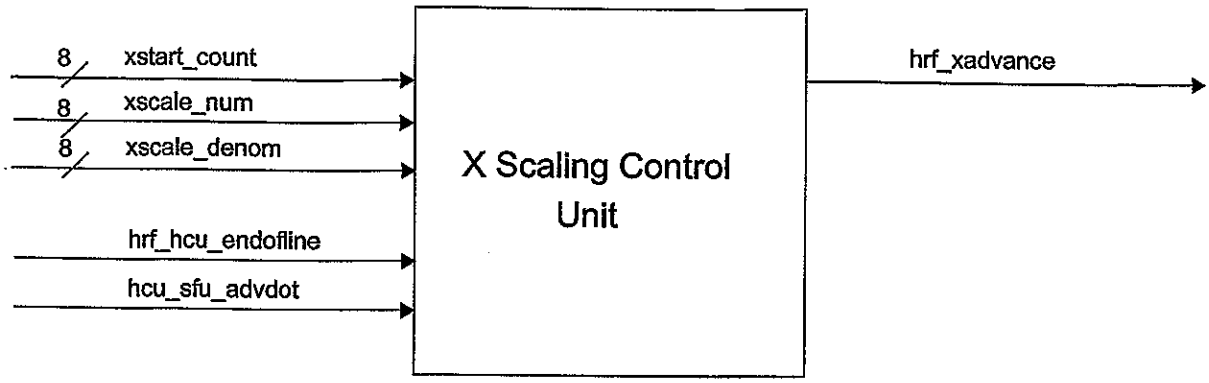


FIG. 174

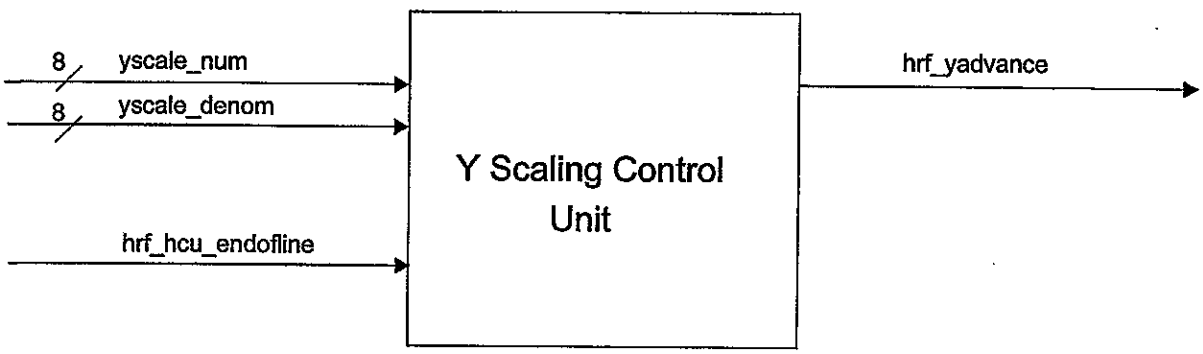


FIG. 175

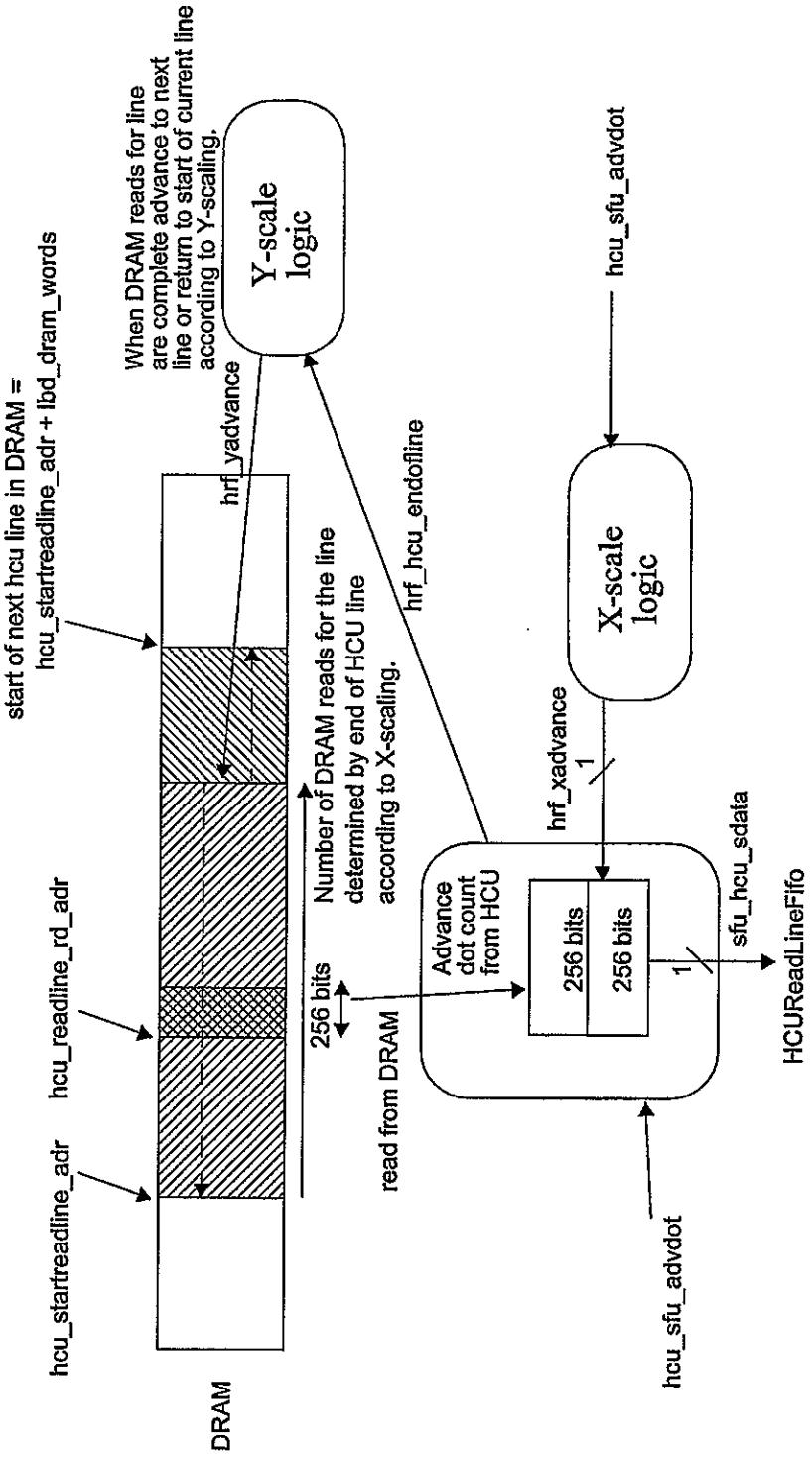

start of next hcu line in DRAM =
hcu_startreadline_adr + lbd_dram_words
When DRAM reads for line are complete advance to next line or return to start of current line according to Y-scaling.
Y-scale logic
hrf_yadvance
hrf_hcu_endofline
hcu_sfu_advdot
X-scale logic
hrf_xadvance
1
hcu_readline_rd_adr
hcu_startreadline_adr
256 bits
Number of DRAM reads for the line determined by end of HCU line according to X-scaling.
read from DRAM
Advance dot count from HCU
256 bits
256 bits
1
sfu_hcu_sdata
HCUReadLineFifo
hcu_sfu_advdot
DRAM

FIG. 176

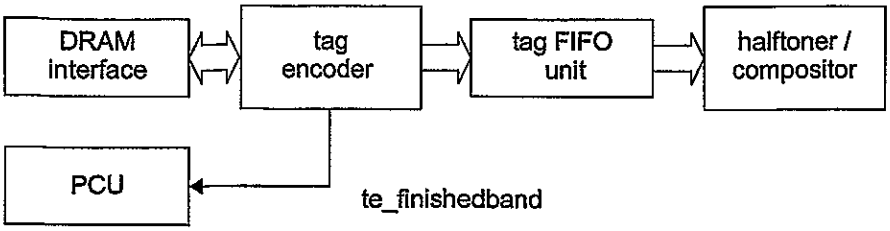


FIG. 177

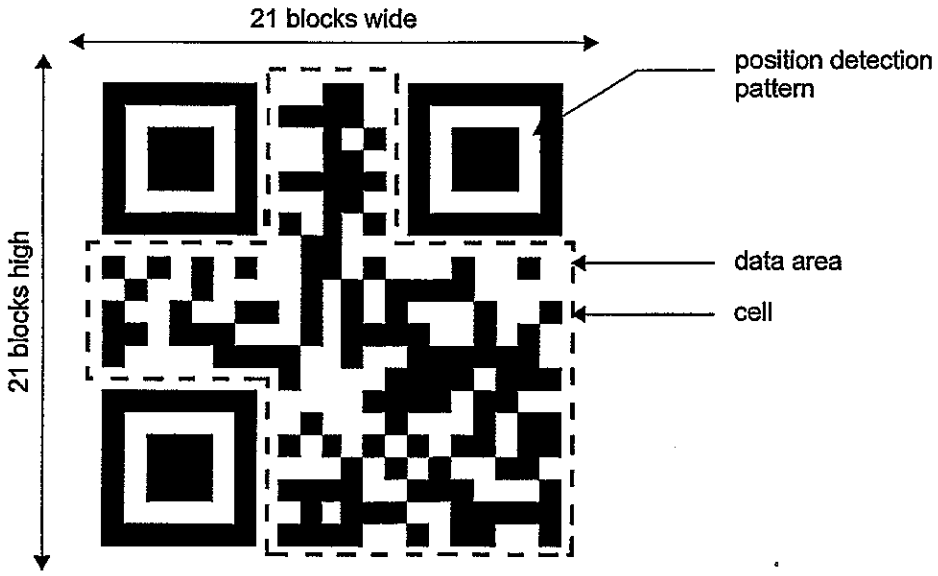
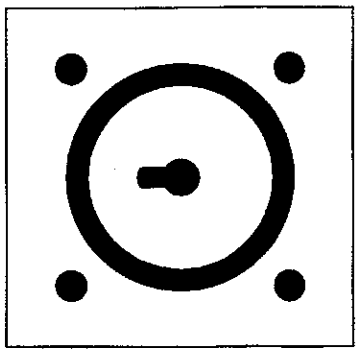
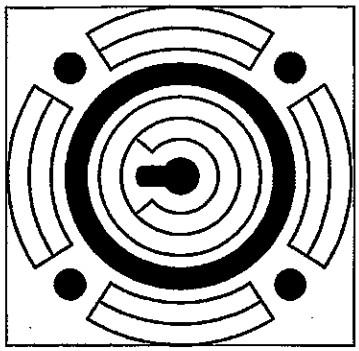


FIG. 178



(a) Netpage tag background pattern



(b) Netpage tag showing data area

FIG. 179

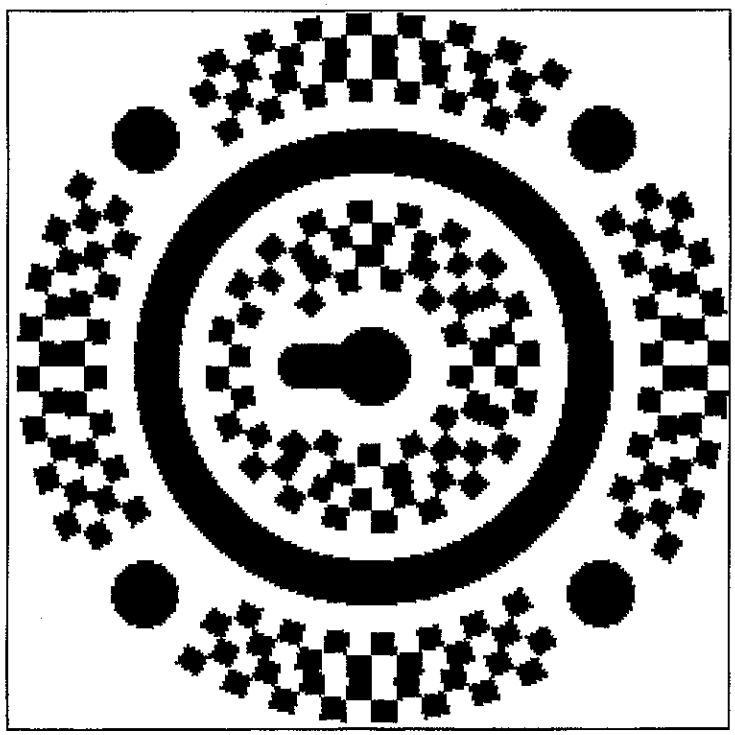


FIG. 180

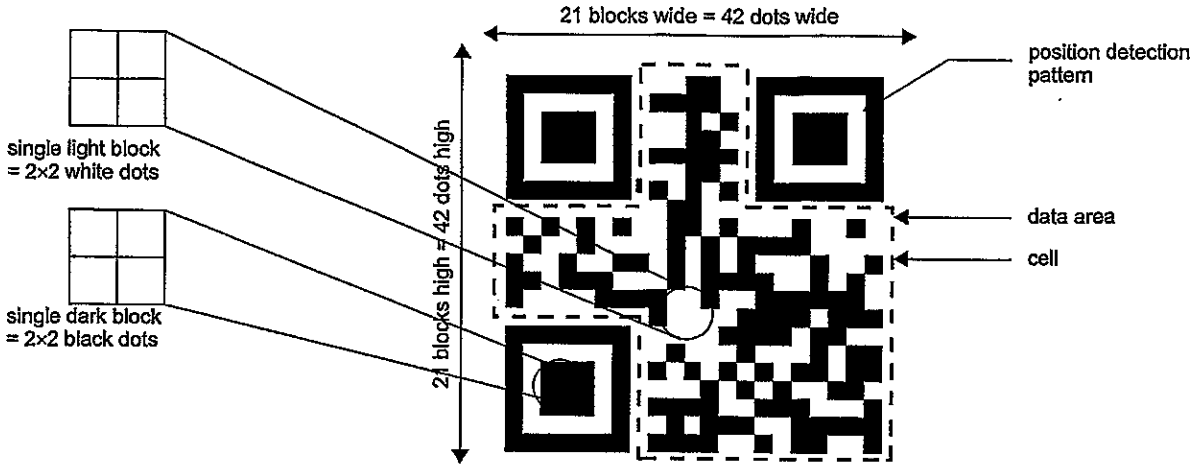


FIG. 181

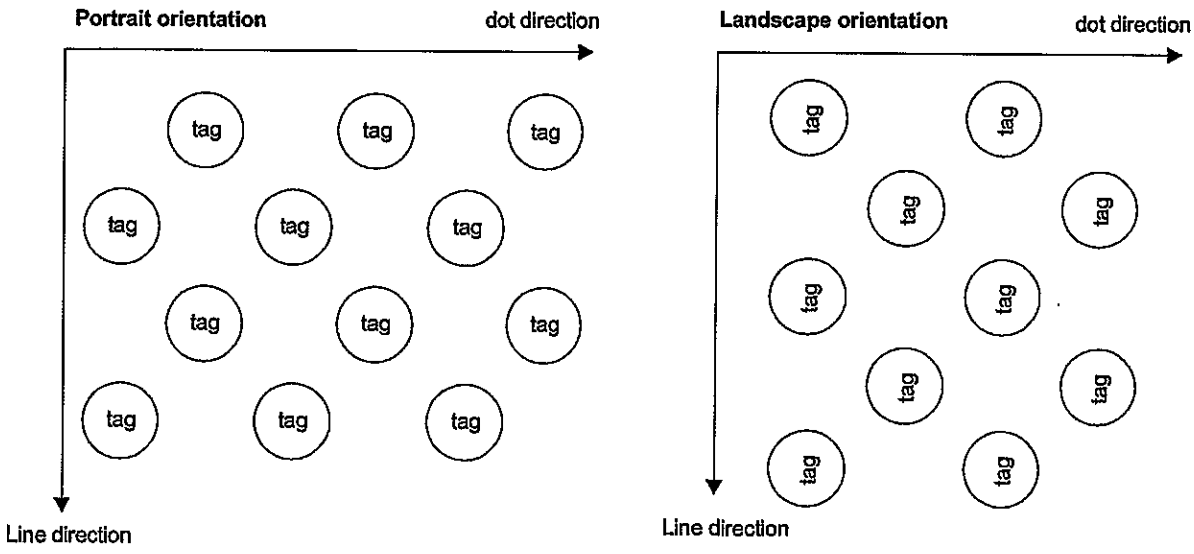


FIG. 182

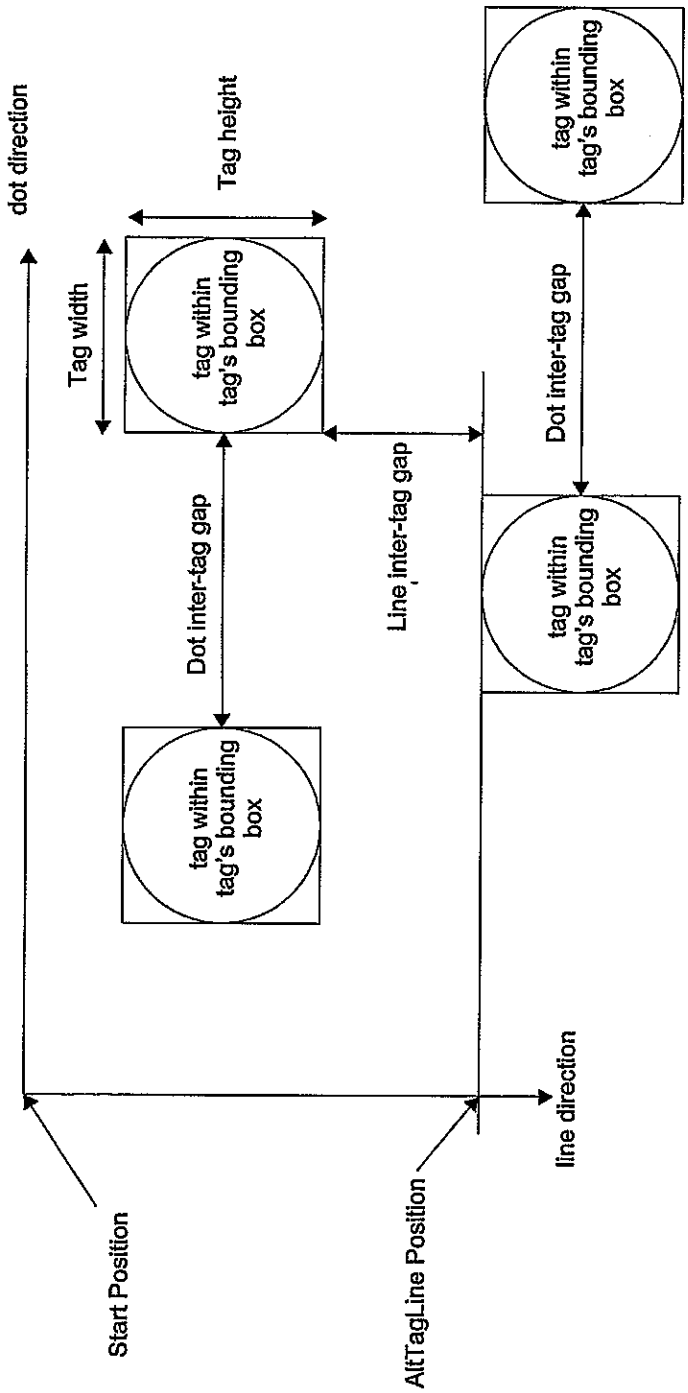

dot direction
Tag height
Tag width
tag within tag's bounding box
Dot inter-tag gap
Line inter-tag gap
Dot inter-tag gap
tag within tag's bounding box
tag within tag's bounding box
tag within tag's bounding box
Start Position
AltTagLine Position
line direction

FIG. 183

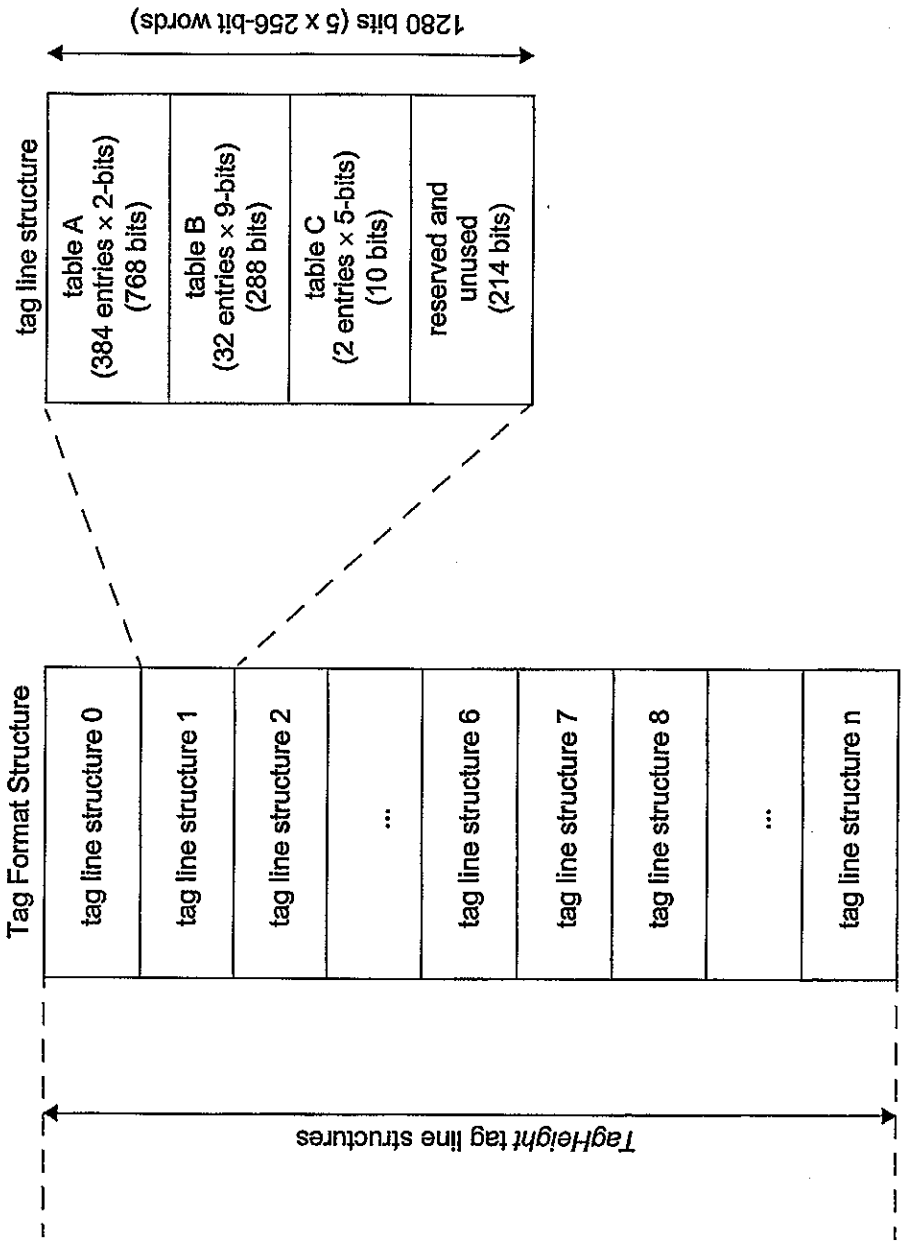
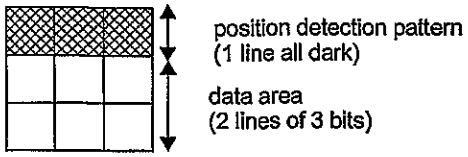

Tag Format Structure
tag line structure 0
tag line structure 1
tag line structure 2
...
tag line structure 6
tag line structure 7
tag line structure 8
...
tag line structure n
TagHeight tag line structures
tag line structure
table A
(384 entries × 2-bits)
(768 bits)
table B
(32 entries × 9-bits)
(288 bits)
table C
(2 entries × 5-bits)
(10 bits)
reserved and
unused
(214 bits)
1280 bits (5 x 256-bit words)

FIG. 184



always 1 (background)	always 1 (background)	always 1 (background)
data bit 0	data bit 1	data bit 2
data bit 4	data bit 5	data bit 3

FIG. 185

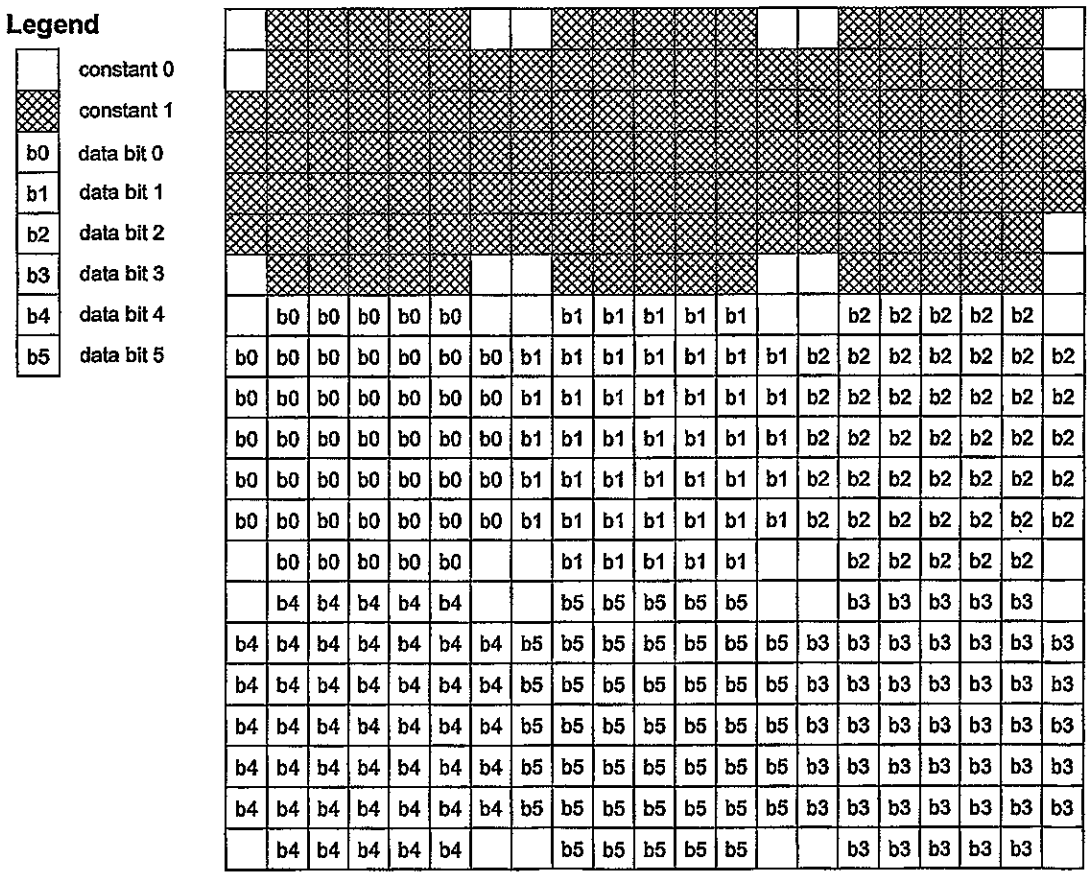


FIG. 186

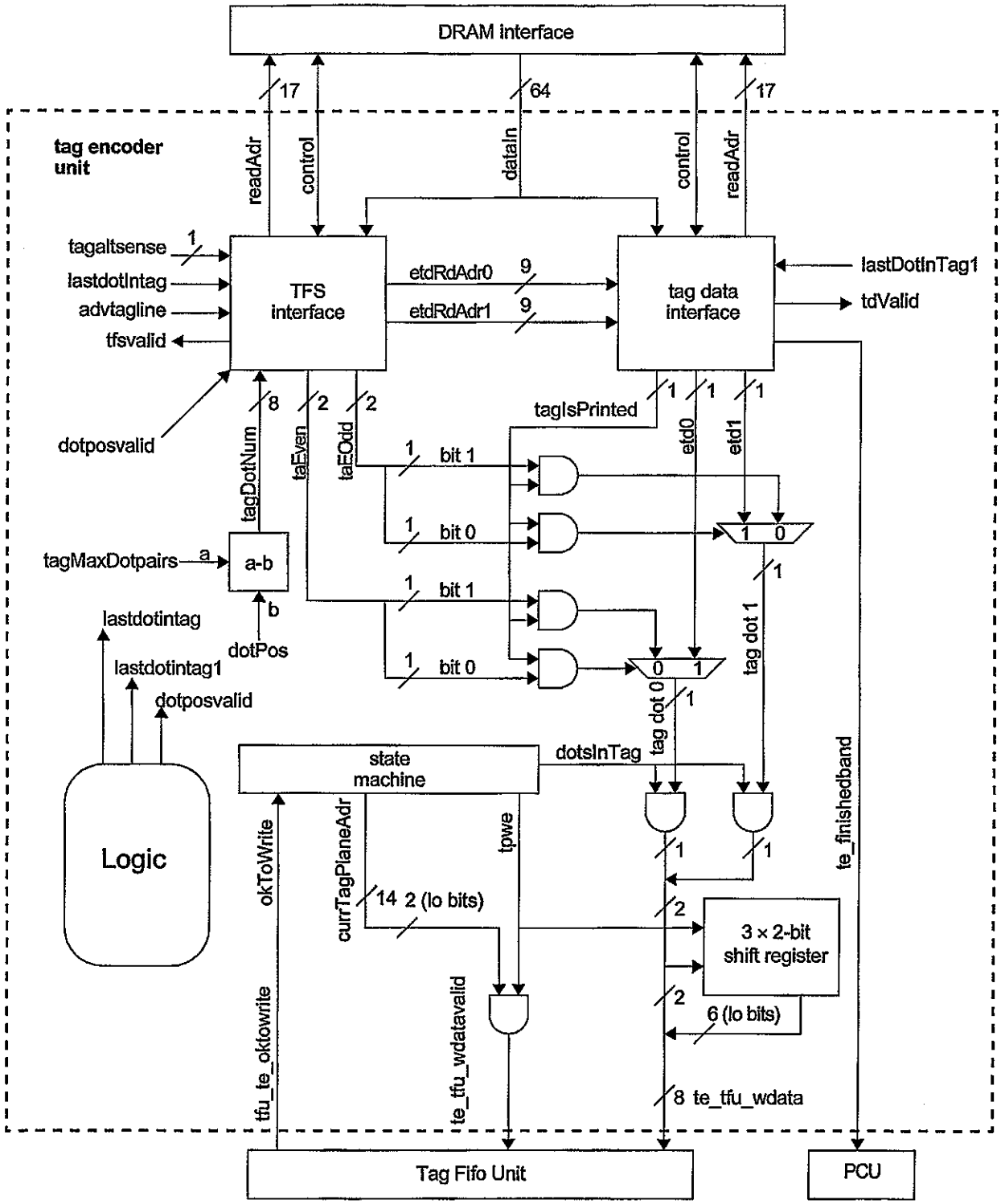

DRAM interface
17
64
17
tag encoder unit
readAdr
control
dataIn
control
readAdr
tagaltsense
1
lastdotIntag
advtagline
tfsvalid
TFS interface
etdRdAdr0
9
etdRdAdr1
9
tag data interface
lastDotInTag1
tdValid
dotposvalid
8
2
2
tagIsPrinted
1
1
1
etd0
etd1
tagDotNum
taEven
taEOdd
1
bit 1
1
bit 0
1
bit 1
1
bit 0
1 0
tagMaxDotpairs
a
a-b
b
dotPos
lastdotintag
lastdotintag1
dotposvalid
0 1
tag dot 1
1
tag dot 0
1
state machine
dotsInTag
Logic
okToWrite
tpwe
currTagPlaneAdr
14
2 (lo bits)
1
1
te_finishedband
2
3 × 2-bit shift register
2
6 (lo bits)
tfu_te_oktowrite
te_tfu_wdatavalid
8
te_tfu_wdata
Tag Fifo Unit
PCU

FIG. 187

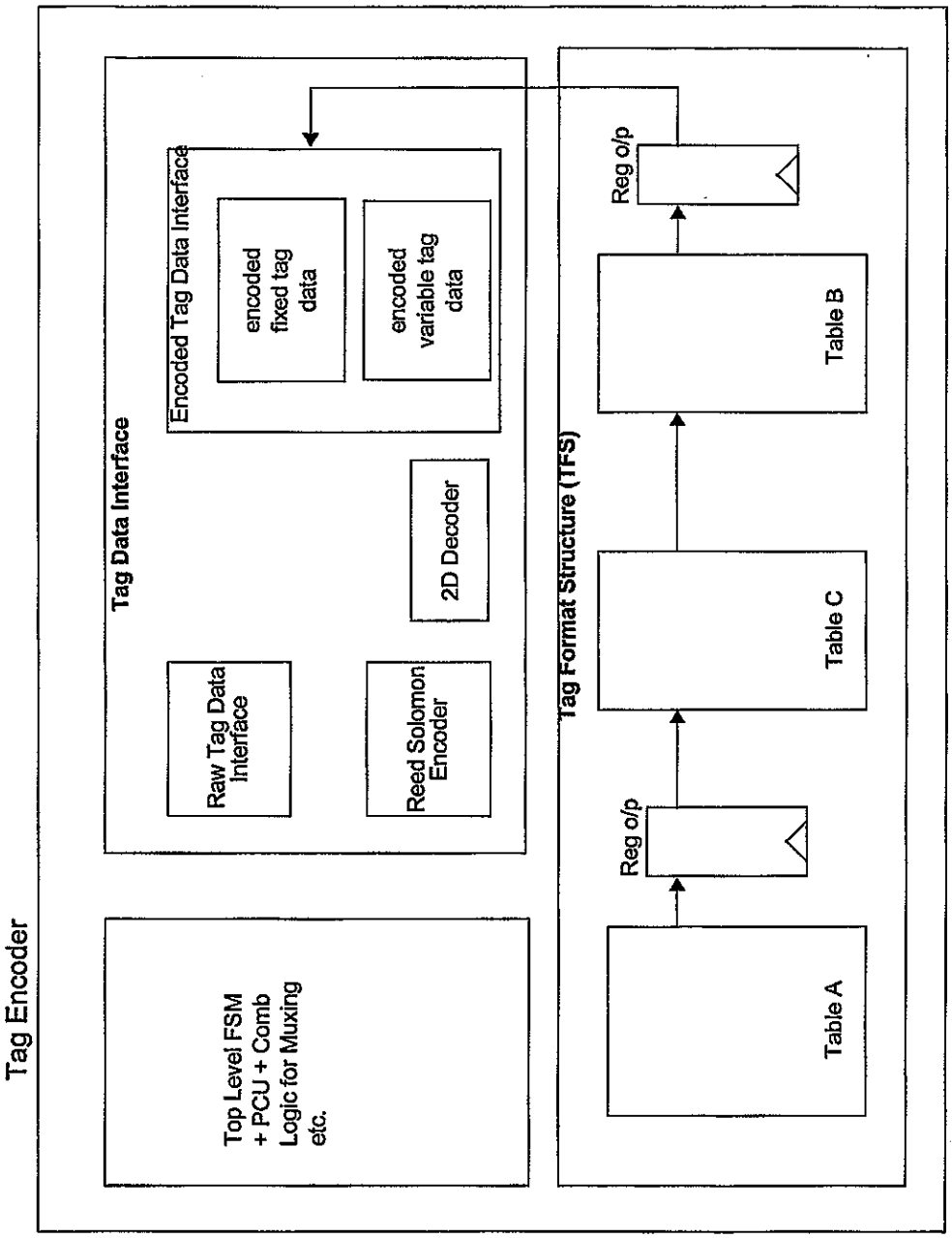

Tag Encoder
Top Level FSM
+ PCU + Comb
Logic for Muxing
etc.
Tag Data Interface
Raw Tag Data
Interface
Reed Solomon
Encoder
2D Decoder
Encoded Tag Data Interface
encoded
fixed tag
data
encoded
variable tag
data
Tag Format Structure (TFS)
Table A
Reg o/p
Table C
Table B
Reg o/p

FIG. 188

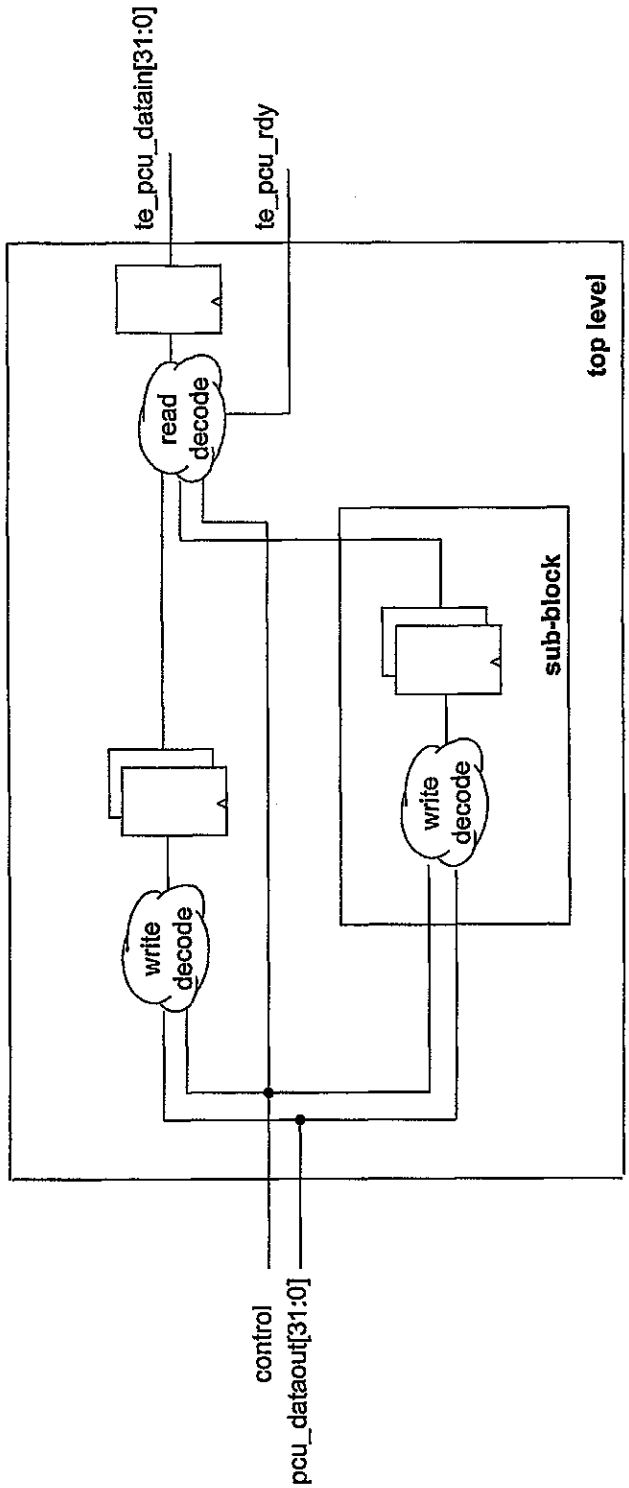

te_pcu_datain[31:0]
te_pcu_rdy
top level
read decode
sub-block
write decode
write decode
control
pcu_dataout[31:0]

FIG. 189

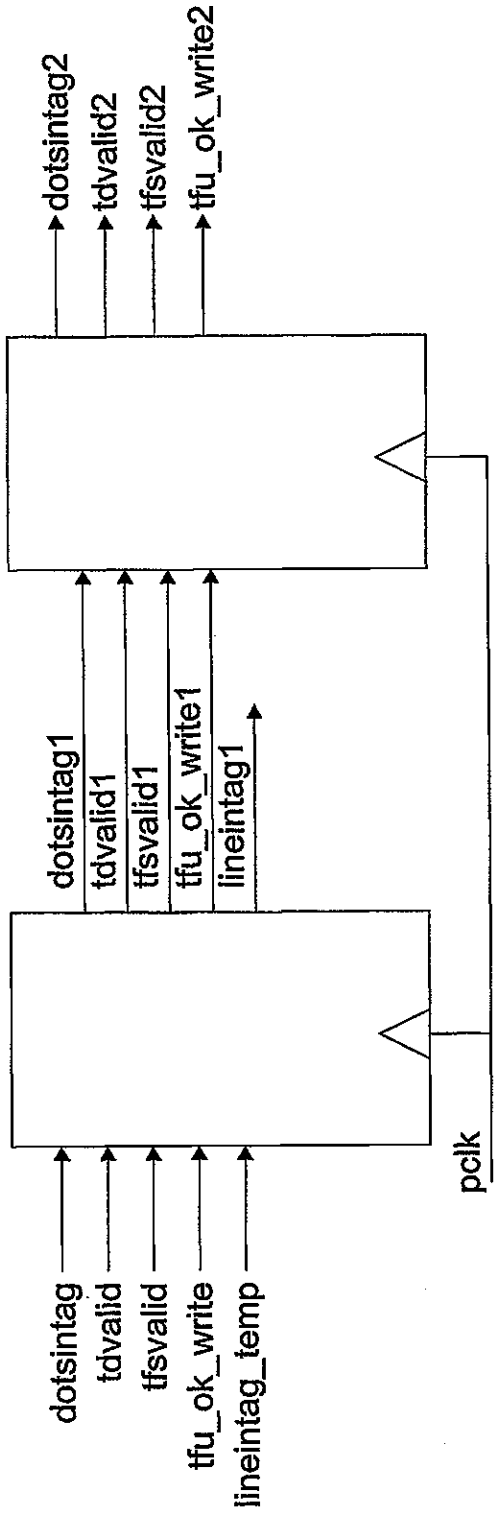

dotsintag2
tdvalid2
tfsvalid2
tfu_ok_write2
dotsintag1
tdvalid1
tfsvalid1
tfu_ok_write1
lineintag1
pclk
dotsintag
tdvalid
tfsvalid
tfu_ok_write
lineintag_temp

FIG. 191

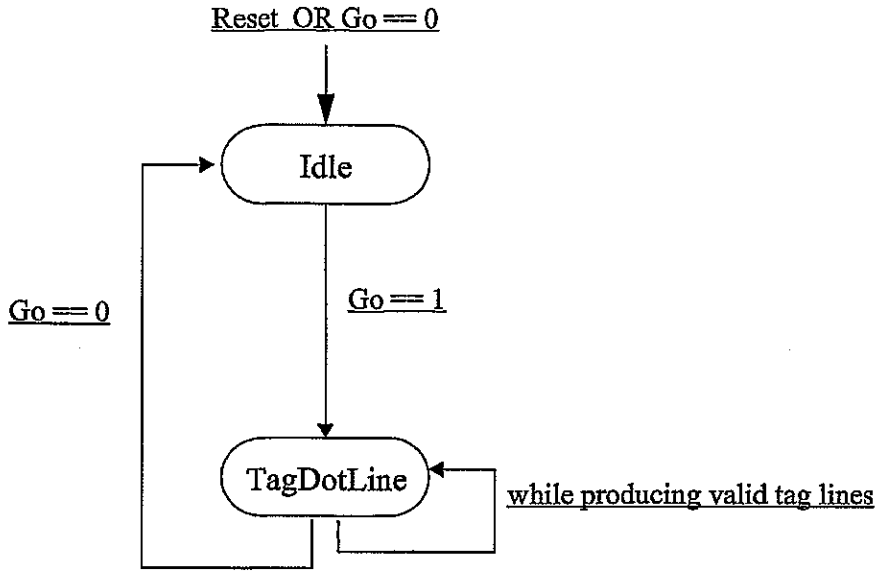


FIG. 190

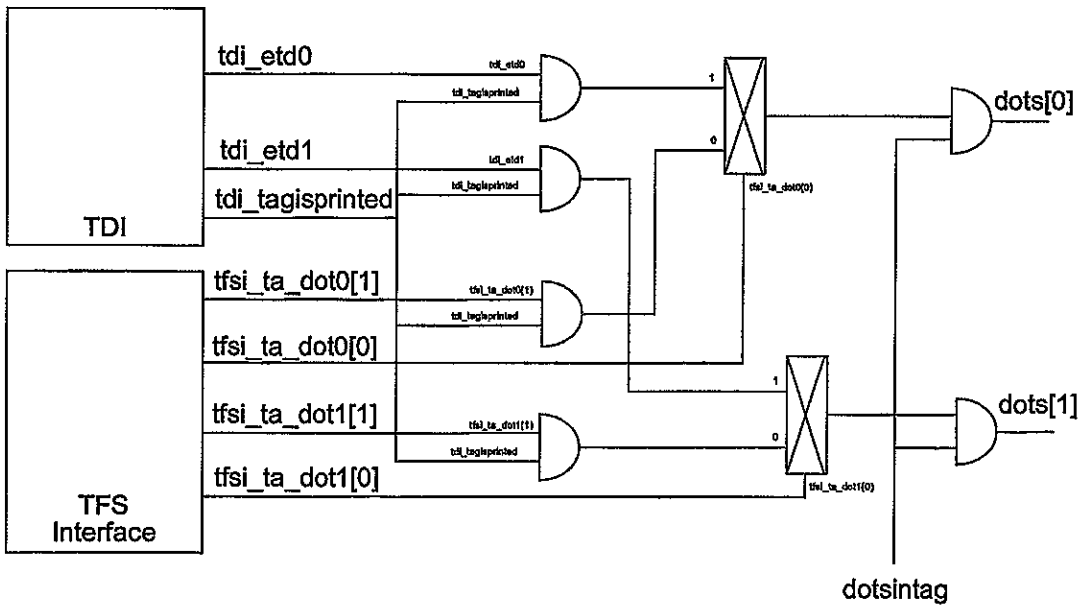


FIG. 192

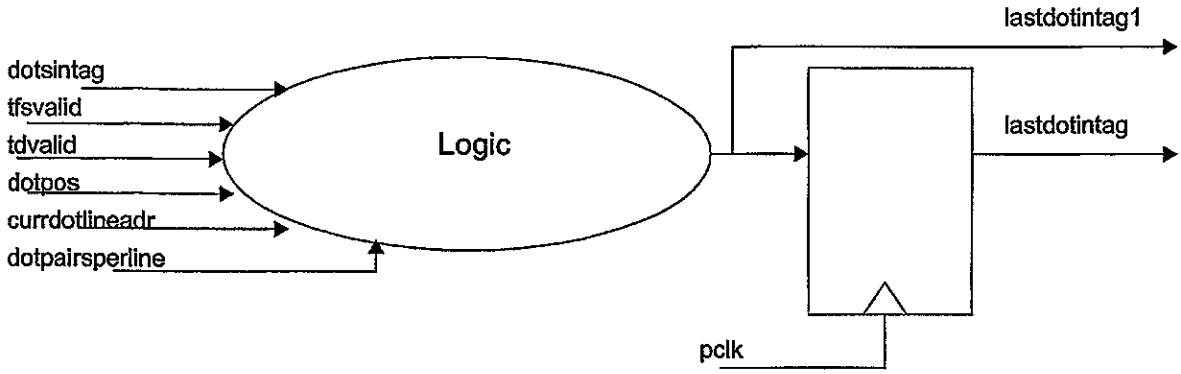


FIG. 193

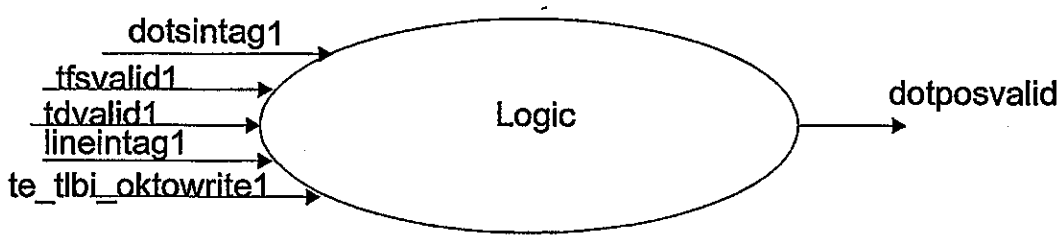


FIG. 194

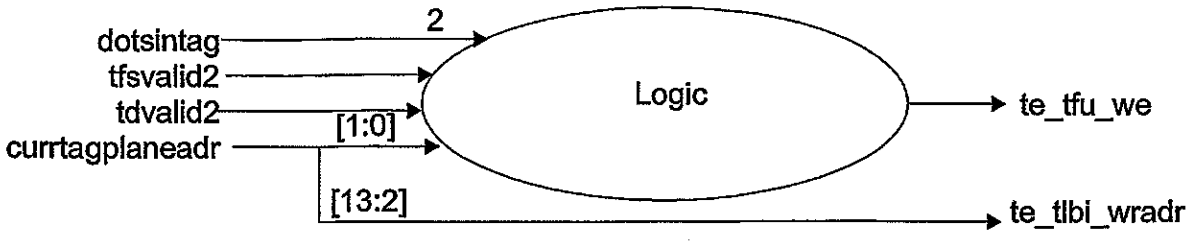


FIG. 195

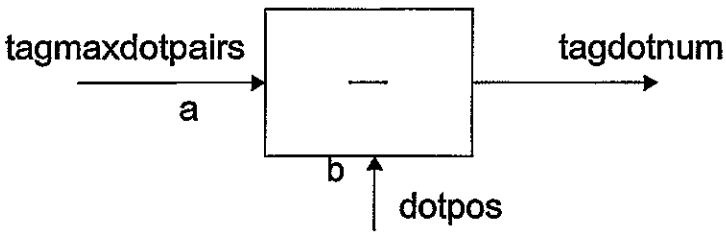


FIG. 196

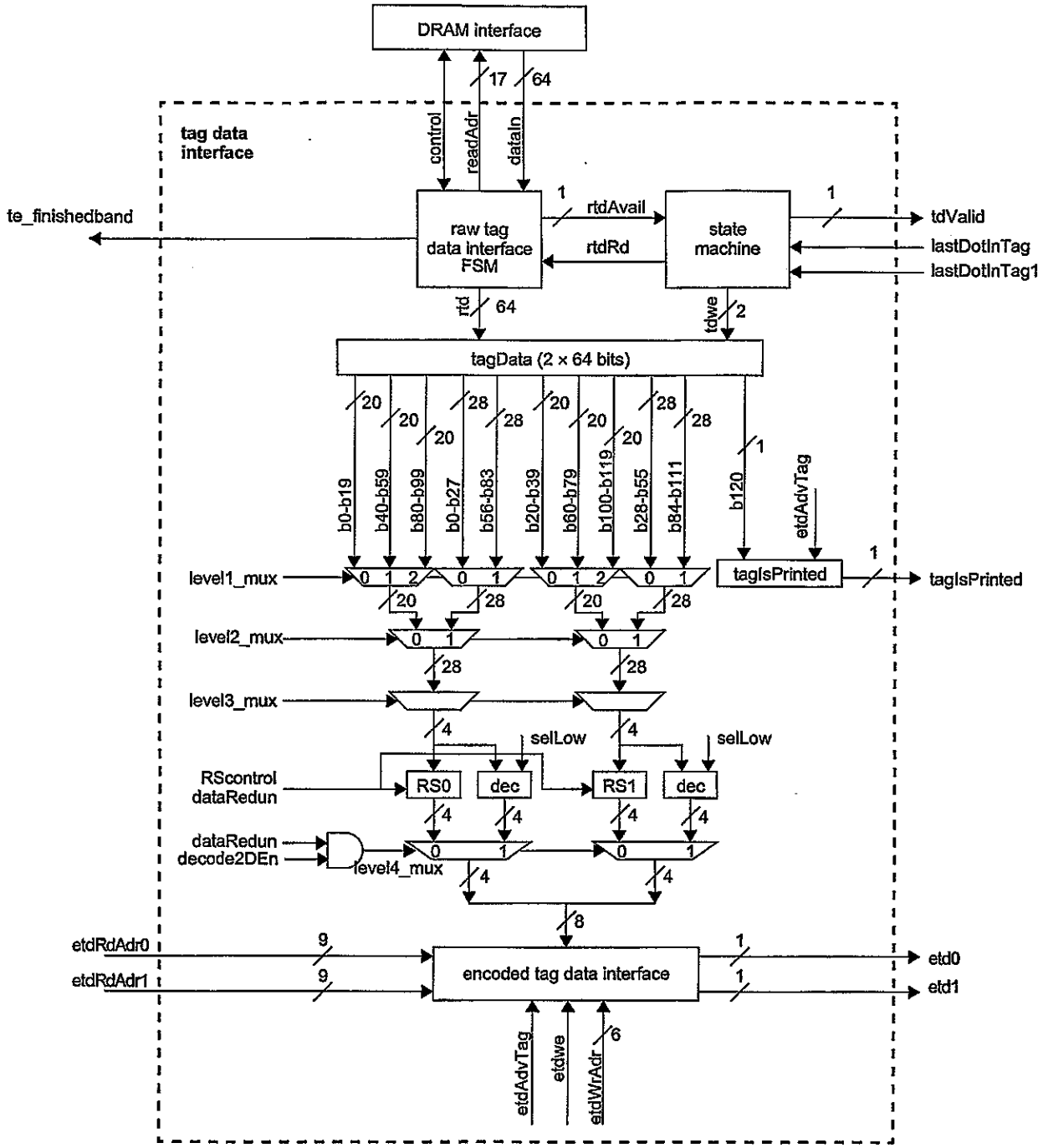

DRAM interface
17
64
tag data interface
control
readAdr
dataIn
te_finishedband
raw tag data interface FSM
rtdAvail
rtdRd
state machine
tdValid
lastDotInTag
lastDotInTag1
rtd
64
tdwe
2
tagData (2 × 64 bits)
b0-b19
b40-b59
b80-b99
b0-b27
b56-b83
b20-b39
b60-b79
b100-b119
b28-b55
b84-b111
b120
etdAdvTag
tagIsPrinted
level1_mux
level2_mux
level3_mux
selLow
RScontrol
dataRedun
RS0
RS1
dec
decode2DEn
level4_mux
etdRdAdr0
etdRdAdr1
encoded tag data interface
etd0
etd1
etdAdvTag
etdwe
etdWrAdr

FIG. 197

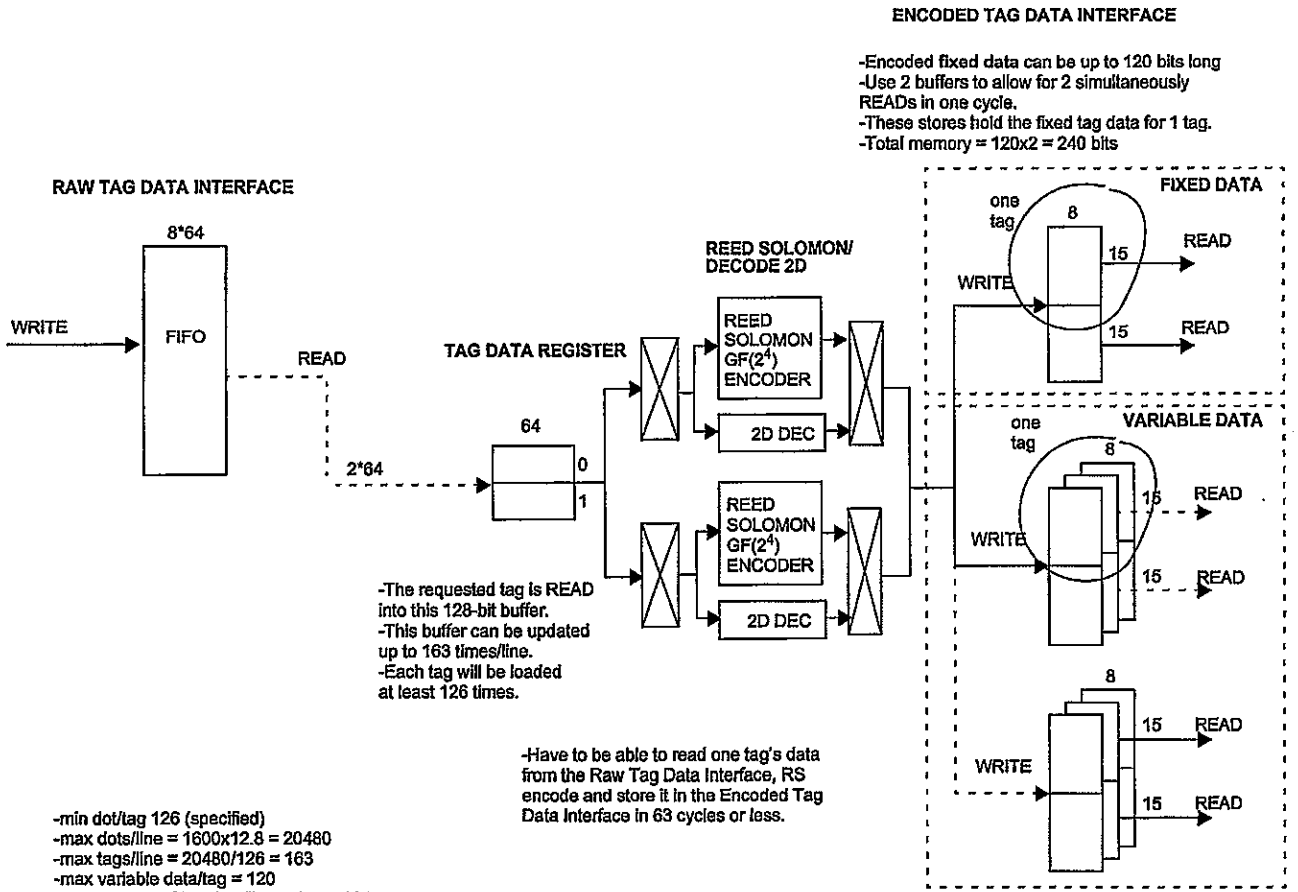

ENCODED TAG DATA INTERFACE
-Encoded fixed data can be up to 120 bits long
-Use 2 buffers to allow for 2 simultaneously READs in one cycle.
-These stores hold the fixed tag data for 1 tag.
-Total memory = 120x2 = 240 bits
RAW TAG DATA INTERFACE
8*64
WRITE
FIFO
READ
2*64
TAG DATA REGISTER
64
0
1
REED SOLOMON/ DECODE 2D
REED SOLOMON GF(2^4) ENCODER
2D DEC
REED SOLOMON GF(2^4) ENCODER
2D DEC
-The requested tag is READ into this 128-bit buffer.
-This buffer can be updated up to 163 times/line.
-Each tag will be loaded at least 126 times.
FIXED DATA
one tag
8
15
READ
WRITE
15
READ
VARIABLE DATA
one tag
8
15
READ
WRITE
15
READ
8
15
READ
WRITE
15
READ
-Have to be able to read one tag's data from the Raw Tag Data Interface, RS encode and store it in the Encoded Tag Data Interface in 63 cycles or less.
-min dot/tag 126 (specified)
-max dots/line = 1600x12.8 = 20480
-max tags/line = 20480/126 = 163
-max variable data/tag = 120
-max amount of tag data/line = 120 x 164
-Split the 120 tag data bits into 2x64-bits (8 spare bits)
-Max memory needed for 1 line of tag data = 2x64x164 = 656x32
-Divide this in half to allow for simultaneous READ/WRITE
-Once all this data is loaded it will be valid for at least 126 lines.
-From the specification, we must be able to process 2 dots/cycle.
-126 lines contains 20480x126 = 2580480 dots.
-Therefore the data will be updated at most every 1290240 cycles.
-Total memory = 164x2x64 = 20992-bits
-The store uses 9-bit addressing. Bit-9 indicates which buffer.
-Once printing has started each half buffer has 1/2 a line in which to be loaded
I.e. for a 12,8 inch line it has 10240 dots or 5120 cycles
for an 8 inch line it has 6400 dots or 3200 cycles
-Encoded variable data can be up to 360 bits long
-Use 2 buffers to allow for 2 simultaneously READs in one cycle.
-Use 2 buffers to allow for simultaneously READ/WRITE
-Total memory = 360x2x2 = 1280 bits
-Min tag width = 126 dots
so the fastest that 1 tag can be read = 126/2 = 63 cycles

FIG. 198

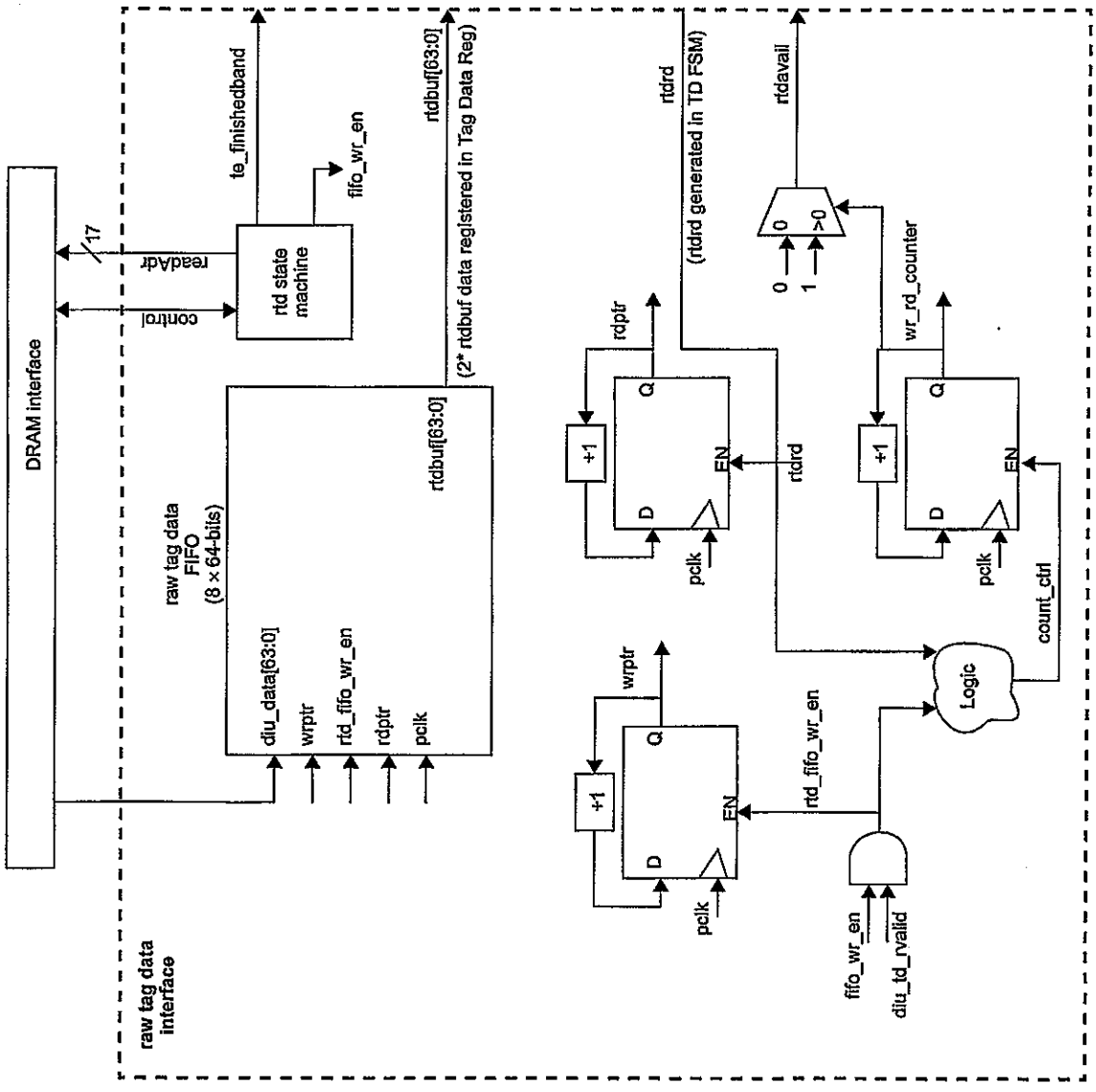

DRAM interface
17
readAdr
control
rtd state machine
te_finishedband
fifo_wr_en
rtdbuf[63:0]
(2* rtdbuf data registered in Tag Data Reg)
raw tag data FIFO (8 × 64-bits)
diu_data[63:0]
wrptr
rtd_fifo_wr_en
rdptr
pclk
rtdbuf[63:0]
raw tag data interface
rtdrd
(rtdrd generated in TD FSM)
rtdavail
0
1
>0
rdptr
wr_rd_counter
+1
D
Q
EN
pclk
rtdrd
wrptr
rtd_fifo_wr_en
Logic
count_ctrl
fifo_wr_en
diu_td_rvalid

FIG. 199

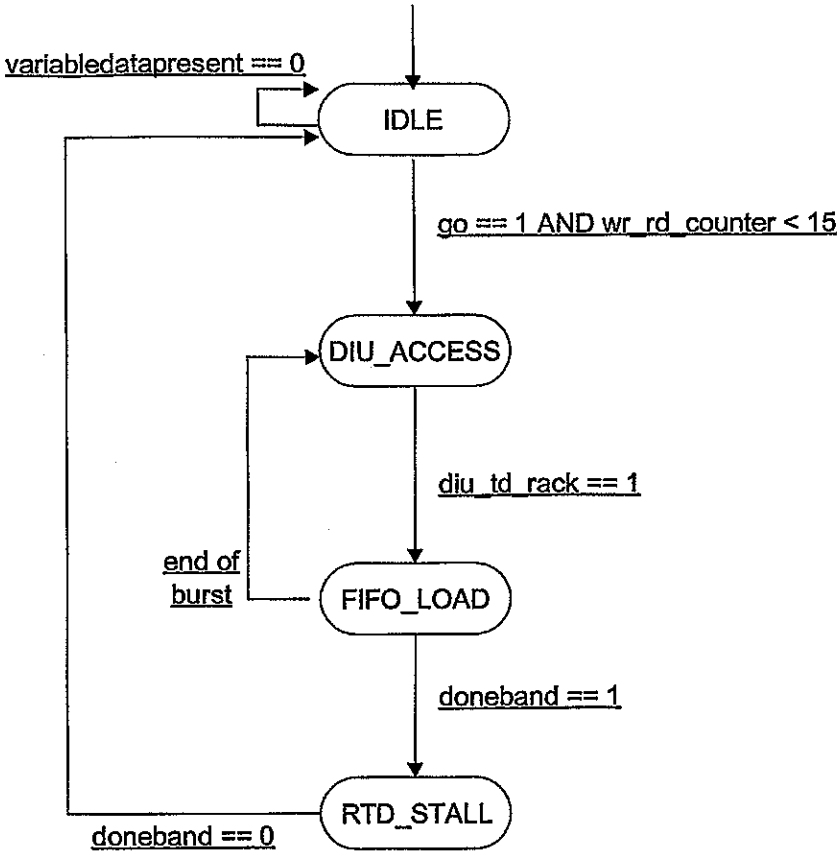


FIG. 200

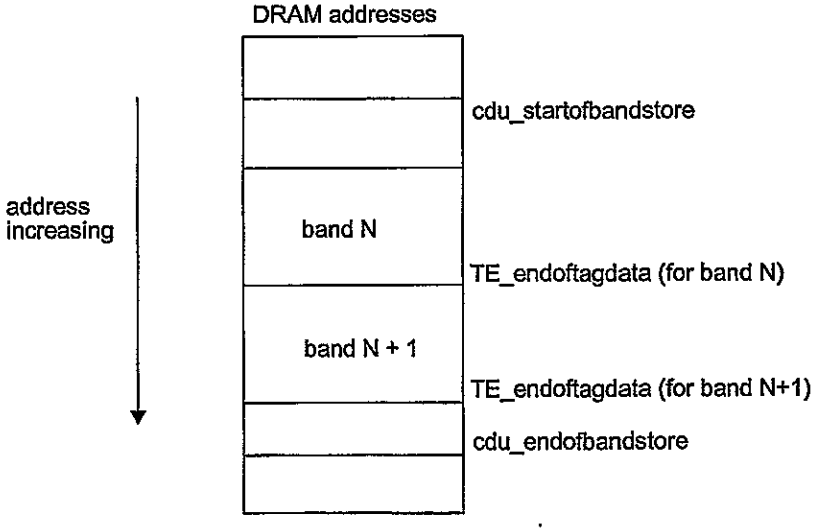


FIG. 201

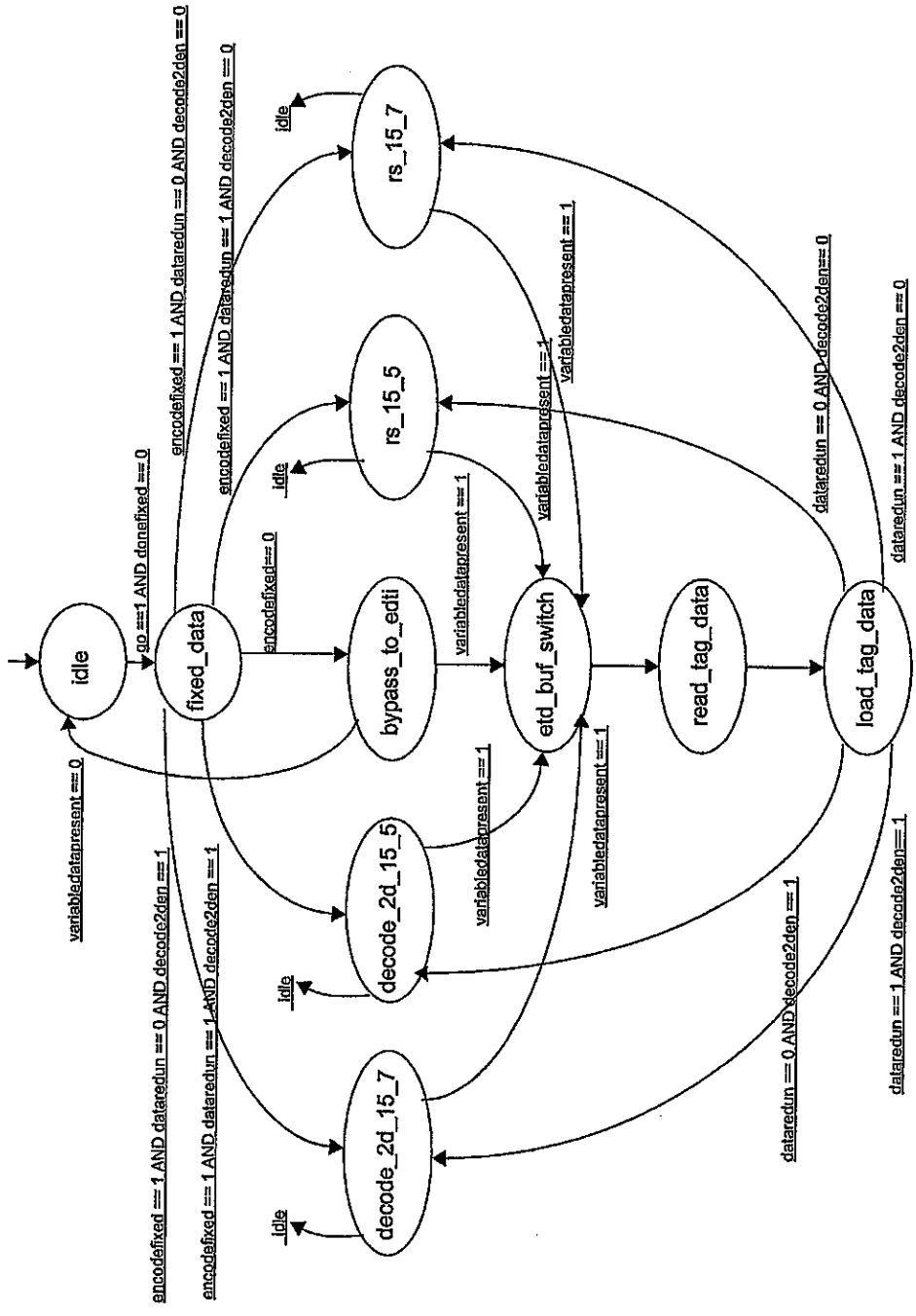

idle
fixed_data
bypass_to_edti
etd_buf_switch
read_tag_data
load_tag_data
rs_15_7
rs_15_5
decode_2d_15_5
decode_2d_15_7
go ==1 AND donefixed == 0
variabledatapresent == 0
encodefixed == 1 AND dataredun == 0 AND decode2den == 0
encodefixed == 1 AND dataredun == 1 AND decode2den == 0
encodefixed == 1 AND dataredun == 0 AND decode2den == 1
encodefixed == 1 AND dataredun == 1 AND decode2den == 1
encodefixed== 0
variabledatapresent == 1
idle
dataredun == 0 AND decode2den == 0
dataredun == 1 AND decode2den == 0
dataredun == 0 AND decode2den == 1
dataredun == 1 AND decode2den == 1

FIG. 202

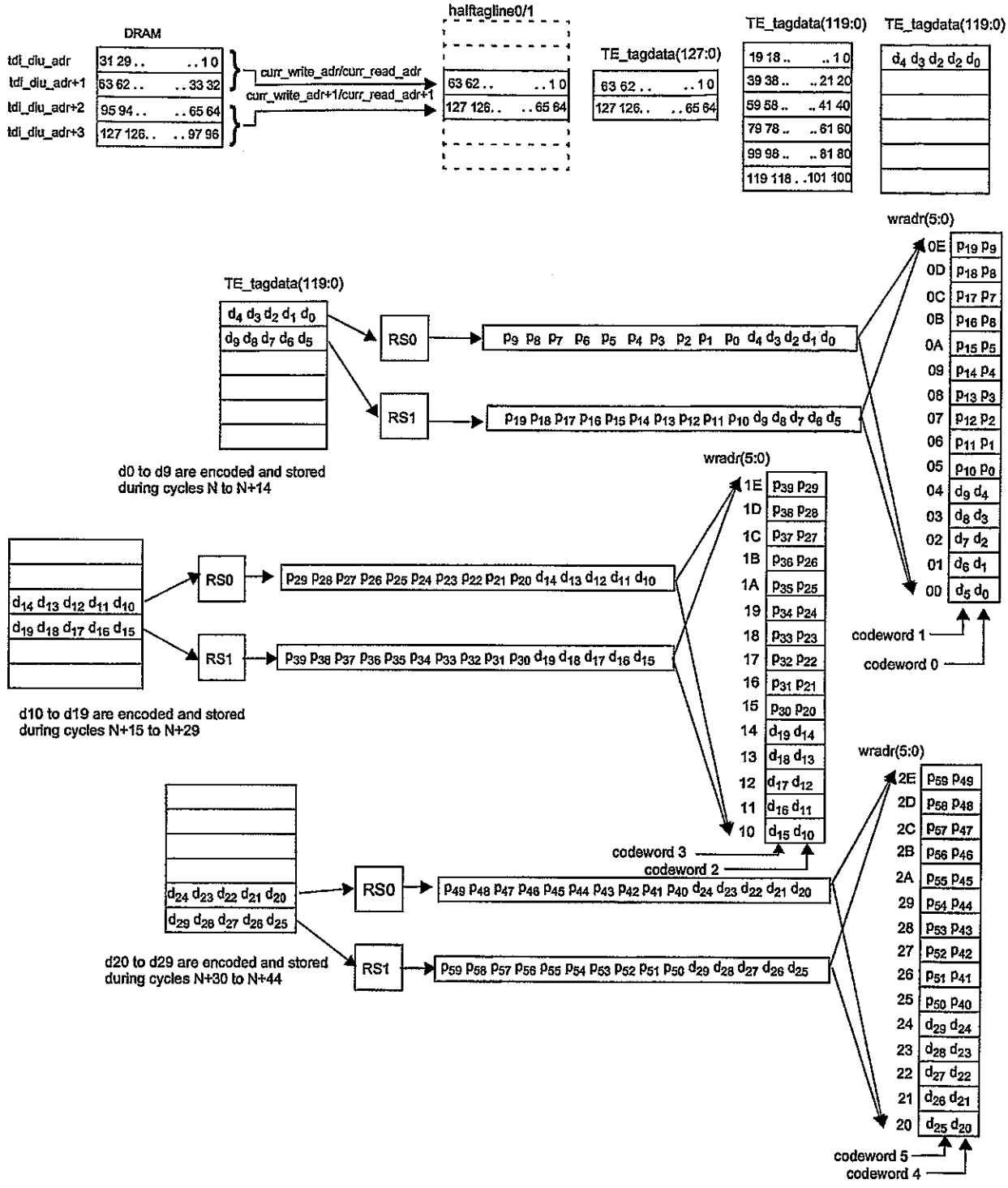

DRAM
tdi_diu_adr
tdi_diu_adr+1
tdi_diu_adr+2
tdi_diu_adr+3
curr_write_adr/curr_read_adr
curr_write_adr+1/curr_read_adr+1
halftagline0/1
TE_tagdata(127:0)
TE_tagdata(119:0)
TE_tagdata(119:0)
wradr(5:0)
RS0
RS1
d0 to d9 are encoded and stored during cycles N to N+14
d10 to d19 are encoded and stored during cycles N+15 to N+29
d20 to d29 are encoded and stored during cycles N+30 to N+44
codeword 1
codeword 0
codeword 3
codeword 2
codeword 5
codeword 4

FIG. 203

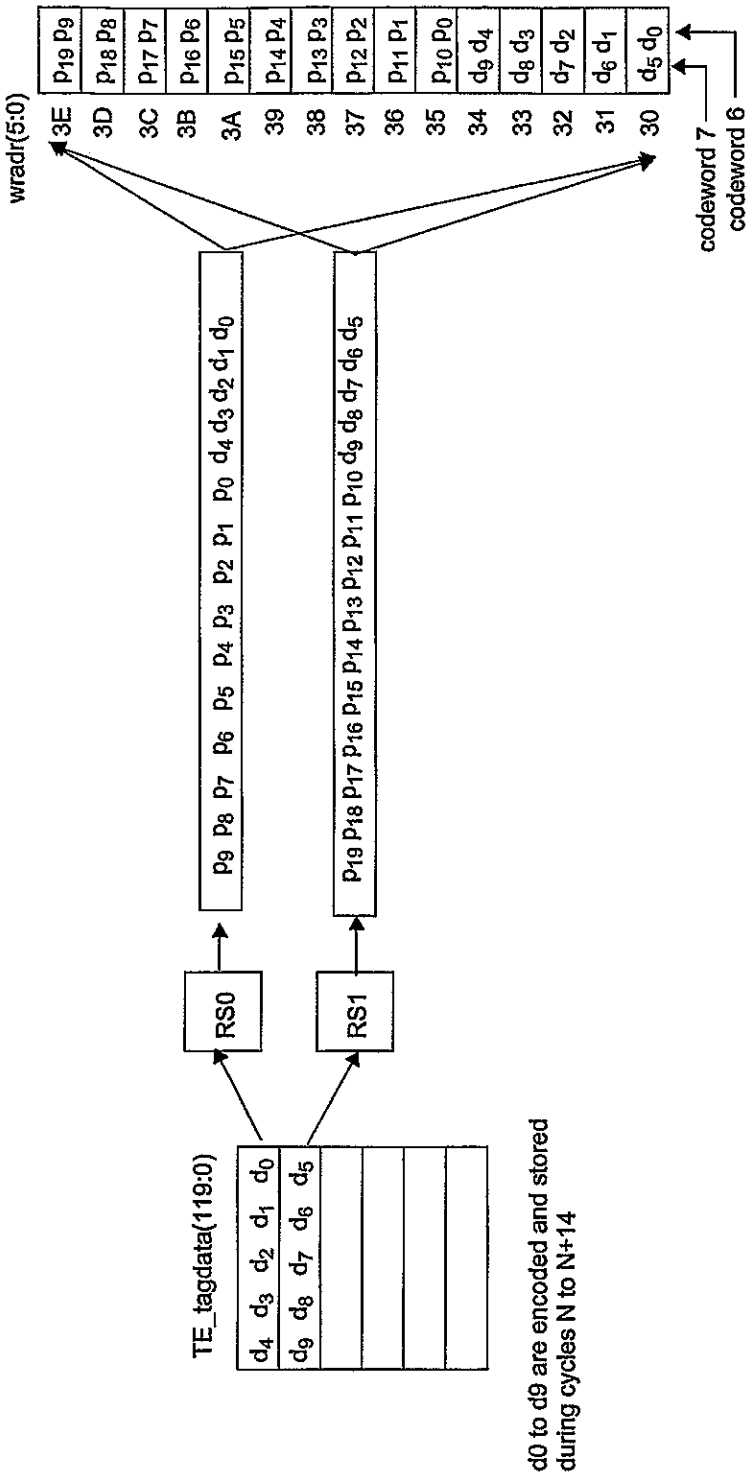

wradr(5:0)
3E
3D
3C
3B
3A
39
38
37
36
35
34
33
32
31
30
p19 p9
p18 p8
p17 p7
p16 p6
p15 p5
p14 p4
p13 p3
p12 p2
p11 p1
p10 p0
d9 d4
d8 d3
d7 d2
d6 d1
d5 d0
codeword 7
codeword 6
p9 p8 p7 p6 p5 p4 p3 p2 p1 p0 d4 d3 d2 d1 d0
p19 p18 p17 p16 p15 p14 p13 p12 p11 p10 d9 d8 d7 d6 d5
RS0
RS1
TE_tagdata(119:0)
d4 d3 d2 d1 d0
d9 d8 d7 d6 d5
d0 to d9 are encoded and stored during cycles N to N+14

FIG. 204

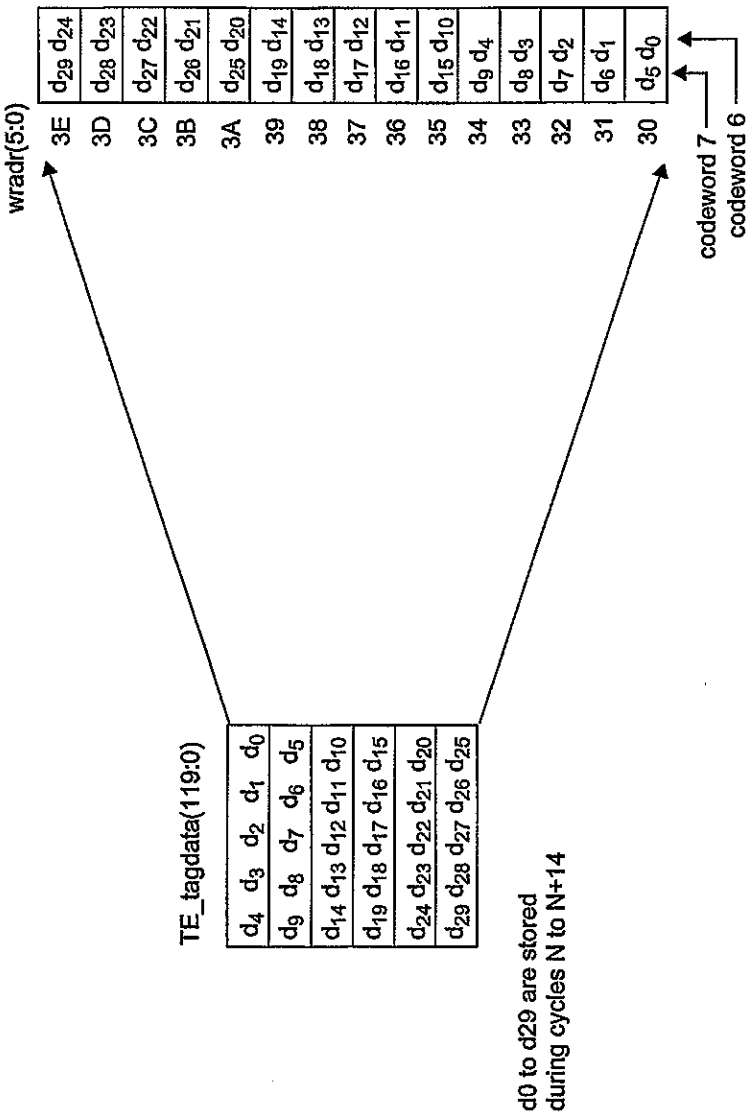

wradr(5:0)
3E
3D
3C
3B
3A
39
38
37
36
35
34
33
32
31
30
d_{29} d_{24}
d_{28} d_{23}
d_{27} d_{22}
d_{26} d_{21}
d_{25} d_{20}
d_{19} d_{14}
d_{18} d_{13}
d_{17} d_{12}
d_{16} d_{11}
d_{15} d_{10}
d_9 d_4
d_8 d_3
d_7 d_2
d_6 d_1
d_5 d_0
codeword 7
codeword 6
TE_tagdata(119:0)
d_4 d_3 d_2 d_1 d_0
d_9 d_8 d_7 d_6 d_5
d_{14} d_{13} d_{12} d_{11} d_{10}
d_{19} d_{18} d_{17} d_{16} d_{15}
d_{24} d_{23} d_{22} d_{21} d_{20}
d_{29} d_{28} d_{27} d_{26} d_{25}
d0 to d29 are stored
during cycles N to N+14

FIG. 205

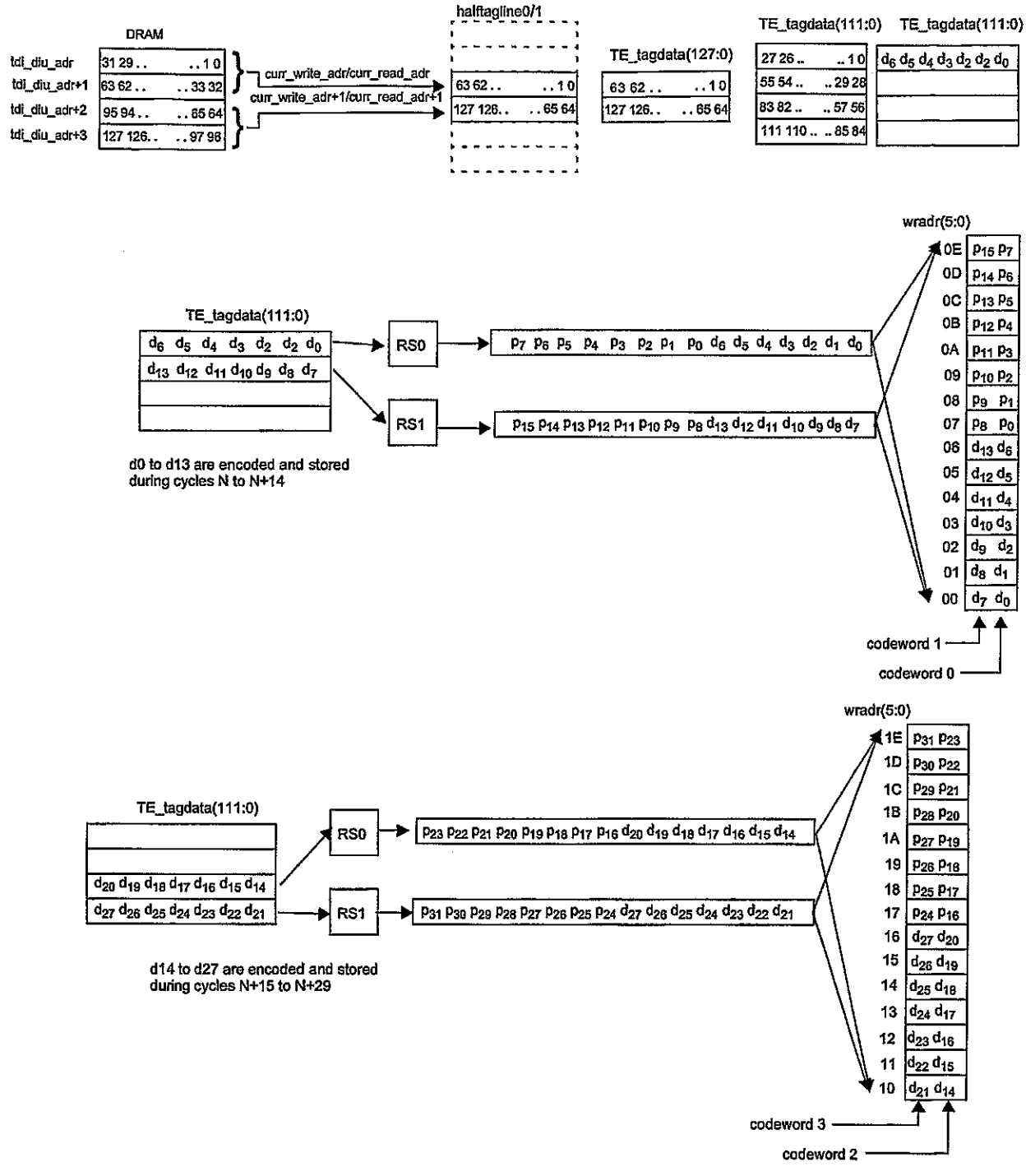

DRAM
tdi_diu_adr
tdi_diu_adr+1
tdi_diu_adr+2
tdi_diu_adr+3
31 29 .. .. 1 0
63 62 .. .. 33 32
95 94 .. .. 65 64
127 126.. .. 97 96
curr_write_adr/curr_read_adr
curr_write_adr+1/curr_read_adr+1
halftagline0/1
63 62 .. .. 1 0
127 126.. .. 65 64
TE_tagdata(127:0)
63 62 .. .. 1 0
127 126.. .. 65 64
TE_tagdata(111:0)
27 26 .. .. 1 0
55 54 .. .. 29 28
83 82 .. .. 57 56
111 110 .. .. 85 84
TE_tagdata(111:0)
d6 d5 d4 d3 d2 d2 d0
wradr(5:0)
TE_tagdata(111:0)
d6 d5 d4 d3 d2 d2 d0
d13 d12 d11 d10 d9 d8 d7
RS0
RS1
p7 p6 p5 p4 p3 p2 p1 p0 d6 d5 d4 d3 d2 d1 d0
p15 p14 p13 p12 p11 p10 p9 p8 d13 d12 d11 d10 d9 d8 d7
d0 to d13 are encoded and stored during cycles N to N+14
0E p15 p7
0D p14 p6
0C p13 p5
0B p12 p4
0A p11 p3
09 p10 p2
08 p9 p1
07 p8 p0
06 d13 d6
05 d12 d5
04 d11 d4
03 d10 d3
02 d9 d2
01 d8 d1
00 d7 d0
codeword 1
codeword 0
wradr(5:0)
TE_tagdata(111:0)
d20 d19 d18 d17 d16 d15 d14
d27 d26 d25 d24 d23 d22 d21
RS0
RS1
p23 p22 p21 p20 p19 p18 p17 p16 d20 d19 d18 d17 d16 d15 d14
p31 p30 p29 p28 p27 p26 p25 p24 d27 d26 d25 d24 d23 d22 d21
d14 to d27 are encoded and stored during cycles N+15 to N+29
1E p31 p23
1D p30 p22
1C p29 p21
1B p28 p20
1A p27 p19
19 p26 p18
18 p25 p17
17 p24 p16
16 d27 d20
15 d26 d19
14 d25 d18
13 d24 d17
12 d23 d16
11 d22 d15
10 d21 d14
codeword 3
codeword 2

FIG. 206

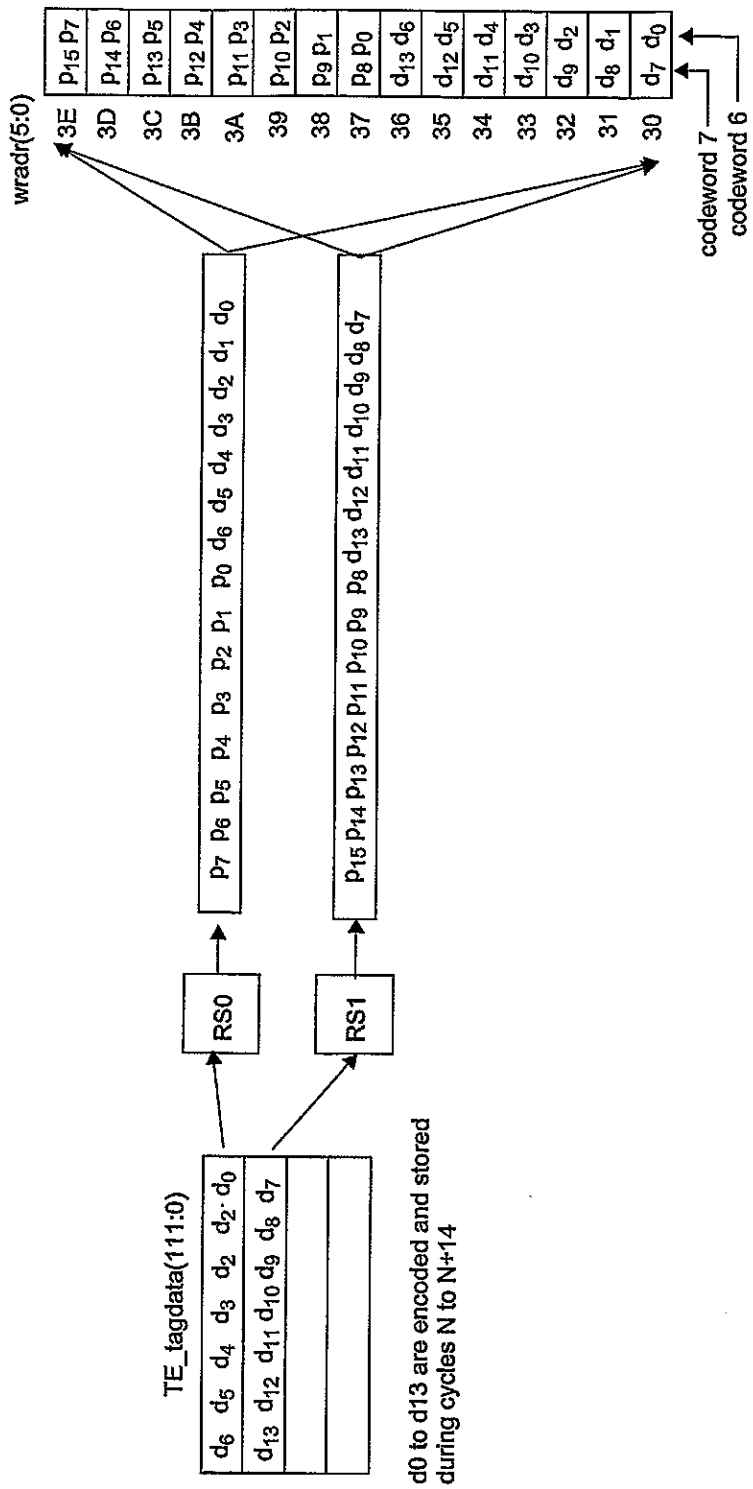

wradr(5:0)
3E
3D
3C
3B
3A
39
38
37
36
35
34
33
32
31
30
codeword 7
codeword 6
RS0
RS1
TE_tagdata(111:0)
d0 to d13 are encoded and stored during cycles N to N+14

FIG. 207

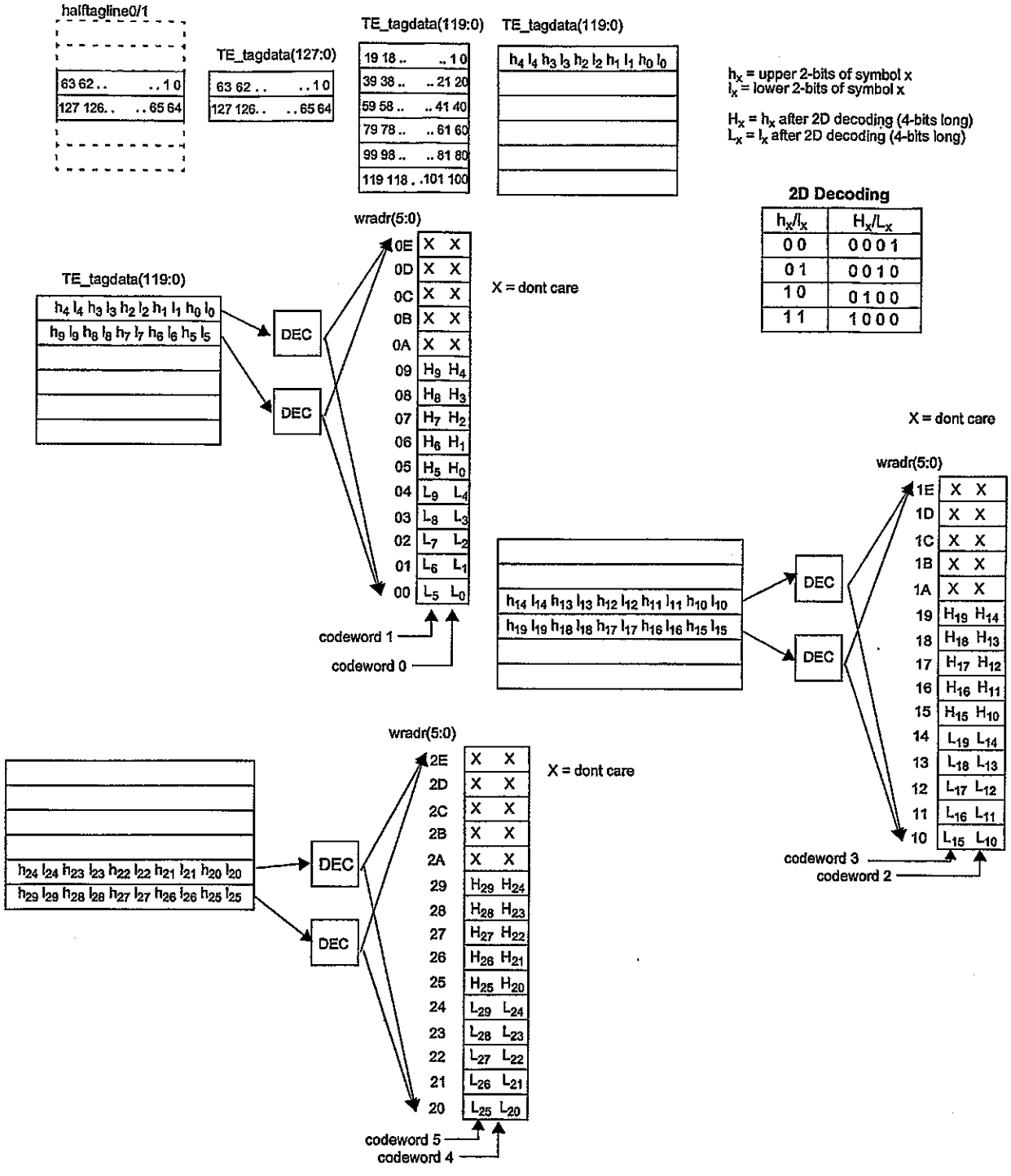

halftagline0/1
63 62 .. .. 1 0
127 126. . .. 65 64
TE_tagdata(127:0)
63 62 . . .. 1 0
127 126. . .. 65 64
TE_tagdata(119:0)
19 18 .. .. 1 0
39 38 .. .. 21 20
59 58 .. .. 41 40
79 78 .. .. 61 60
99 98 .. .. 81 80
119 118 . . 101 100
TE_tagdata(119:0)
h4 l4 h3 l3 h2 l2 h1 l1 h0 l0
hx = upper 2-bits of symbol x
lx = lower 2-bits of symbol x
Hx = hx after 2D decoding (4-bits long)
Lx = lx after 2D decoding (4-bits long)
2D Decoding
hx/lx
Hx/Lx
0 0
0 0 0 1
0 1
0 0 1 0
1 0
0 1 0 0
1 1
1 0 0 0
wradr(5:0)
0E X X
0D X X
0C X X
0B X X
0A X X
09 H9 H4
08 H8 H3
07 H7 H2
06 H6 H1
05 H5 H0
04 L9 L4
03 L8 L3
02 L7 L2
01 L6 L1
00 L5 L0
X = dont care
TE_tagdata(119:0)
h4 l4 h3 l3 h2 l2 h1 l1 h0 l0
h9 l9 h8 l8 h7 l7 h6 l6 h5 l5
DEC
DEC
codeword 1
codeword 0
X = dont care
wradr(5:0)
1E X X
1D X X
1C X X
1B X X
1A X X
19 H19 H14
18 H18 H13
17 H17 H12
16 H16 H11
15 H15 H10
14 L19 L14
13 L18 L13
12 L17 L12
11 L16 L11
10 L15 L10
h14 l14 h13 l13 h12 l12 h11 l11 h10 l10
h19 l19 h18 l18 h17 l17 h16 l16 h15 l15
DEC
DEC
codeword 3
codeword 2
wradr(5:0)
2E X X
2D X X
2C X X
2B X X
2A X X
29 H29 H24
28 H28 H23
27 H27 H22
26 H26 H21
25 H25 H20
24 L29 L24
23 L28 L23
22 L27 L22
21 L26 L21
20 L25 L20
X = dont care
h24 l24 h23 l23 h22 l22 h21 l21 h20 l20
h29 l29 h28 l28 h27 l27 h26 l26 h25 l25
DEC
DEC
codeword 5
codeword 4

FIG. 208

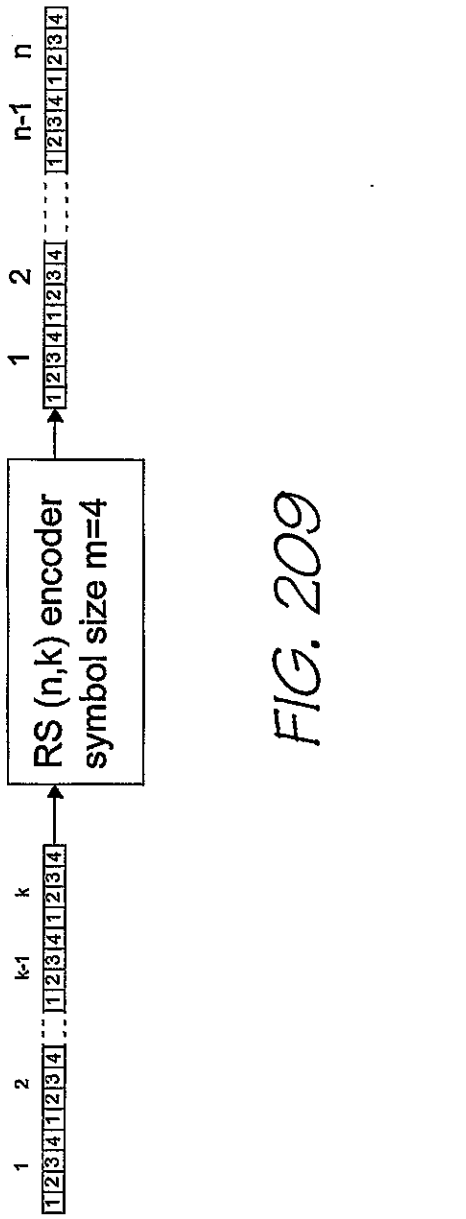


FIG. 209

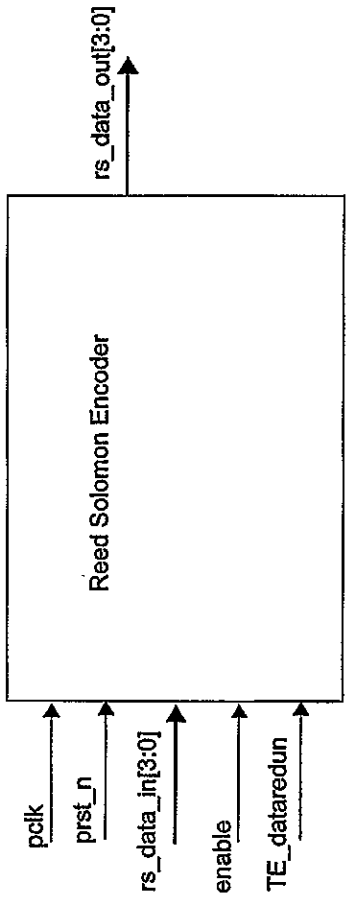


FIG. 210

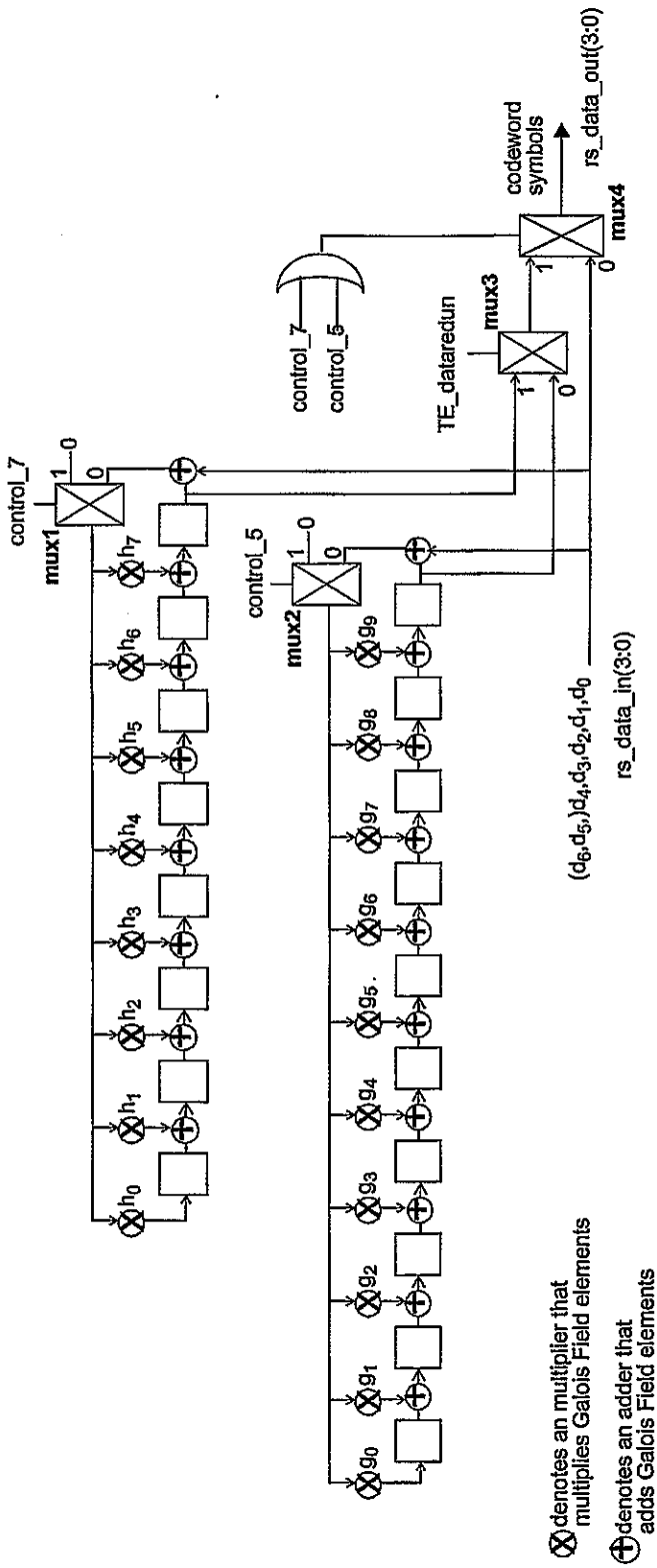

control_7
mux1
control_5
mux2
control_7
control_5
TE_dataredun
mux3
codeword symbols
rs_data_out(3:0)
mux4
$(d_6,d_5,)d_4,d_3,d_2,d_1,d_0$
rs_data_in(3:0)
h_0 h_1 h_2 h_3 h_4 h_5 h_6 h_7
g_0 g_1 g_2 g_3 g_4 g_5 g_6 g_7 g_8 g_9
denotes an multiplier that multiplies Galois Field elements
denotes an adder that adds Galois Field elements

FIG. 211

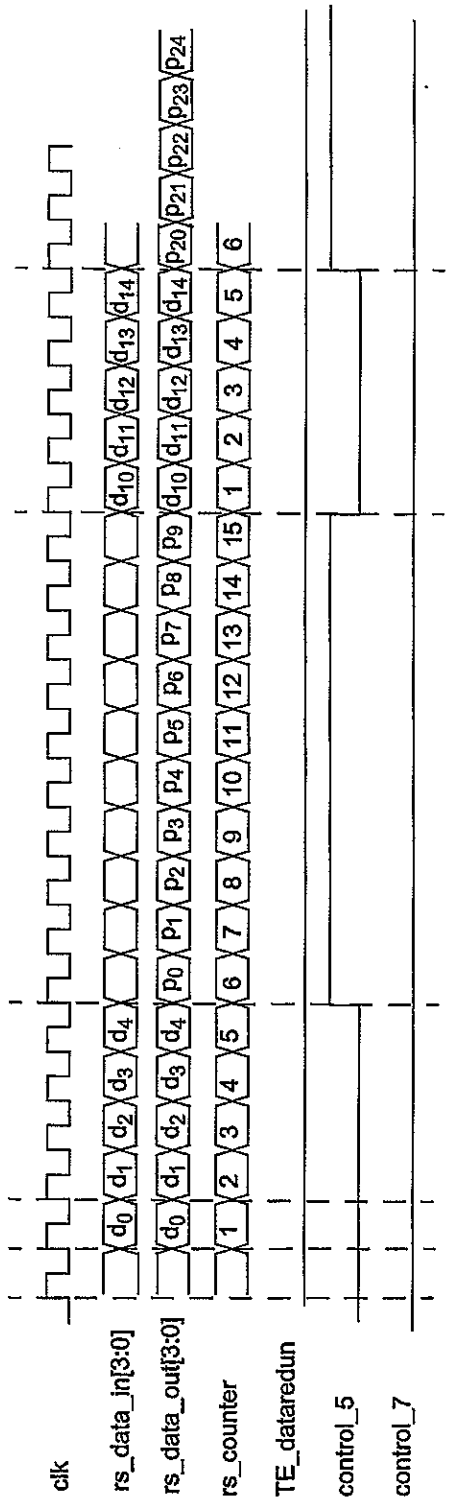

clk
rs_data_in[3:0]
rs_data_out[3:0]
rs_counter
TE_dataredun
control_5
control_7

FIG. 212

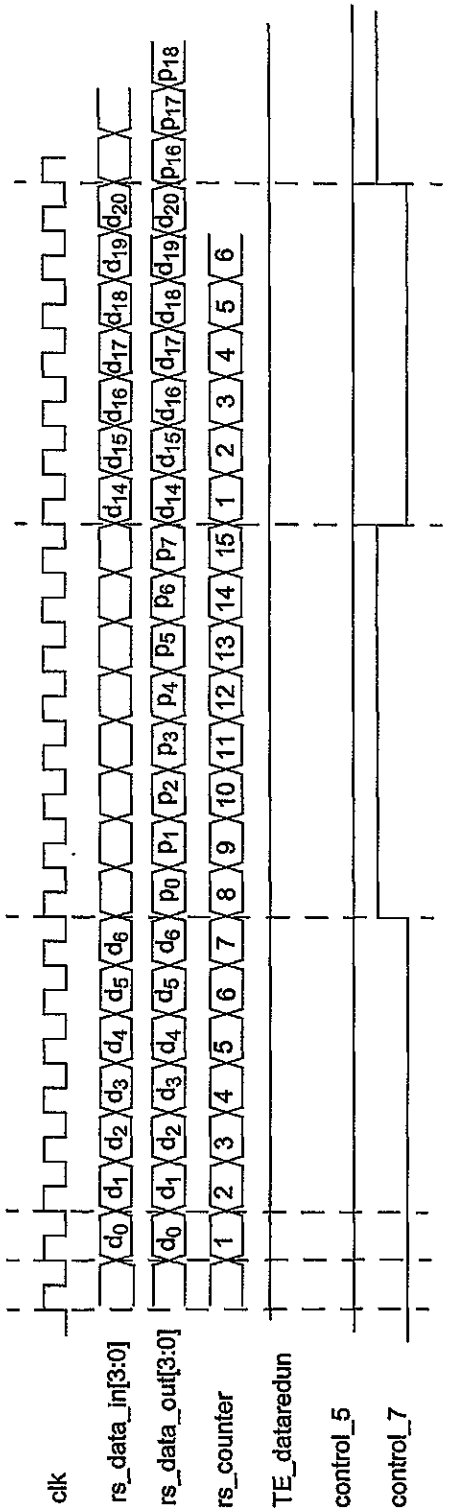

clk
rs_data_in[3:0]
rs_data_out[3:0]
rs_counter
TE_dataredun
control_5
control_7

FIG. 213

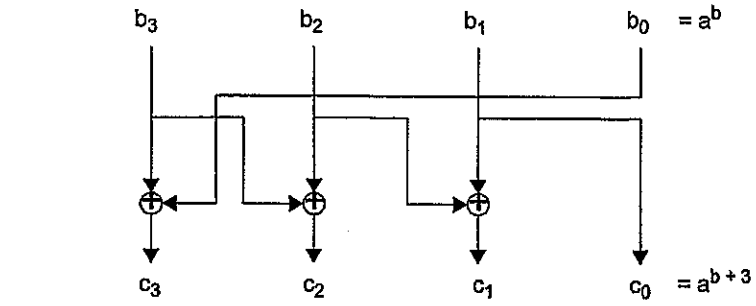


FIG. 214

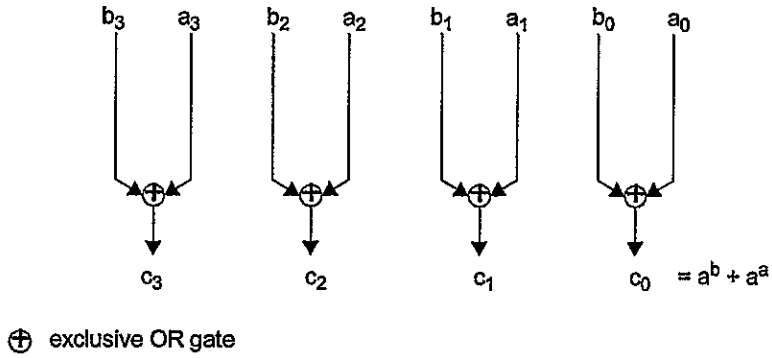


FIG. 215

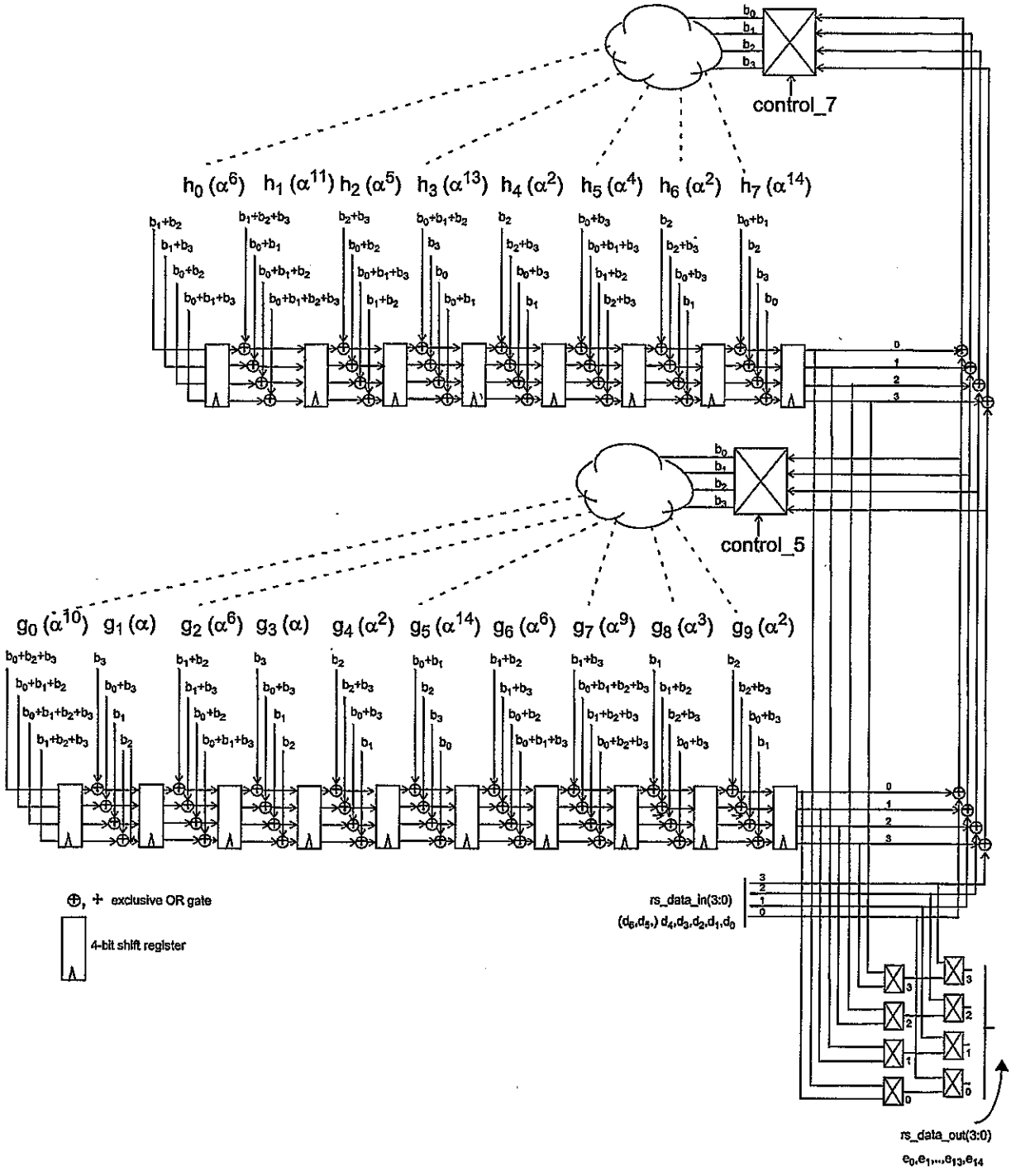

control_7
h0 (α6) h1 (α11) h2 (α5) h3 (α13) h4 (α2) h5 (α4) h6 (α2) h7 (α14)
control_5
g0 (α10) g1 (α) g2 (α6) g3 (α) g4 (α2) g5 (α14) g6 (α6) g7 (α9) g8 (α3) g9 (α2)
⊕, + exclusive OR gate
4-bit shift register
rs_data_in(3:0)
(d6,d5,) d4,d3,d2,d1,d0
rs_data_out(3:0)
e0,e1,...,e13,e14

FIG. 216

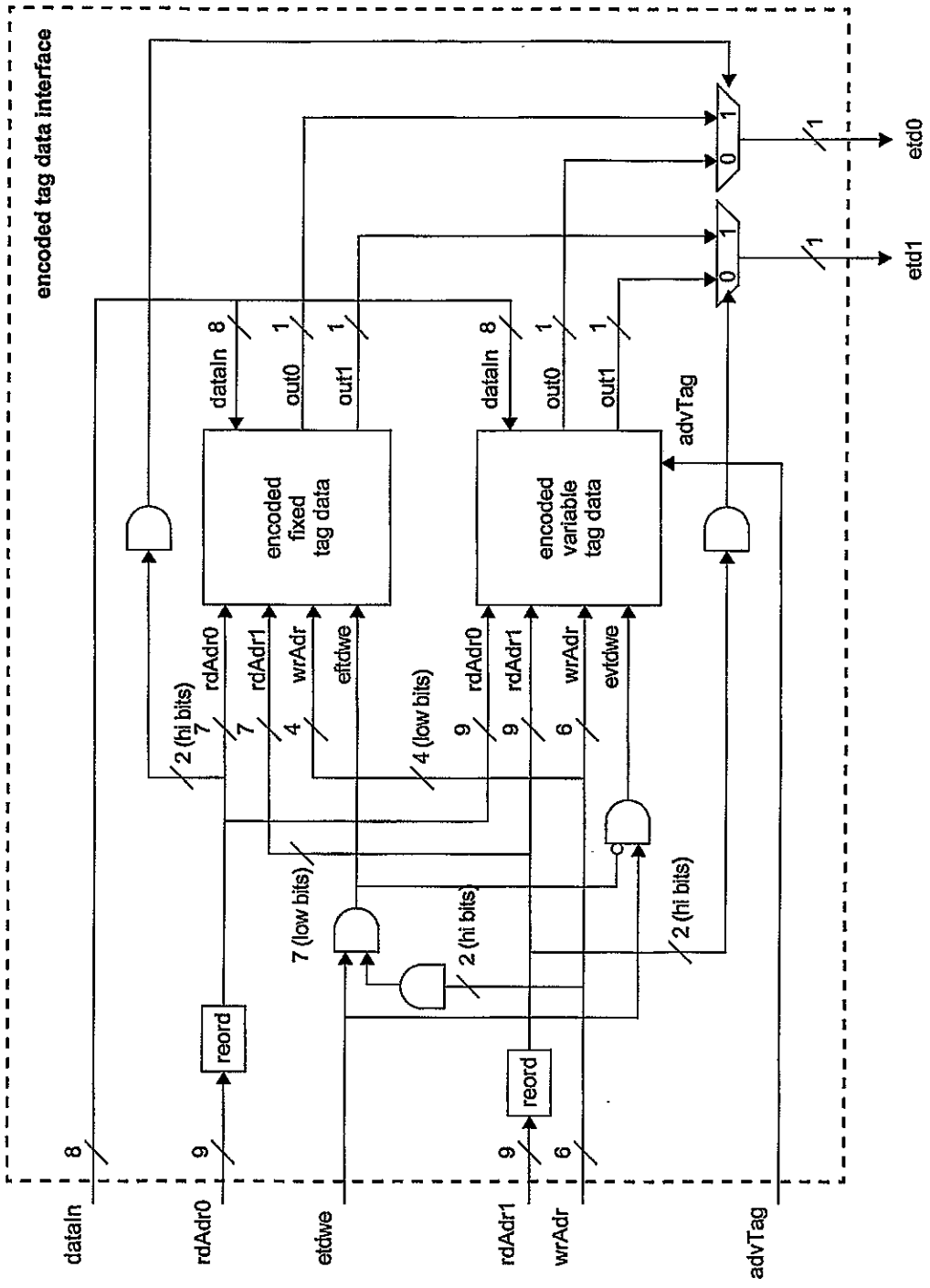

encoded tag data interface
dataIn
rdAdr0
etdwe
rdAdr1
wrAdr
advTag
reord
reord
encoded fixed tag data
encoded variable tag data
dataIn
out0
out1
rdAdr0
rdAdr1
wrAdr
etdwe
evtdwe
advTag
2 (hi bits)
4 (low bits)
7 (low bits)
2 (hi bits)
2 (hi bits)
etd0
etd1

FIG. 217

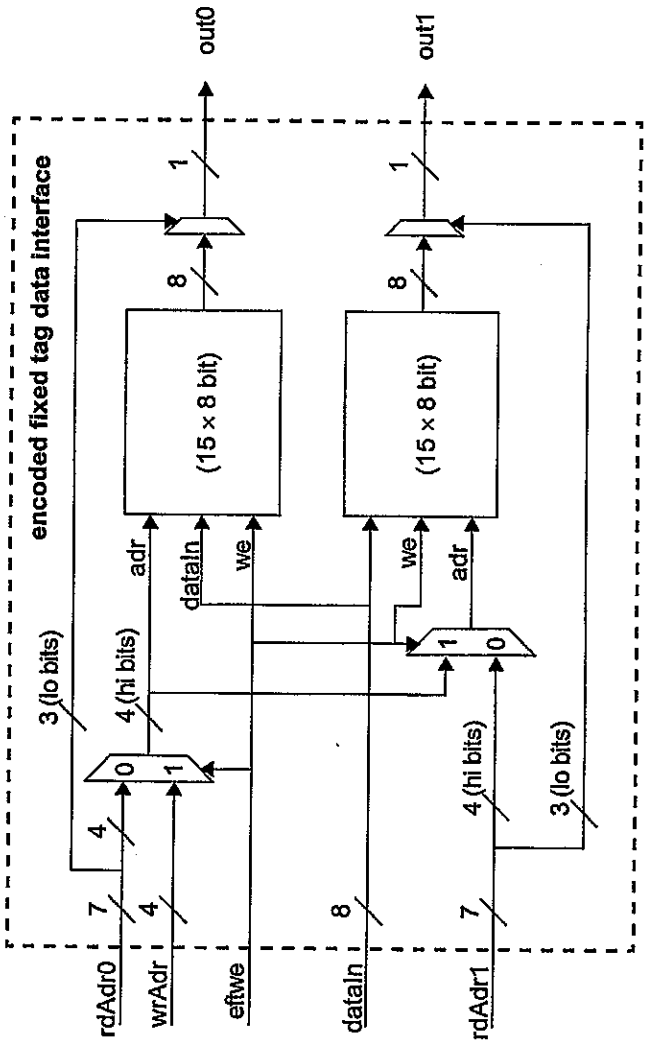

encoded fixed tag data interface
out0
out1
1
8
(15 × 8 bit)
adr
dataIn
we
3 (lo bits)
4 (hi bits)
0
1
4
7
rdAdr0
wrAdr
4
eftwe
8
dataIn
7
rdAdr1
4 (hi bits)
3 (lo bits)

FIG. 218

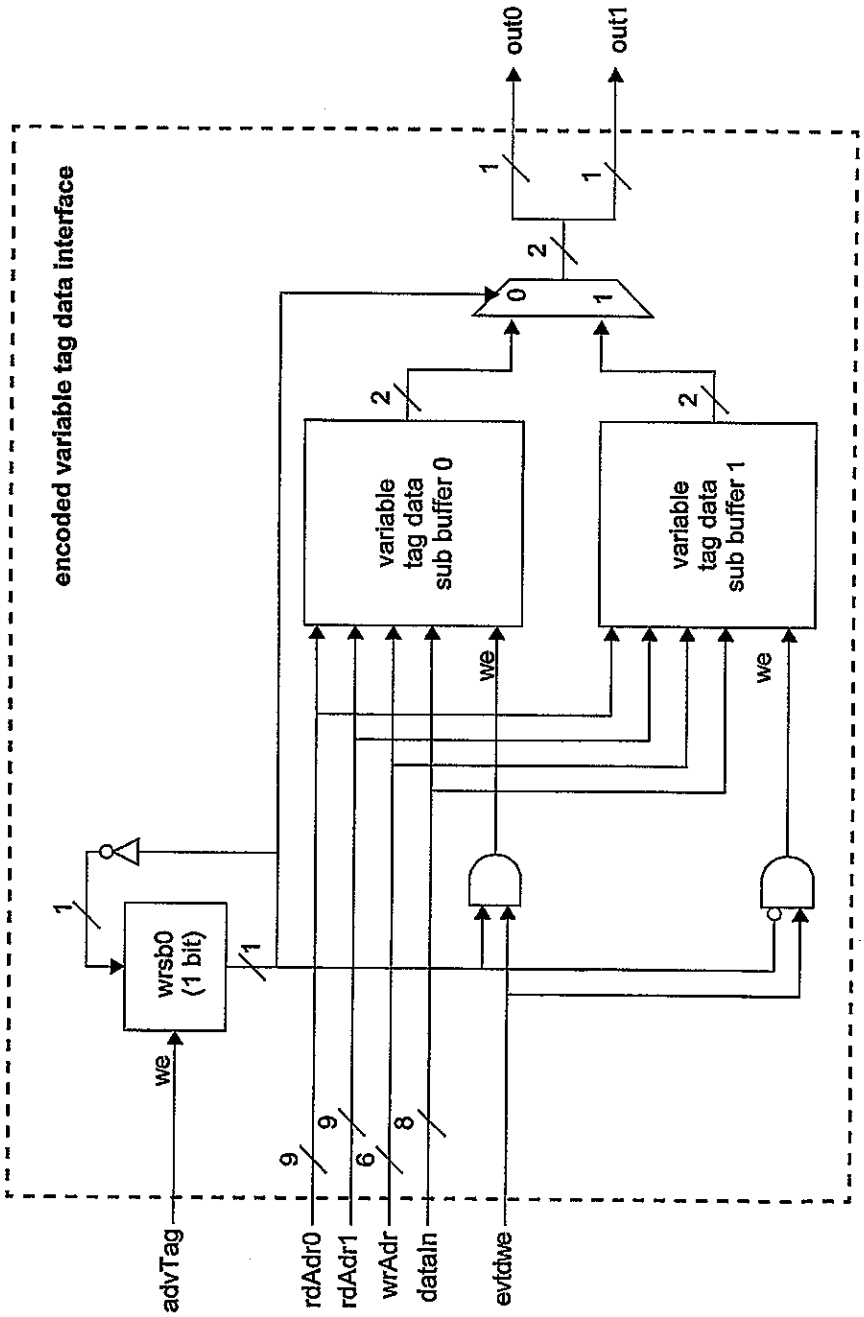

encoded variable tag data interface
out0
out1
variable tag data sub buffer 0
variable tag data sub buffer 1
wrsb0 (1 bit)
we
advTag
rdAdr0
rdAdr1
wrAdr
dataIn
evtdwe

FIG. 219

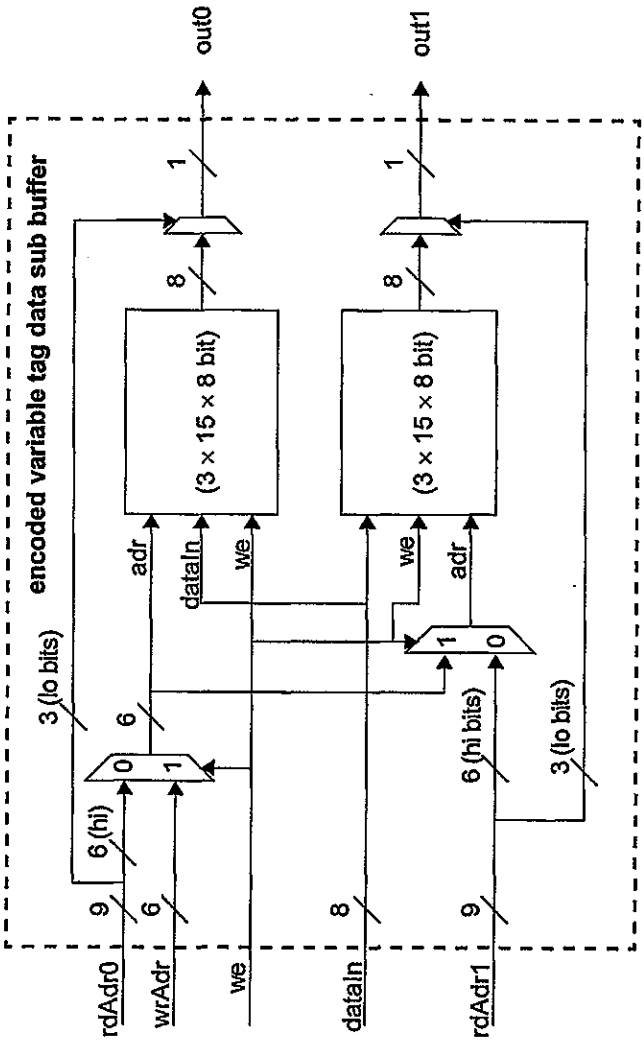

encoded variable tag data sub buffer
out0
out1
(3 x 15 x 8 bit)
(3 x 15 x 8 bit)
adr
dataIn
we
3 (lo bits)
6 (hi)
6 (hi bits)
rdAdr0
wrAdr
rdAdr1

FIG. 220

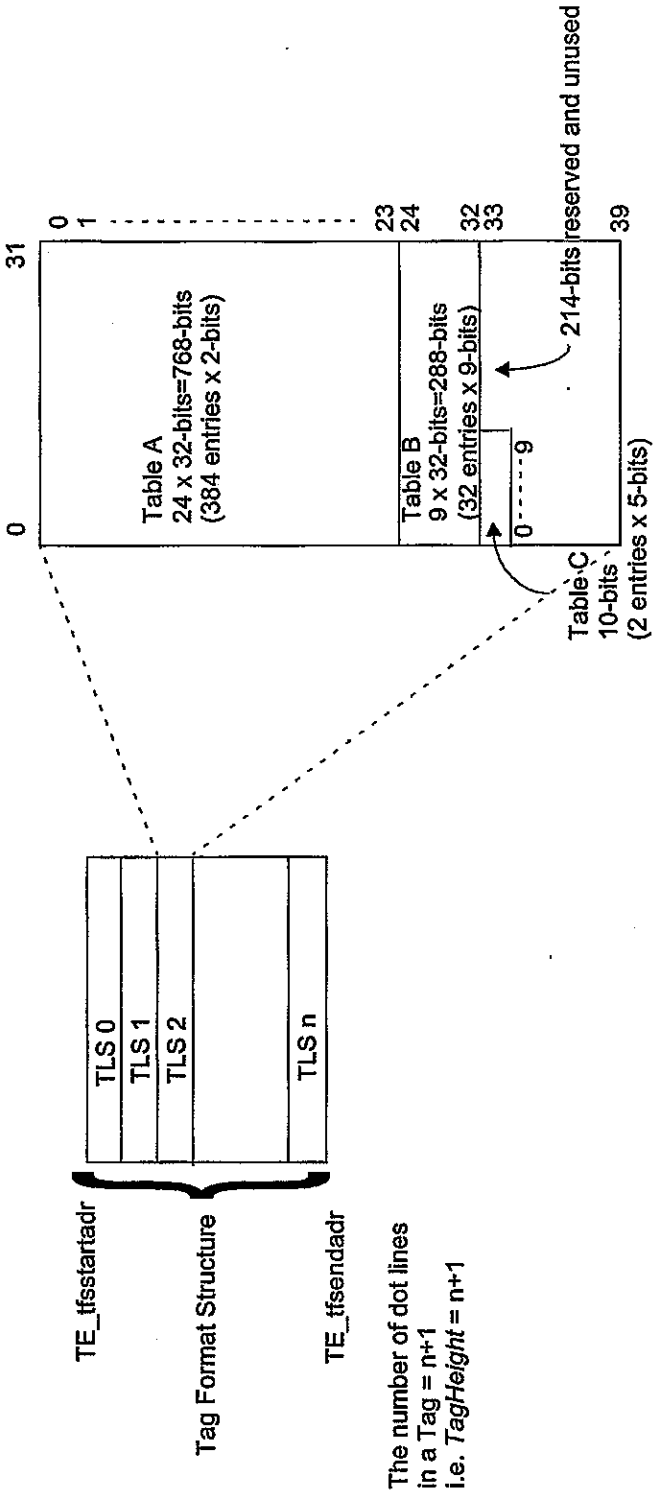

0
31
0
1
23
24
32
33
39
Table A
24 x 32-bits=768-bits
(384 entries x 2-bits)
Table B
9 x 32-bits=288-bits
(32 entries x 9-bits)
214-bits reserved and unused
0
9
Table C
10-bits
(2 entries x 5-bits)
TLS 0
TLS 1
TLS 2
TLS n
TE_tfsstartadr
Tag Format Structure
TE_tfsendadr
The number of dot lines
in a Tag = n+1
i.e. TagHeight = n+1

FIG. 221

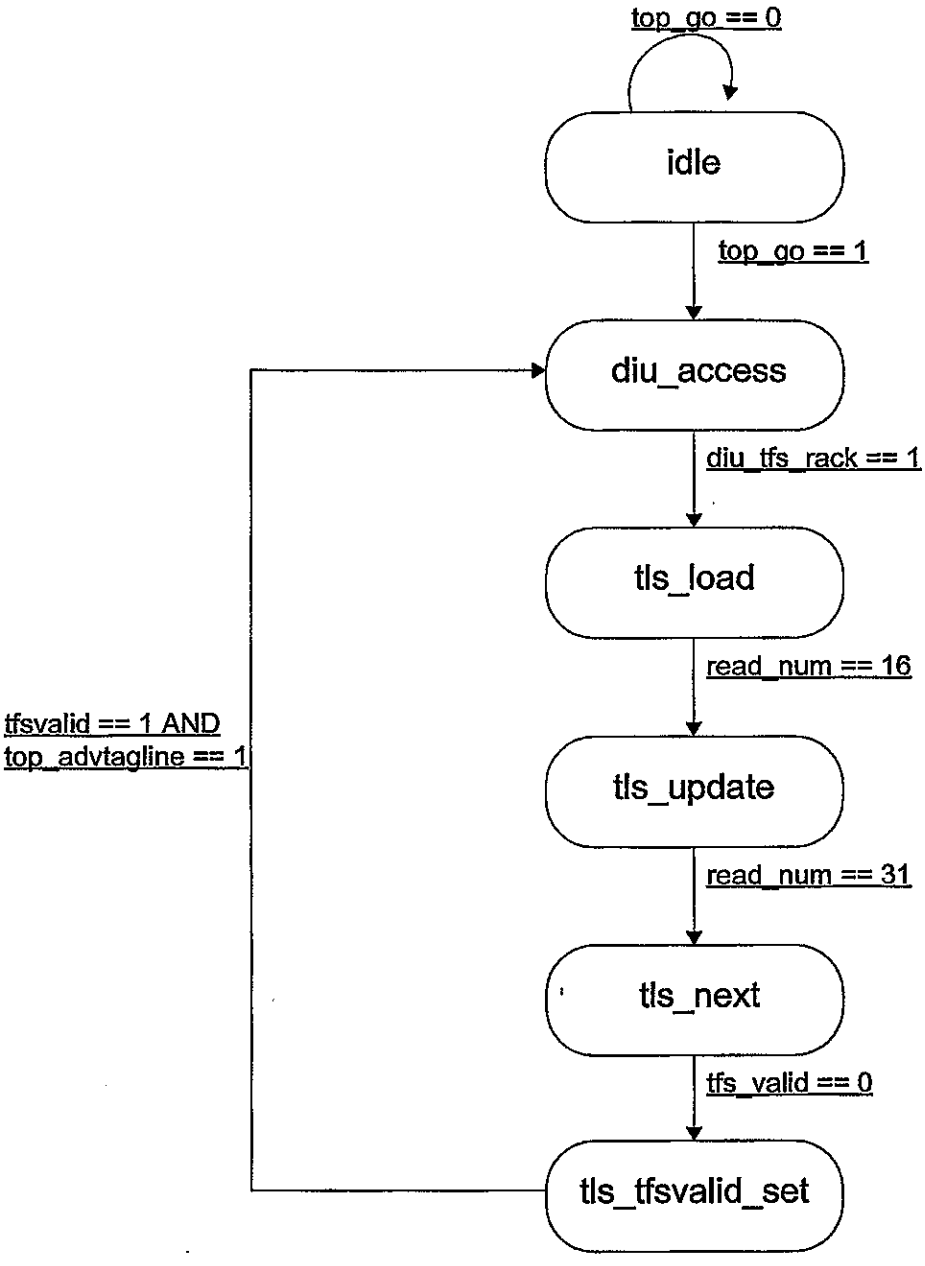

top_go == 0
idle
top_go == 1
diu_access
diu_tfs_rack == 1
tls_load
read_num == 16
tfsvalid == 1 AND
top_advtagline == 1
tls_update
read_num == 31
tls_next
tfs_valid == 0
tls_tfsvalid_set

FIG. 222

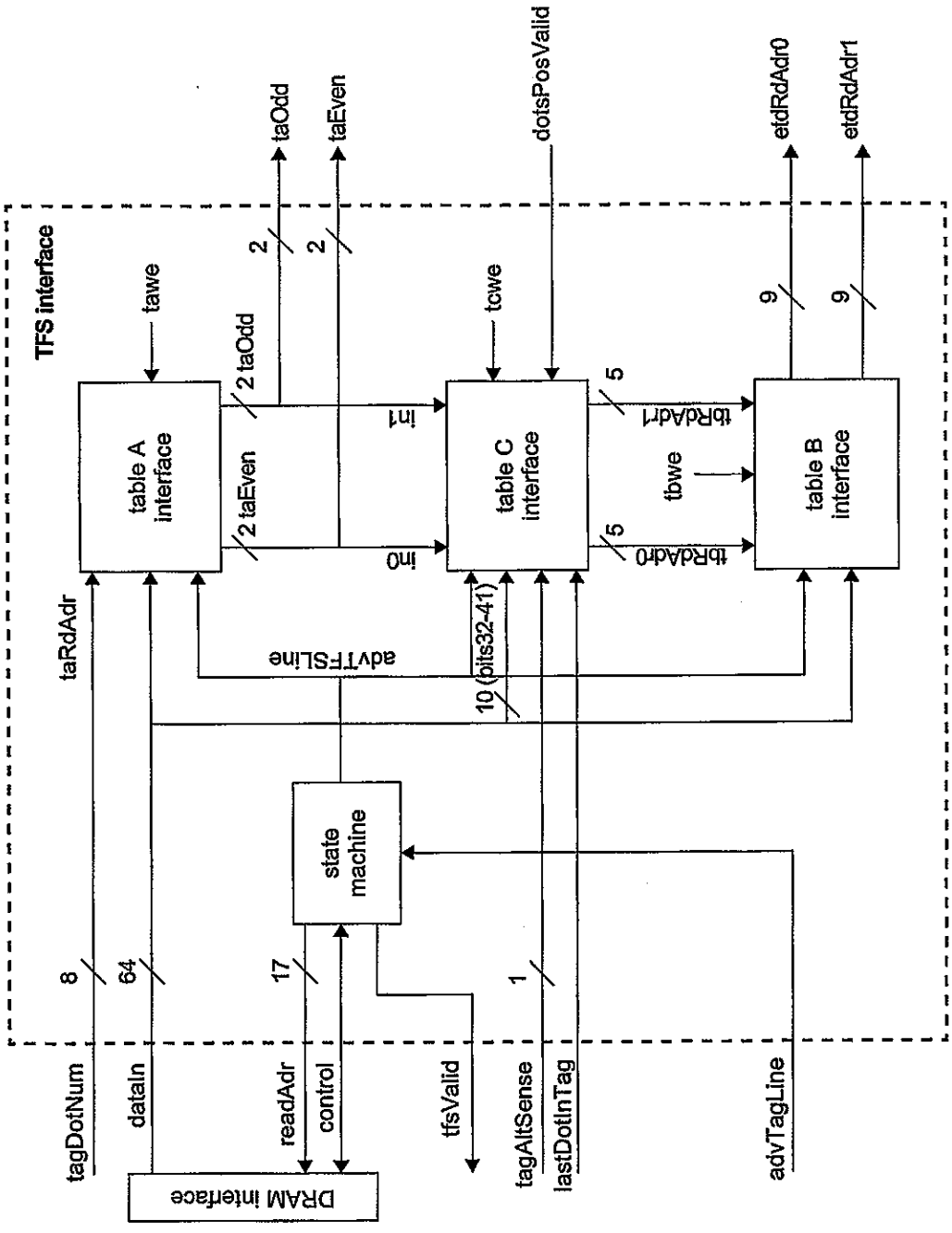

TFS interface
tawe
taOdd
taEven
dotsPosValid
etdRdAdr0
etdRdAdr1
2
2
9
9
2 taOdd
2 taEven
table A interface
in1
in0
tcwe
table C interface
5
5
tbRdAdr1
tbRdAdr0
tbwe
table B interface
taRdAdr
advTFSLine
10 (bits32-41)
state machine
8
64
17
1
tagDotNum
dataIn
readAdr
control
tfsValid
tagAltSense
lastDotInTag
advTagLine
DRAM interface

FIG. 223

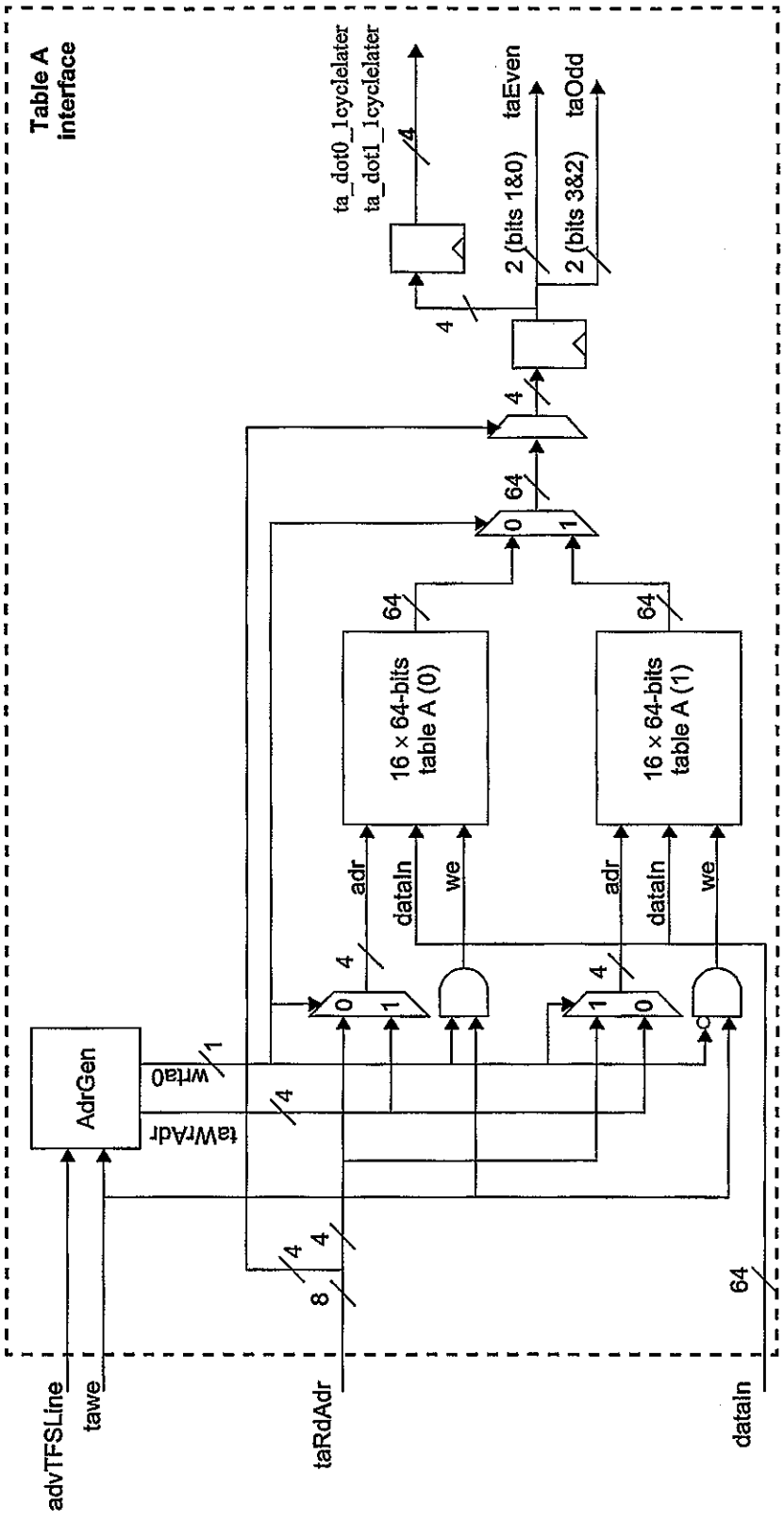

Table A interface
AdrGen
wrta0
taWrAdr
advTFSLine
tawe
taRdAdr
datain
16 × 64-bits table A (0)
16 × 64-bits table A (1)
adr
datain
we
ta_dot0_1cyclelater
ta_dot1_1cyclelater
2 (bits 1&0) taEven
2 (bits 3&2) taOdd

FIG. 224

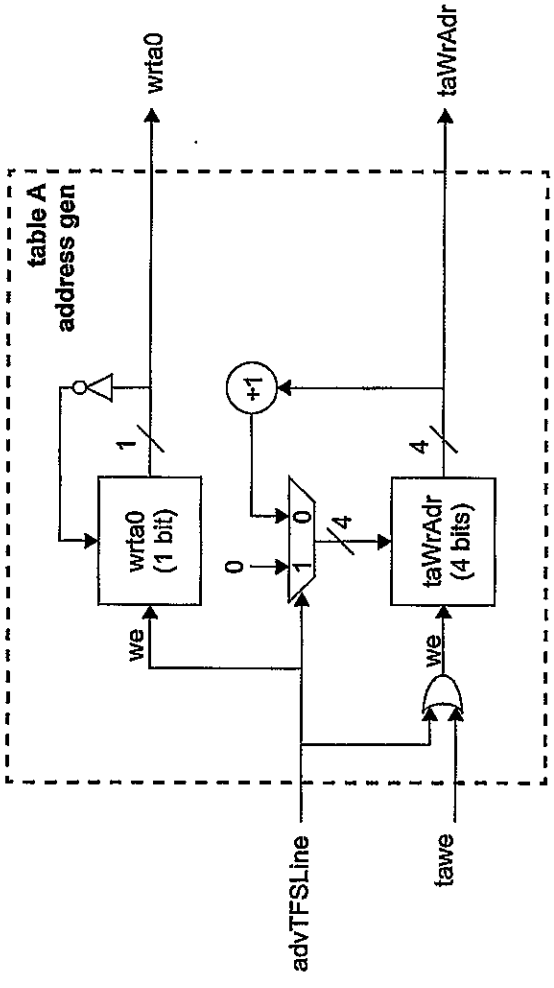

table A
address gen
wrta0
taWrAdr
wrta0
(1 bit)
taWrAdr
(4 bits)
+1
1
4
0
1
0
4
we
we
advTFSLine
tawe

FIG. 225

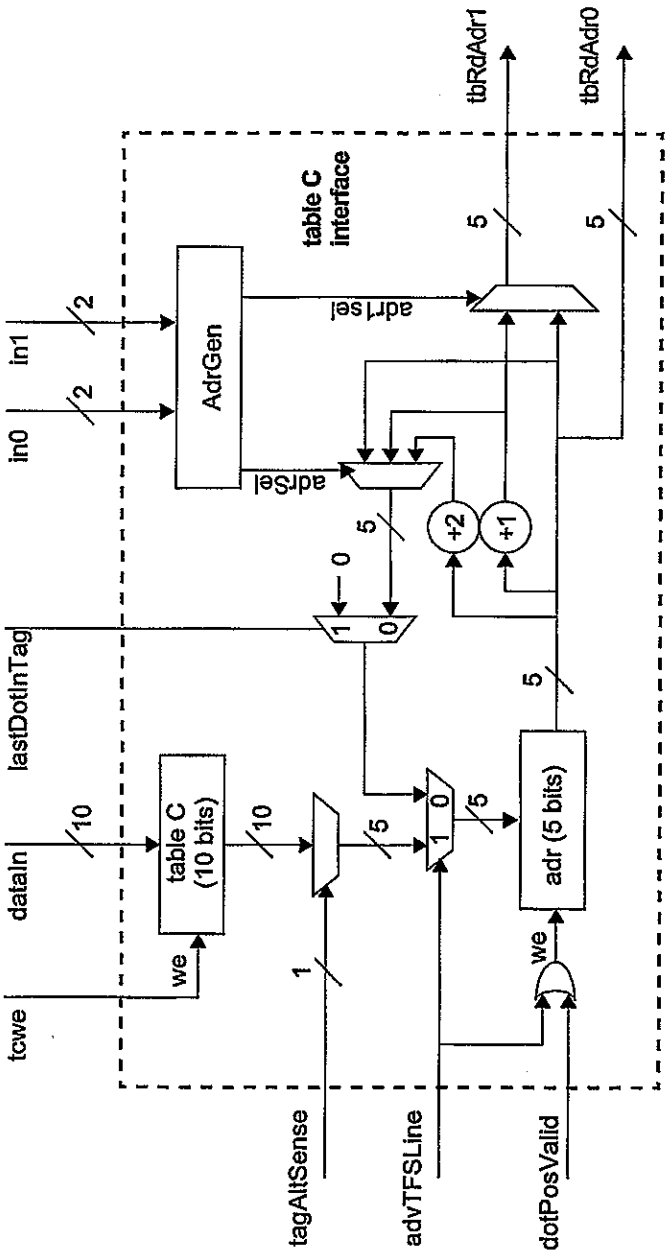

tbRdAdr1
tbRdAdr0
table C interface
AdrGen
adrTsel
adrSel
in1
in0
2
2
5
5
+2
+1
0
1
0
5
lastDotInTag
dataIn
10
table C (10 bits)
10
5
1
0
5
adr (5 bits)
5
tcwe
we
1
we
tagAltSense
advTFSLine
dotPosValid

FIG. 226

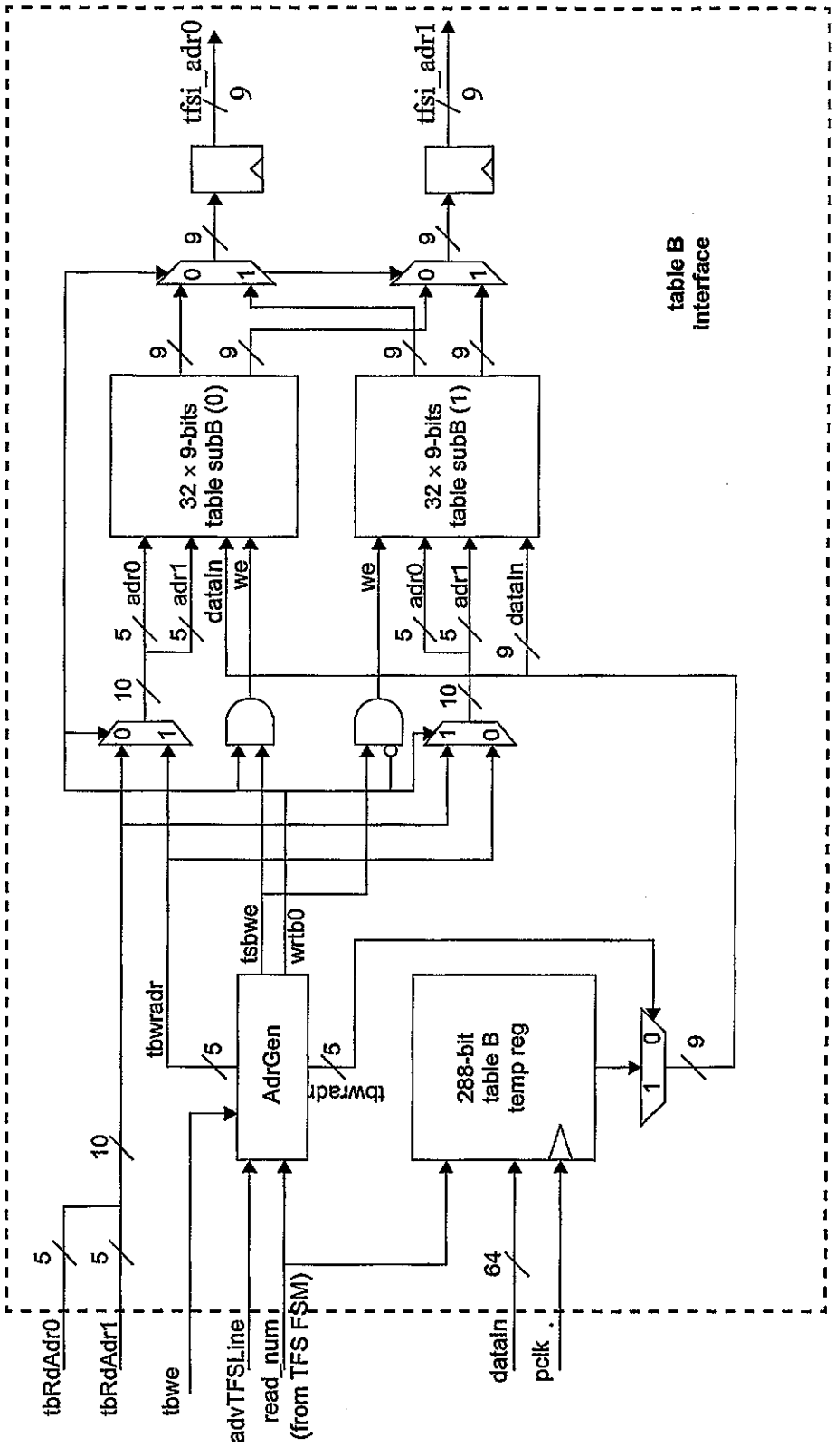

table B interface
tfsi_adr0
tfsi_adr1
32 × 9-bits table subB (0)
32 × 9-bits table subB (1)
adr0
adr1
dataIn
we
tsbwe
wrtb0
tbwradr
AdrGen
288-bit table B temp reg
tbRdAdr0
tbRdAdr1
tbwe
advTFSLine
read_num (from TFS FSM)
dataIn
pclk

FIG. 227

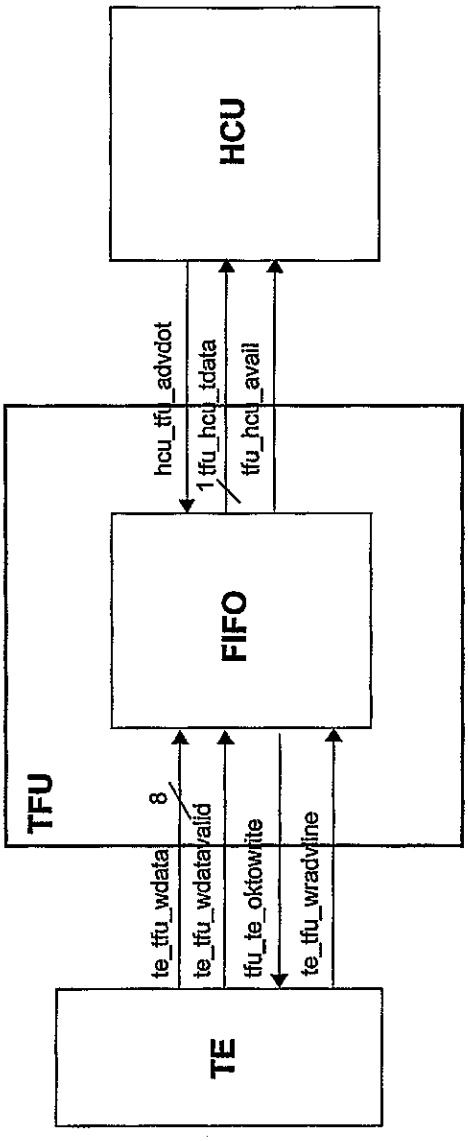


FIG. 228

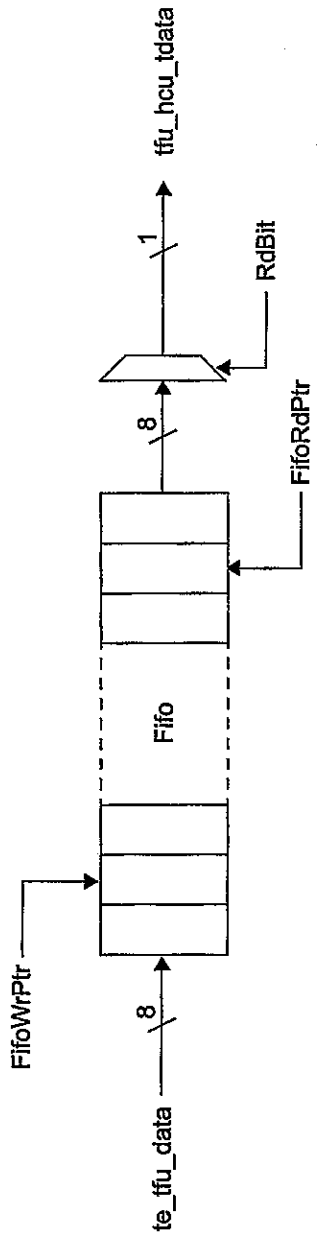


FIG. 229

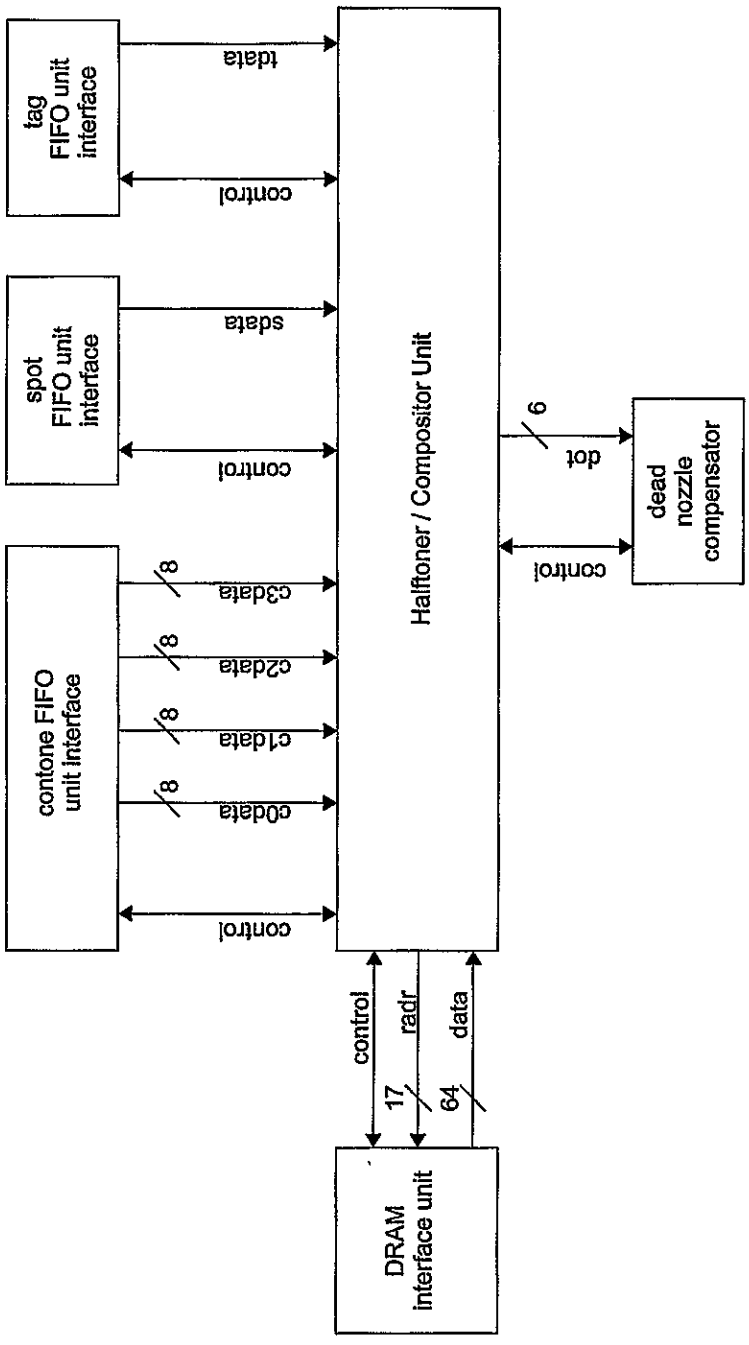

tag FIFO unit interface
tdata
control
spot FIFO unit interface
sdata
control
contone FIFO unit interface
8
c3data
8
c2data
8
c1data
8
c0data
control
Halftoner / Compositor Unit
6
dot
control
dead nozzle compensator
control
17
radr
64
data
DRAM interface unit

FIG. 230

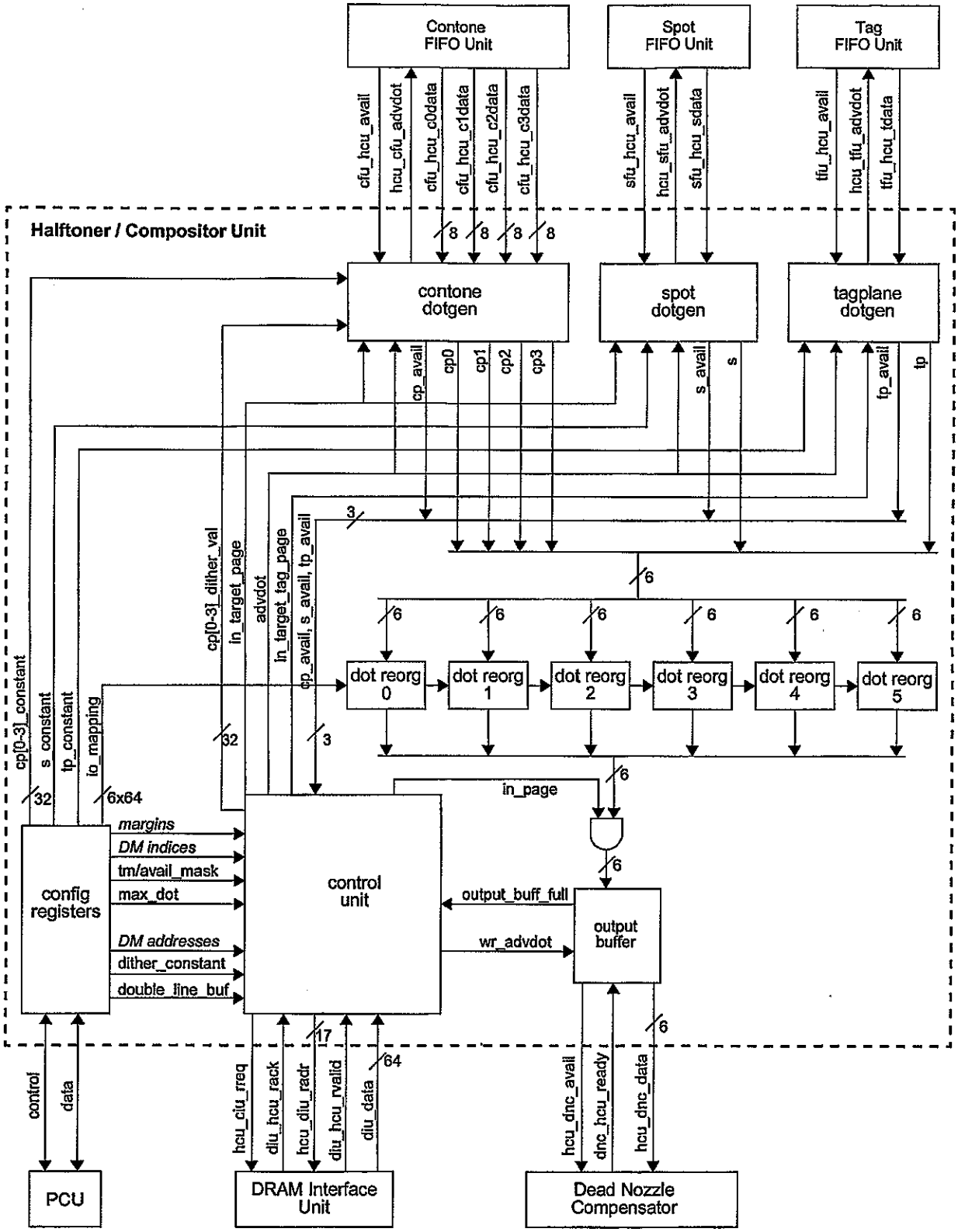

Contone FIFO Unit
Spot FIFO Unit
Tag FIFO Unit
cfu_hcu_avail
hcu_cfu_advdot
cfu_hcu_c0data
cfu_hcu_c1data
cfu_hcu_c2data
cfu_hcu_c3data
sfu_hcu_avail
hcu_sfu_advdot
sfu_hcu_sdata
tfu_hcu_avail
hcu_tfu_advdot
tfu_hcu_tdata
Halftoner / Compositor Unit
contone dotgen
spot dotgen
tagplane dotgen
cp_avail
cp0
cp1
cp2
cp3
s_avail
s
tp_avail
tp
cp[0-3]_dither_val
in_target_page
advdot
in_target_tag_page
cp_avail, s_avail, tp_avail
cp[0-3]_constant
s_constant
tp_constant
io_mapping
dot reorg 0
dot reorg 1
dot reorg 2
dot reorg 3
dot reorg 4
dot reorg 5
in_page
margins
DM indices
tm/avail_mask
max_dot
DM addresses
dither_constant
double_line_buf
config registers
control unit
output_buff_full
wr_advdot
output buffer
control
data
hcu_ciu_freq
diu_hcu_rack
hcu_diu_radr
diu_hcu_rvalid
diu_data
hcu_dnc_avail
dnc_hcu_ready
hcu_dnc_data
PCU
DRAM Interface Unit
Dead Nozzle Compensator

FIG. 231

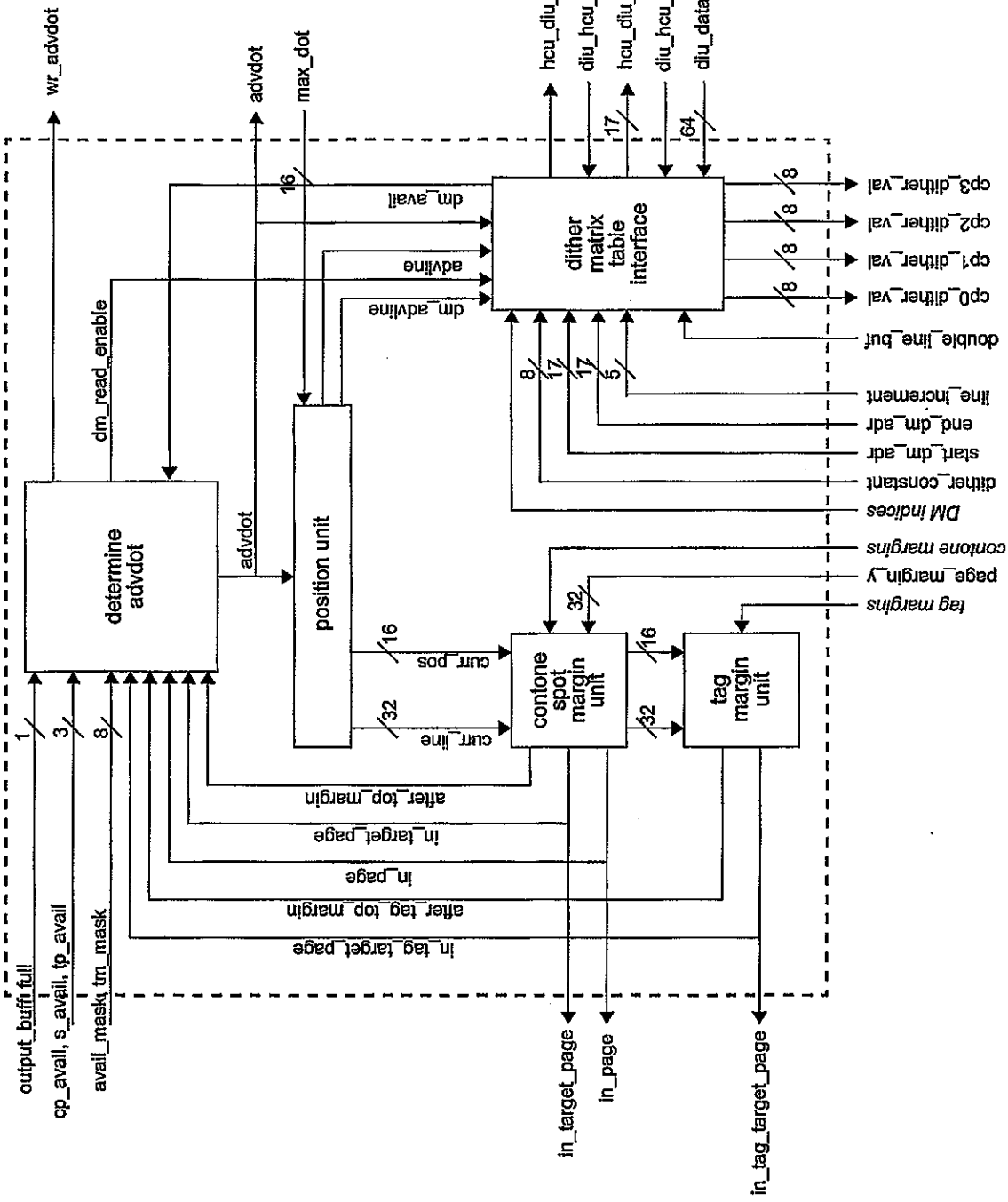

wr_advdot
advdot
max_dot
hcu_diu_rreq
diu_hcu_rack
hcu_diu_radr
diu_hcu_rvalid
diu_data
dm_read_enable
dm_avail
advline
dm_advline
dither matrix table interface
cp0_dither_val
cp1_dither_val
cp2_dither_val
cp3_dither_val
double_line_buf
line_increment
end_dm_adr
start_dm_adr
dither_constant
DM indices
contone margins
page_margin_y
tag margins
determine advdot
position unit
contone spot margin unit
tag margin unit
curr_pos
curr_line
after_top_margin
in_target_page
in_page
after tag top margin
in_tag_target_page
output_buffi full
cp_avail, s_avail, tp_avail
avail_mask, tm_mask
in_target_page
in_page
in_tag_target_page

FIG. 232

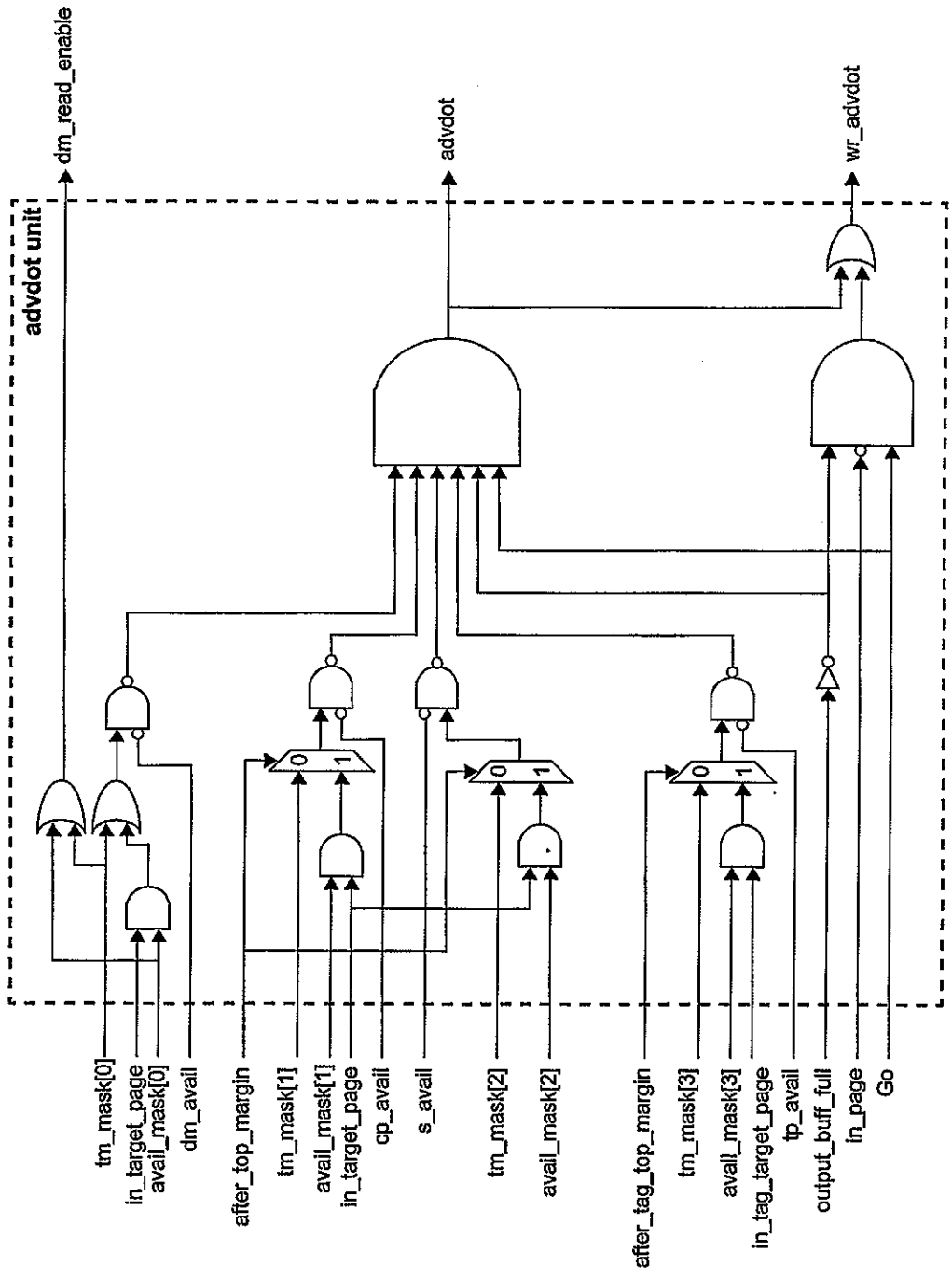

advdot unit
dm_read_enable
advdot
wr_advdot
tm_mask[0]
in_target_page
avail_mask[0]
dm_avail
after_top_margin
tm_mask[1]
avail_mask[1]
in_target_page
cp_avail
s_avail
tm_mask[2]
avail_mask[2]
after_tag_top_margin
tm_mask[3]
avail_mask[3]
in_tag_target_page
tp_avail
output_buff_full
in_page
Go

FIG. 233

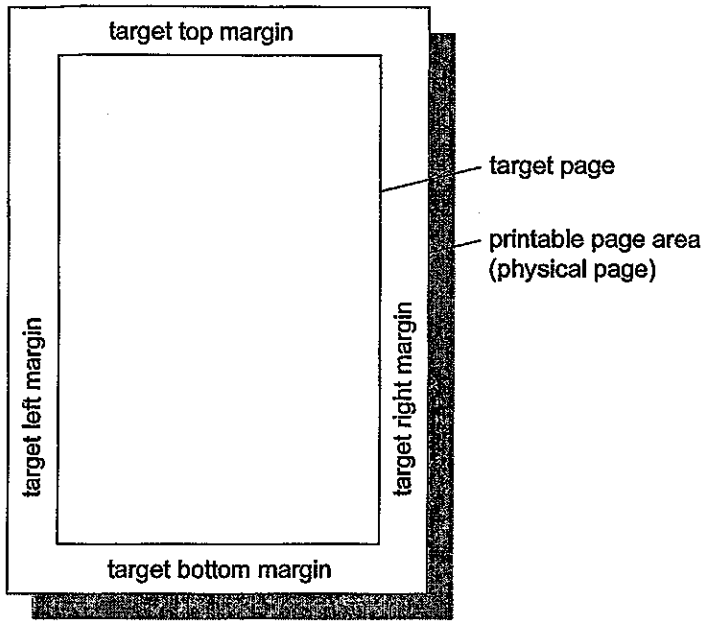

target top margin
target page
printable page area
(physical page)
target left margin
target right margin
target bottom margin

FIG. 234

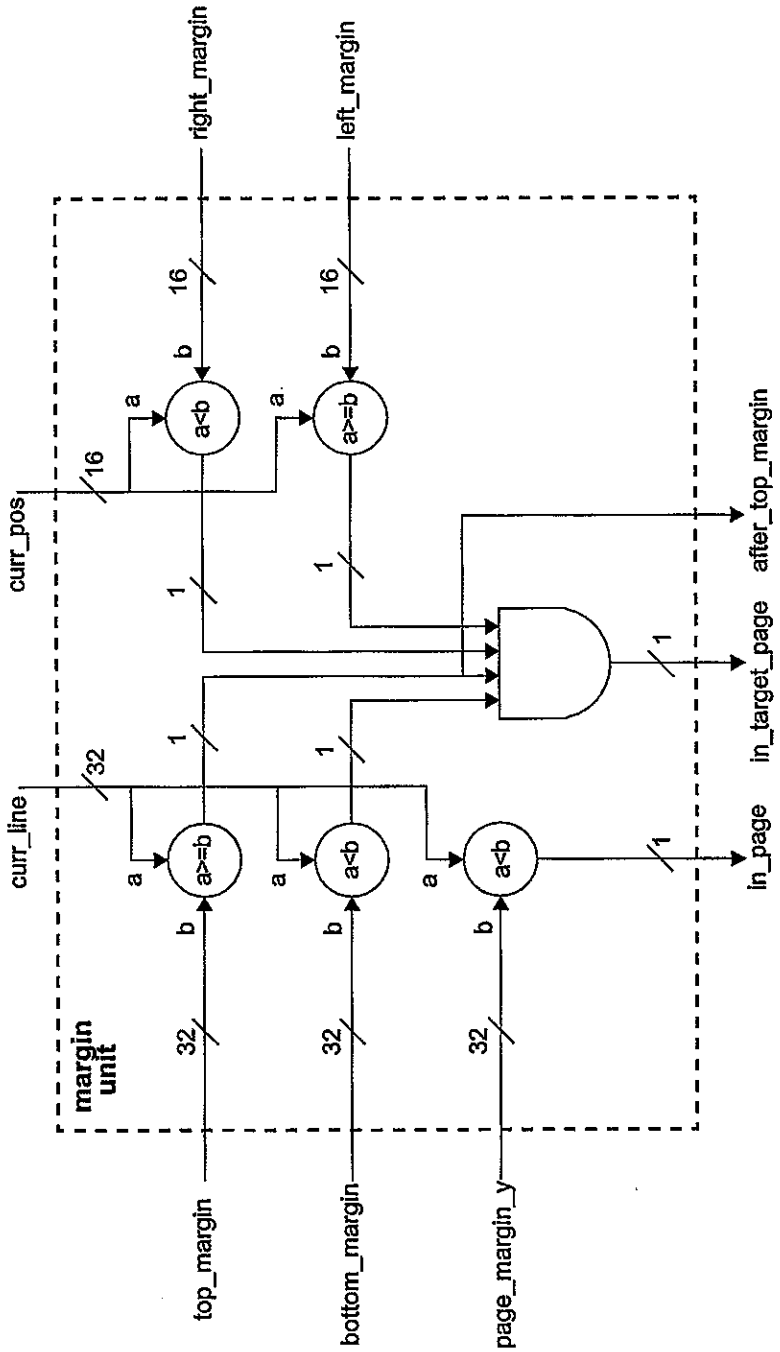

right_margin
left_margin
curr_pos
curr_line
top_margin
bottom_margin
page_margin_y
margin unit
a<b
a>=b
a>=b
a<b
a<b
in_target_page
after_top_margin
in_page
16
32
1

FIG. 235

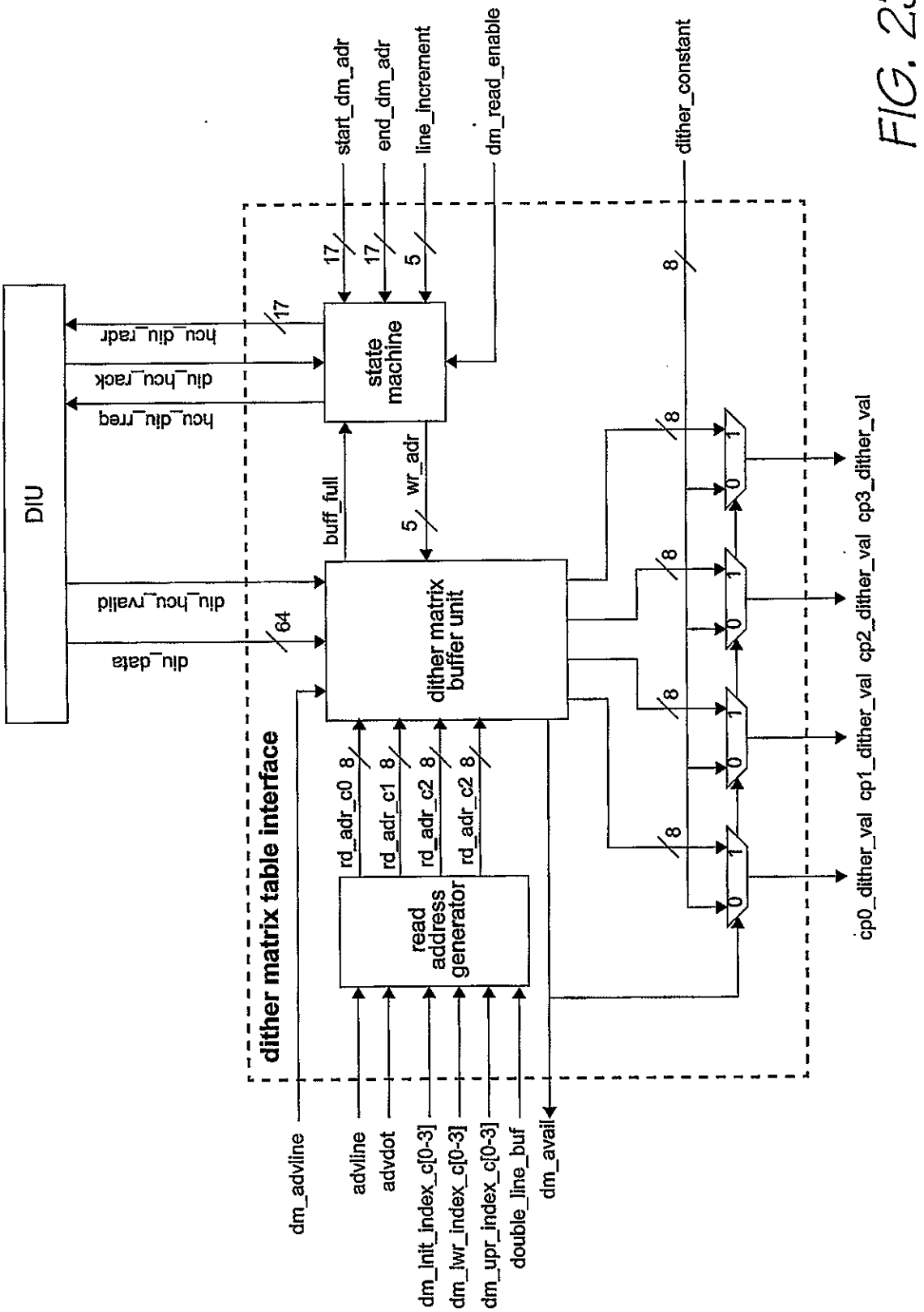

DIU
dither matrix table interface
state machine
dither matrix buffer unit
read address generator
hcu_diu_radr
diu_hcu_rack
hcu_diu_rreq
diu_hcu_rvalid
diu_data
buff_full
wr_adr
start_dm_adr
end_dm_adr
line_increment
dm_read_enable
dither_constant
rd_adr_c0
rd_adr_c1
rd_adr_c2
rd_adr_c2
dm_advline
advline
advdot
dm_init_index_c[0-3]
dm_iwr_index_c[0-3]
dm_upr_index_c[0-3]
double_line_buf
dm_avail
cp0_dither_val
cp1_dither_val
cp2_dither_val
cp3_dither_val

FIG. 236

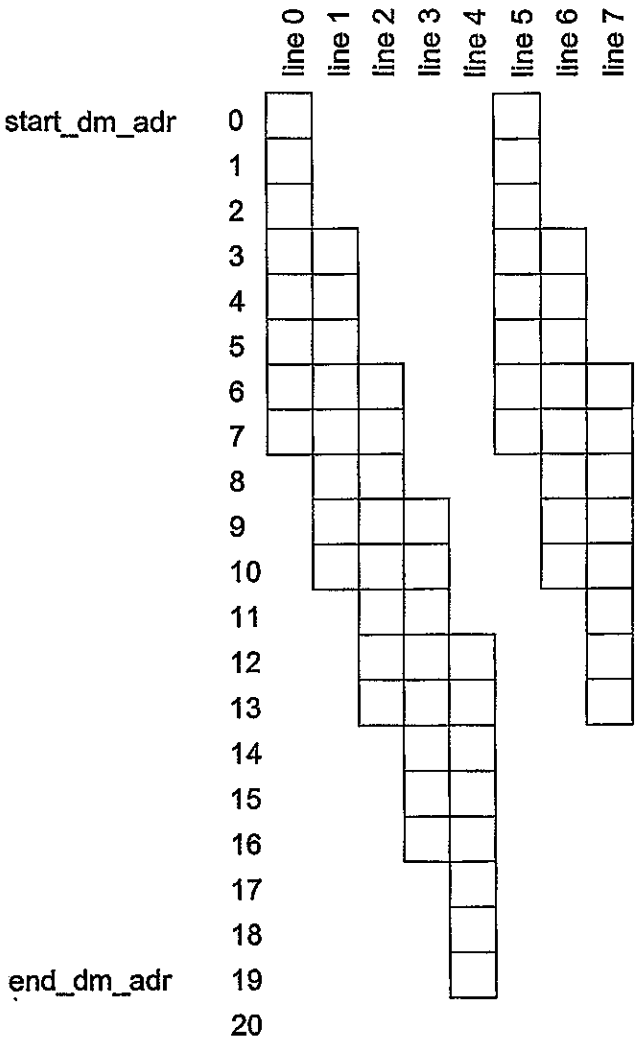

line 0
line 1
line 2
line 3
line 4
line 5
line 6
line 7
start_dm_adr
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
end_dm_adr
line_increment = 3
double_line_buf = 0

FIG. 237

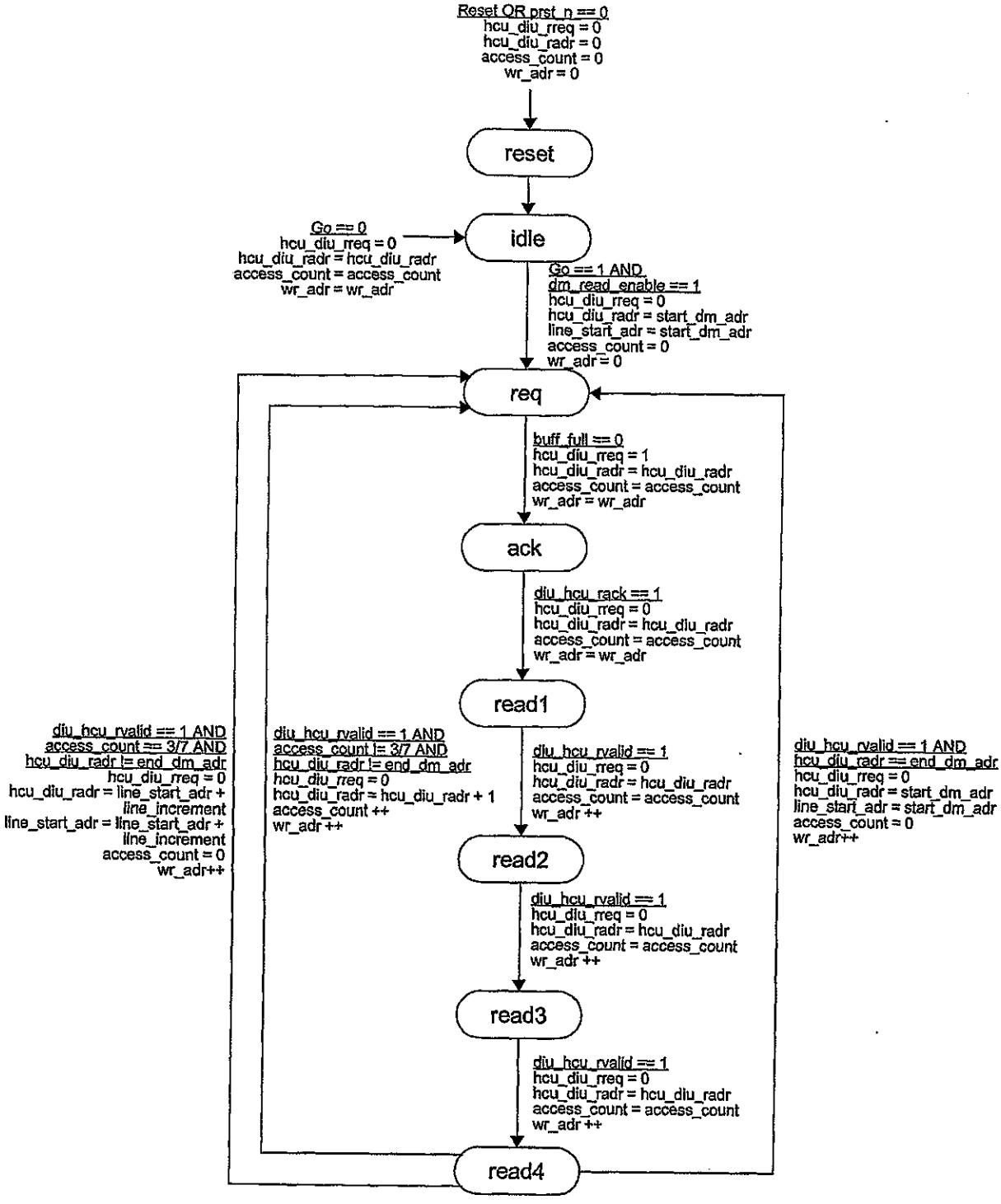

Reset OR prst_n == 0
hcu_diu_rreq = 0
hcu_diu_radr = 0
access_count = 0
wr_adr = 0
reset
Go == 0
hcu_diu_rreq = 0
hcu_diu_radr = hcu_diu_radr
access_count = access_count
wr_adr = wr_adr
idle
Go == 1 AND
dm_read_enable == 1
hcu_diu_rreq = 0
hcu_diu_radr = start_dm_adr
line_start_adr = start_dm_adr
access_count = 0
wr_adr = 0
req
buff_full == 0
hcu_diu_rreq = 1
hcu_diu_radr = hcu_diu_radr
access_count = access_count
wr_adr = wr_adr
ack
diu_hcu_rack == 1
hcu_diu_rreq = 0
hcu_diu_radr = hcu_diu_radr
access_count = access_count
wr_adr = wr_adr
read1
diu_hcu_rvalid == 1 AND
access_count == 3/7 AND
hcu_diu_radr != end_dm_adr
hcu_diu_rreq = 0
hcu_diu_radr = line_start_adr +
line_increment
line_start_adr = line_start_adr +
line_increment
access_count = 0
wr_adr++
diu_hcu_rvalid == 1 AND
access_count != 3/7 AND
hcu_diu_radr != end_dm_adr
hcu_diu_rreq = 0
hcu_diu_radr = hcu_diu_radr + 1
access_count ++
wr_adr ++
diu_hcu_rvalid == 1
hcu_diu_rreq = 0
hcu_diu_radr = hcu_diu_radr
access_count = access_count
wr_adr ++
diu_hcu_rvalid == 1 AND
hcu_diu_radr == end_dm_adr
hcu_diu_rreq = 0
hcu_diu_radr = start_dm_adr
line_start_adr = start_dm_adr
access_count = 0
wr_adr++
read2
diu_hcu_rvalid == 1
hcu_diu_rreq = 0
hcu_diu_radr = hcu_diu_radr
access_count = access_count
wr_adr ++
read3
diu_hcu_rvalid == 1
hcu_diu_rreq = 0
hcu_diu_radr = hcu_diu_radr
access_count = access_count
wr_adr ++
read4

FIG. 238

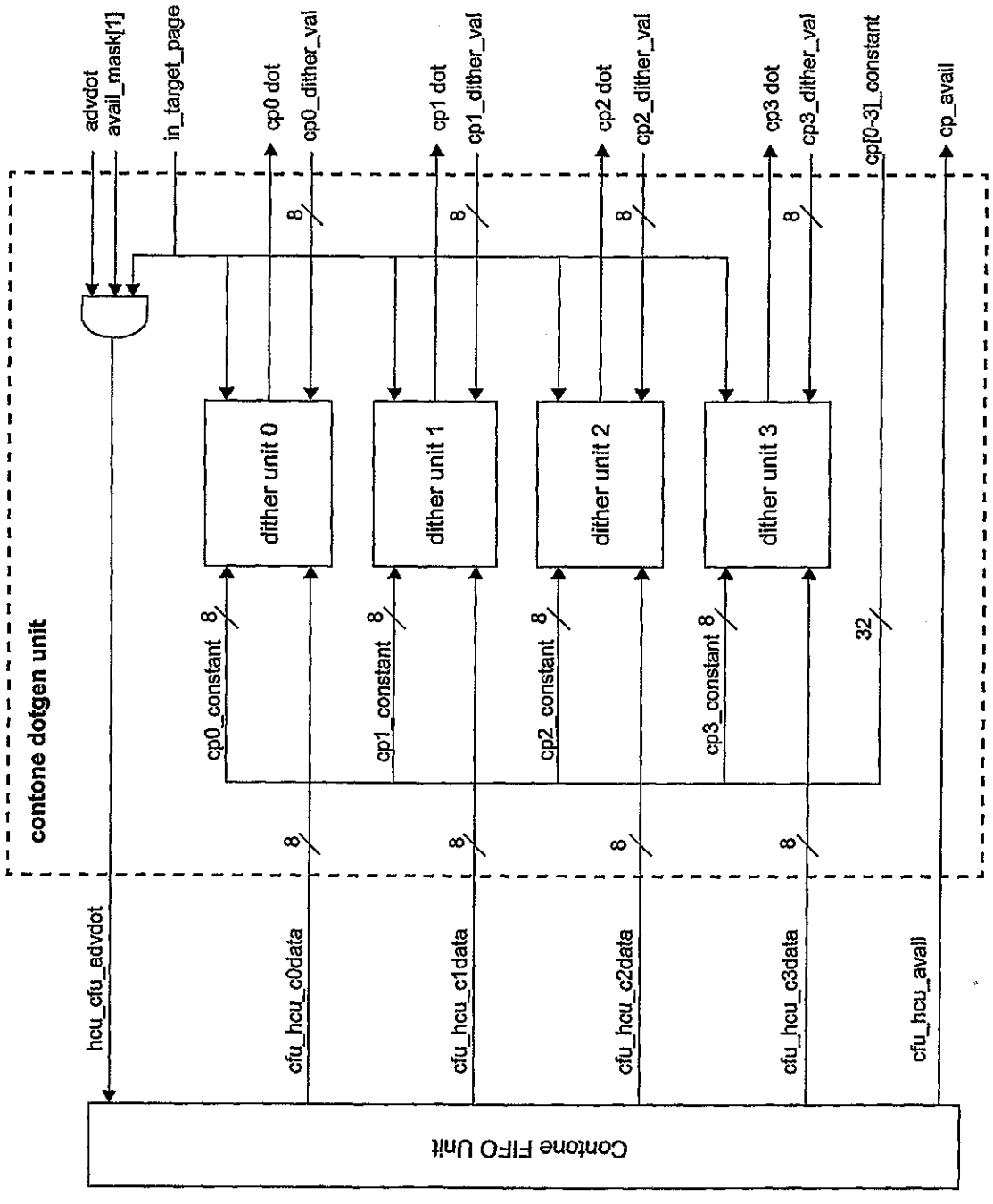

contone dotgen unit
advdot
avail_mask[1]
in_target_page
cp0 dot
cp0_dither_val
cp1 dot
cp1_dither_val
cp2 dot
cp2_dither_val
cp3 dot
cp3_dither_val
cp[0-3]_constant
cp_avail
dither unit 0
dither unit 1
dither unit 2
dither unit 3
cp0_constant
cp1_constant
cp2_constant
cp3_constant
8
32
hcu_cfu_advdot
cfu_hcu_c0data
cfu_hcu_c1data
cfu_hcu_c2data
cfu_hcu_c3data
cfu_hcu_avail
Contone FIFO Unit

FIG. 239

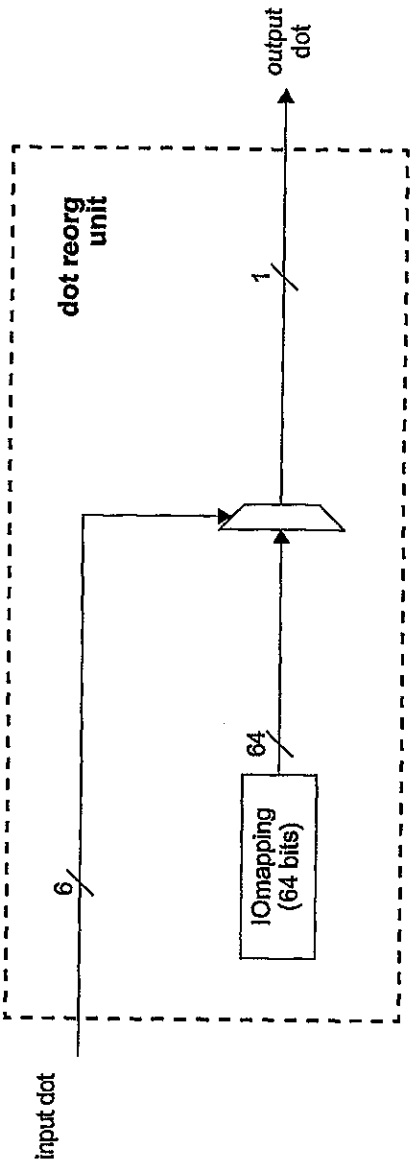


FIG. 240

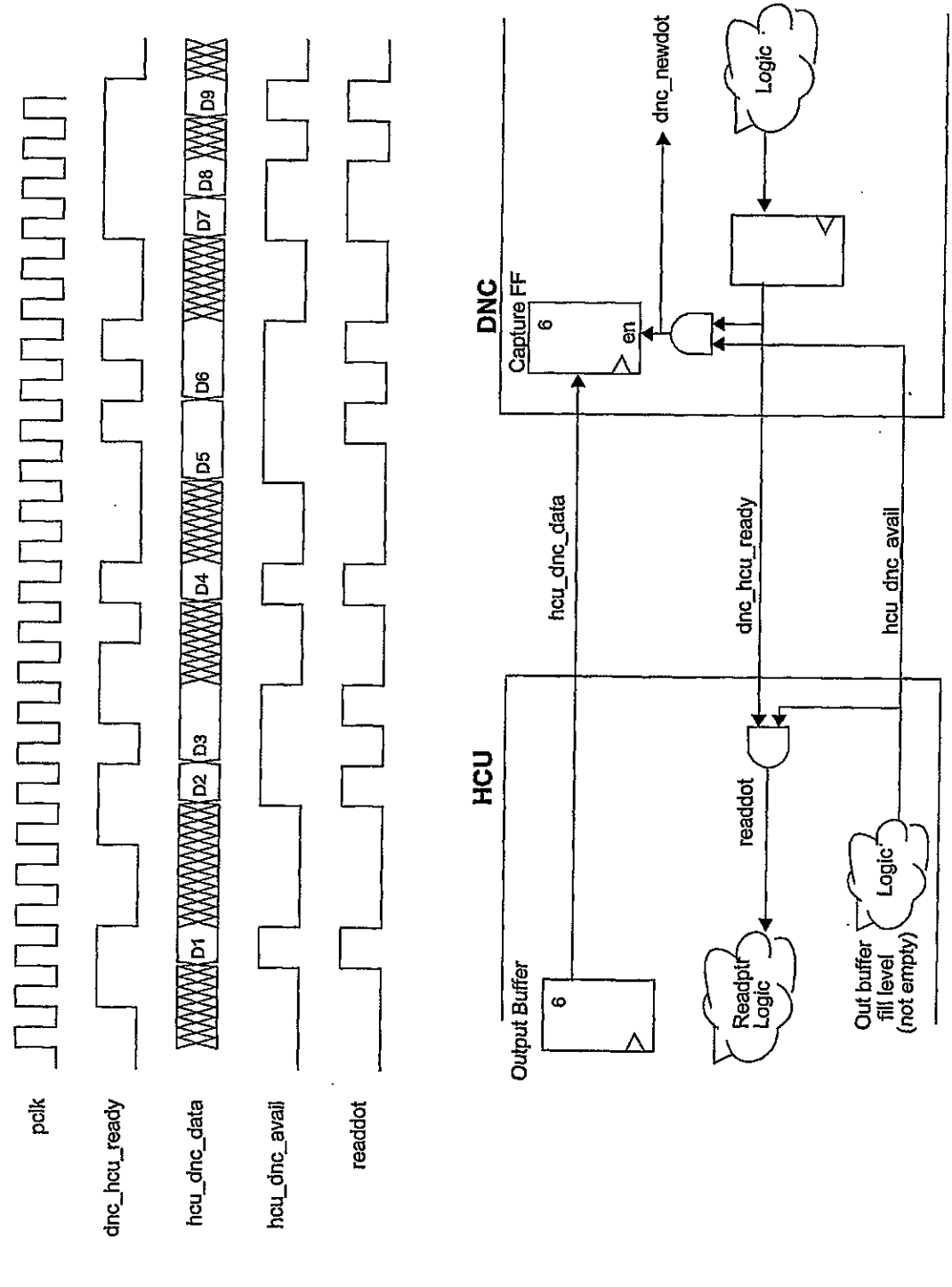

pclk
dnc_hcu_ready
hcu_dnc_data
hcu_dnc_avail
readdot
D1
D2
D3
D4
D5
D6
D7
D8
D9
HCU
Output Buffer
6
hcu_dnc_data
DNC
Capture FF
6
en
dnc_newdot
Logic
dnc_hcu_ready
hcu_dnc_avail
readdot
Readptr Logic
Logic
Out buffer fill level (not empty)

FIG. 241

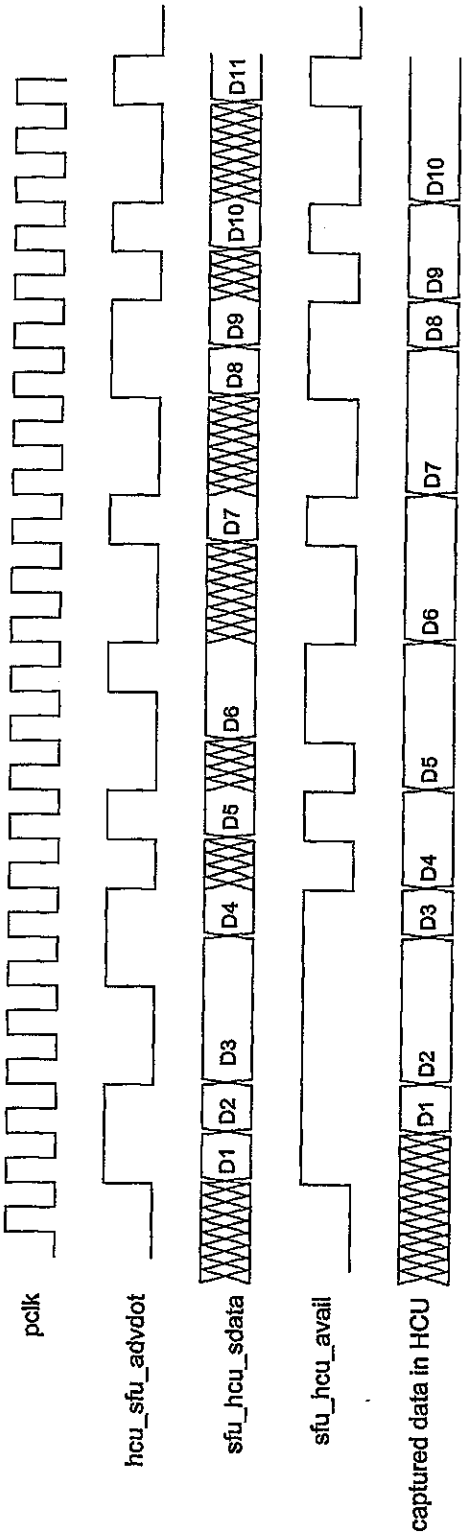

pclk
hcu_sfu_advdot
sfu_hcu_sdata
sfu_hcu_avail
captured data in HCU
D1
D2
D3
D4
D5
D6
D7
D8
D9
D10
D11

FIG. 242

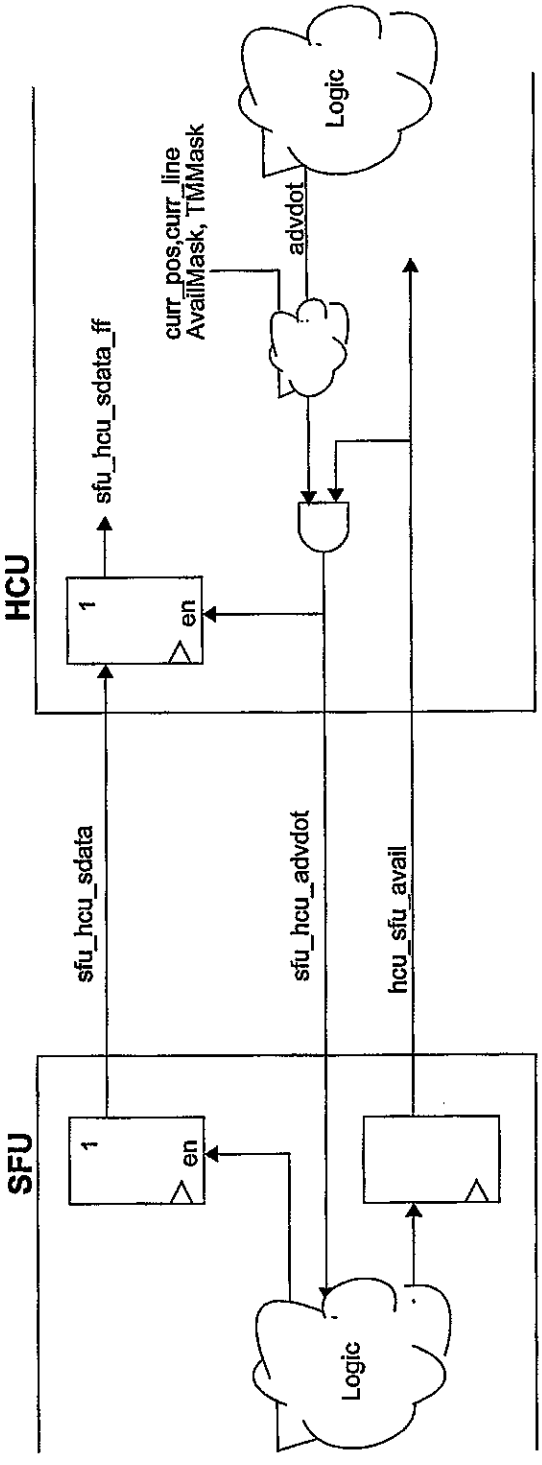

HCU
SFU
Logic
curr_pos,curr_line
AvailMask, TMMask
advdot
sfu_hcu_sdata_ff
1
en
sfu_hcu_sdata
sfu_hcu_advdot
hcu_sfu_avail
Logic

FIG. 243

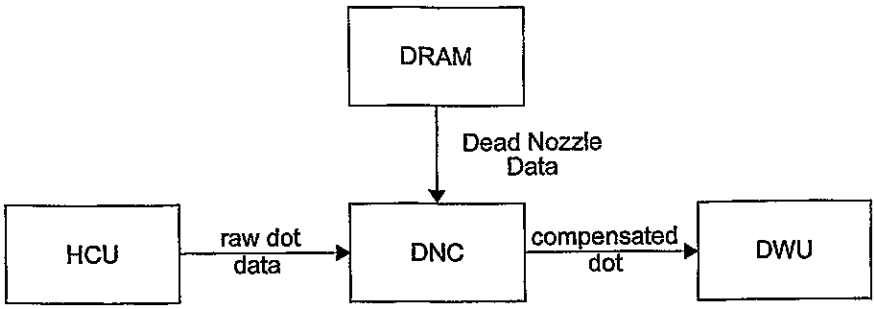


FIG. 244

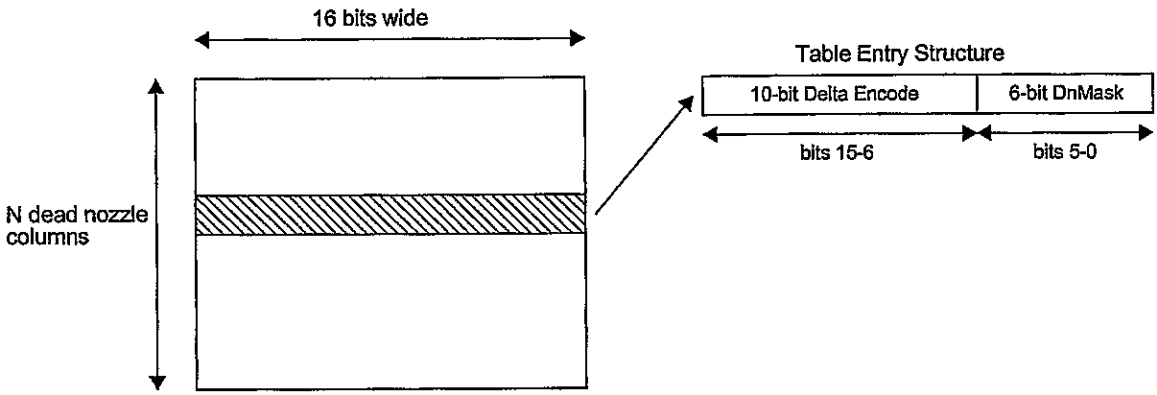


FIG. 245

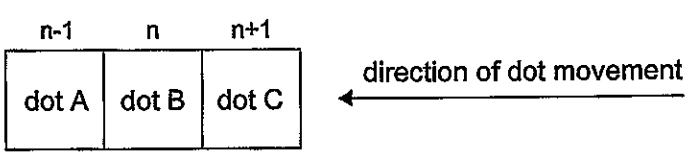


FIG. 246

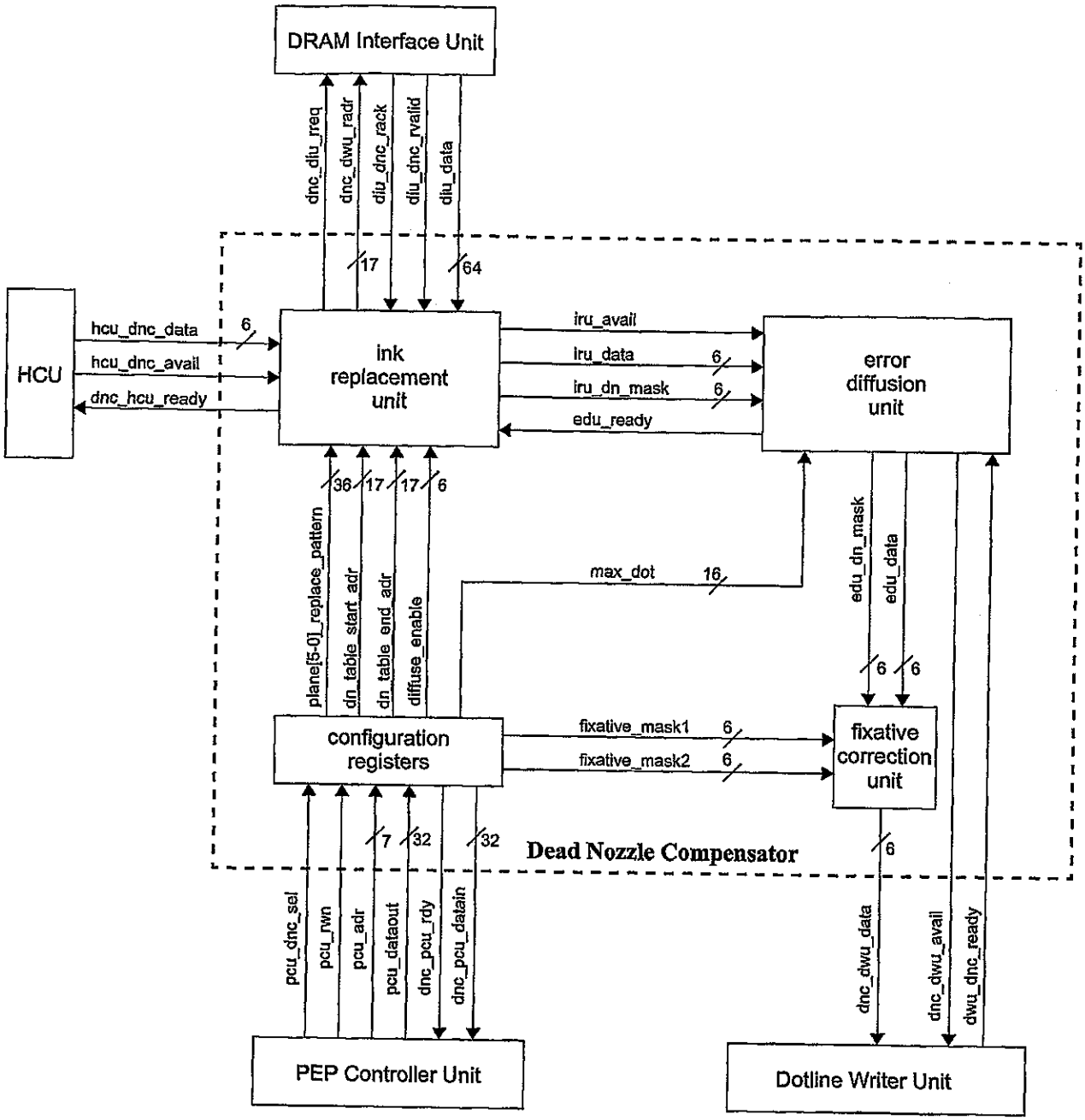

DRAM Interface Unit
dnc_diu_rreq
dnc_dwu_radr
diu_dnc_rack
diu_dnc_rvalid
diu_data
17
64
HCU
hcu_dnc_data
hcu_dnc_avail
dnc_hcu_ready
6
ink replacement unit
iru_avail
iru_data
iru_dn_mask
edu_ready
6
6
error diffusion unit
36
17
17
6
plane[5-0]_replace_pattern
dn_table_start_adr
dn_table_end_adr
diffuse_enable
max_dot
16
edu_dn_mask
edu_data
6
6
configuration registers
fixative_mask1
fixative_mask2
6
6
fixative correction unit
7
32
32
Dead Nozzle Compensator
6
pcu_dnc_sel
pcu_rwn
pcu_adr
pcu_dataout
dnc_pcu_rdy
dnc_pcu_datain
dnc_dwu_data
dnc_dwu_avail
dwu_dnc_ready
PEP Controller Unit
Dotline Writer Unit

FIG. 247

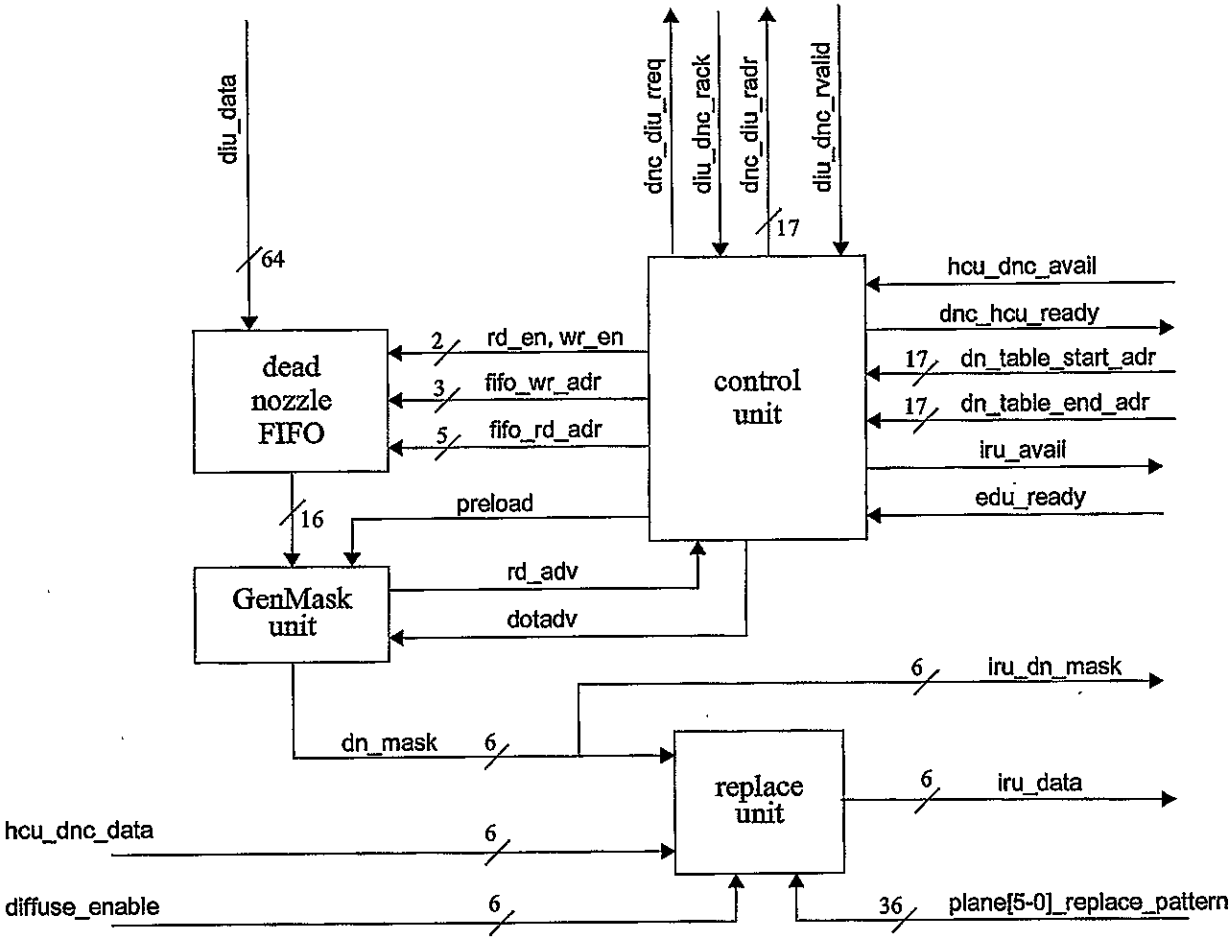

diu_data
64
dnc_diu_rreq
diu_dnc_rack
dnc_diu_radr
diu_dnc_rvalid
17
hcu_dnc_avail
dnc_hcu_ready
17 dn_table_start_adr
17 dn_table_end_adr
iru_avail
edu_ready
dead nozzle FIFO
2 rd_en, wr_en
3 fifo_wr_adr
5 fifo_rd_adr
control unit
16
preload
GenMask unit
rd_adv
dotadv
6 iru_dn_mask
dn_mask 6
replace unit
6 iru_data
hcu_dnc_data
6
diffuse_enable
6
36 plane[5-0]_replace_pattern

FIG. 248

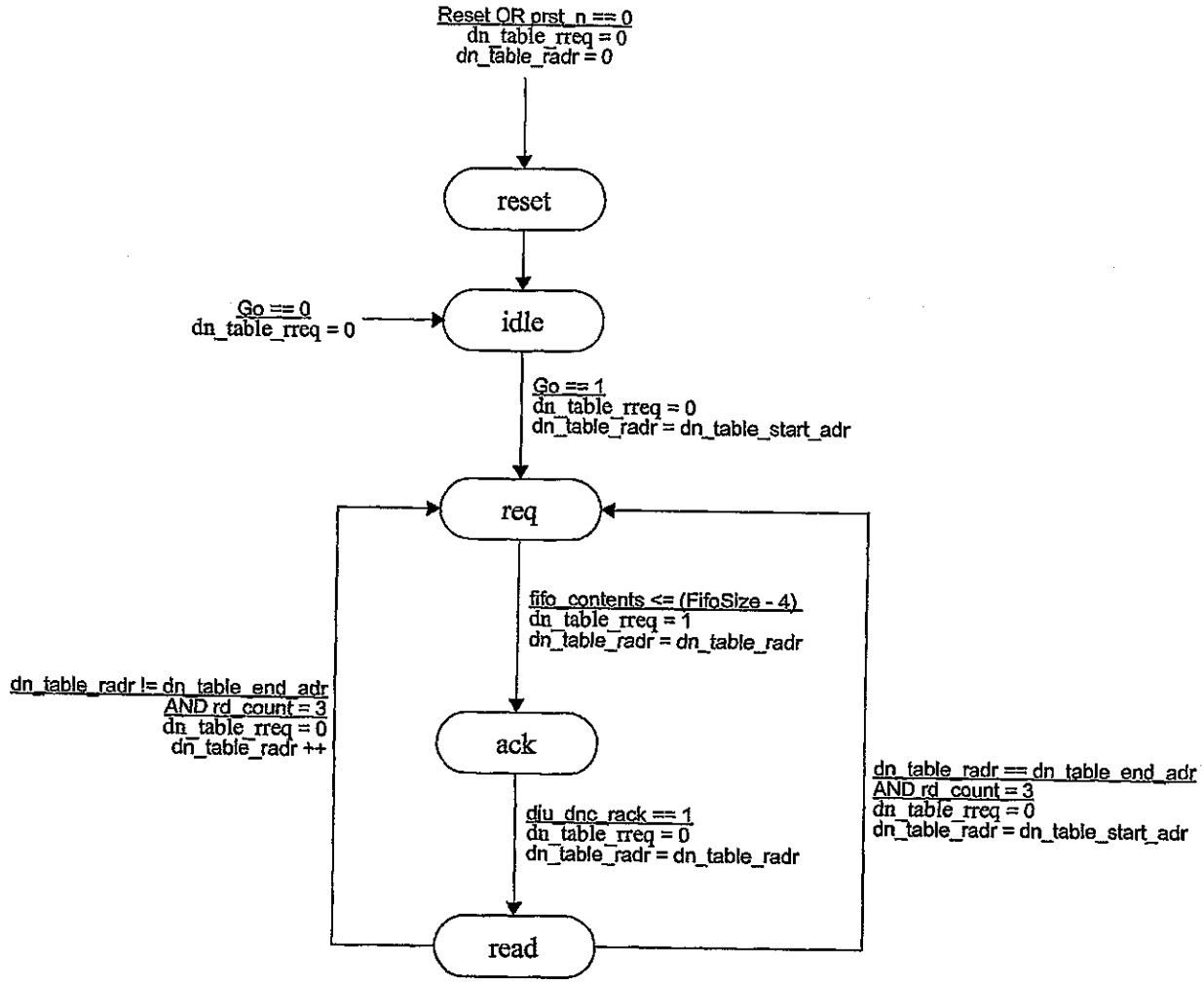

Reset OR prst_n == 0
dn_table_rreq = 0
dn_table_radr = 0
reset
Go == 0
dn_table_rreq = 0
idle
Go == 1
dn_table_rreq = 0
dn_table_radr = dn_table_start_adr
req
fifo_contents <= (FifoSize - 4)
dn_table_rreq = 1
dn_table_radr = dn_table_radr
dn_table_radr != dn_table_end_adr
AND rd_count = 3
dn_table_rreq = 0
dn_table_radr ++
ack
dn_table_radr == dn_table_end_adr
AND rd_count = 3
dn_table_rreq = 0
dn_table_radr = dn_table_start_adr
diu_dnc_rack == 1
dn_table_rreq = 0
dn_table_radr = dn_table_radr
read

FIG. 249

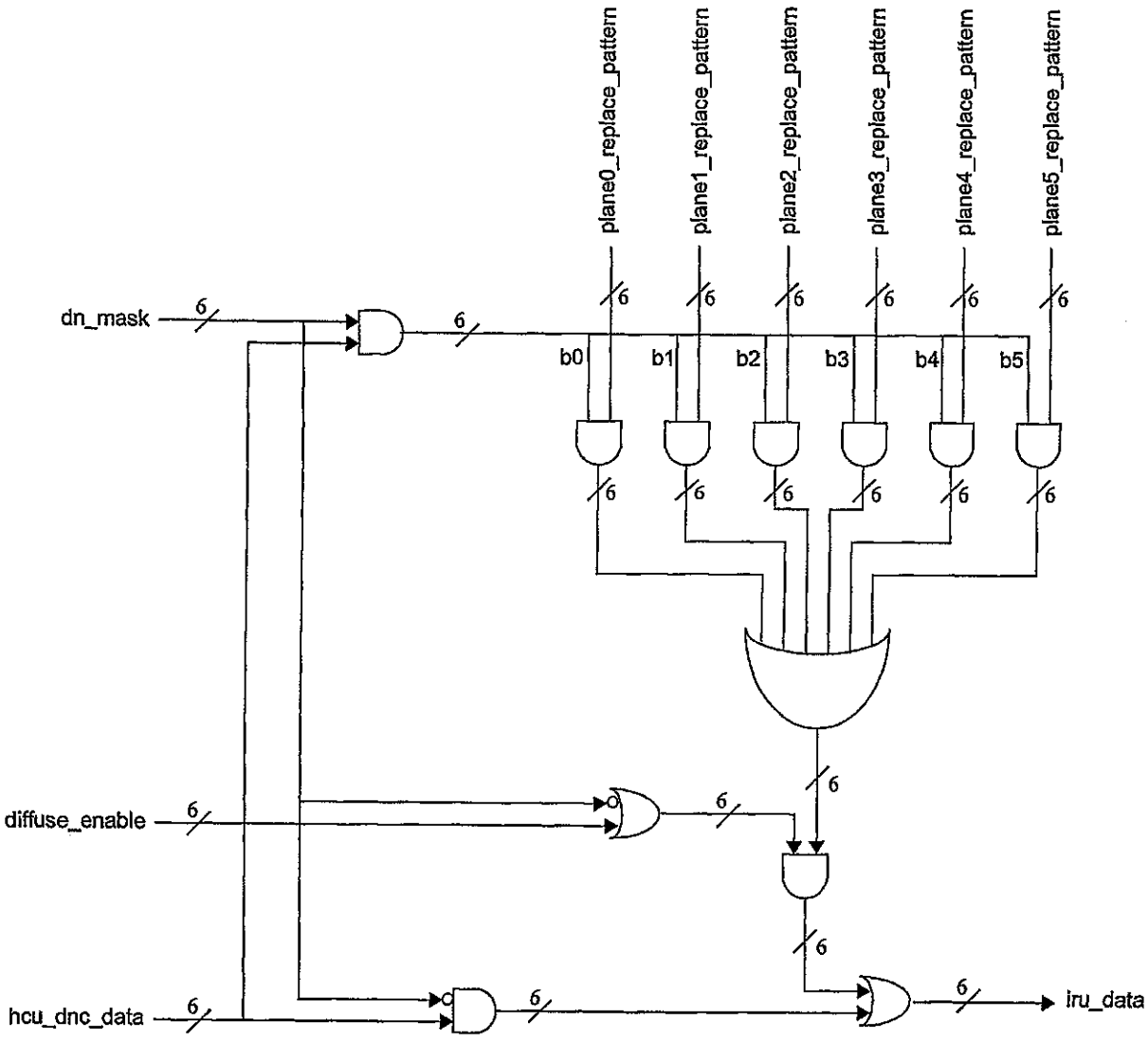

plane0_replace_pattern
plane1_replace_pattern
plane2_replace_pattern
plane3_replace_pattern
plane4_replace_pattern
plane5_replace_pattern
dn_mask
b0
b1
b2
b3
b4
b5
diffuse_enable
hcu_dnc_data
iru_data
6

FIG. 250

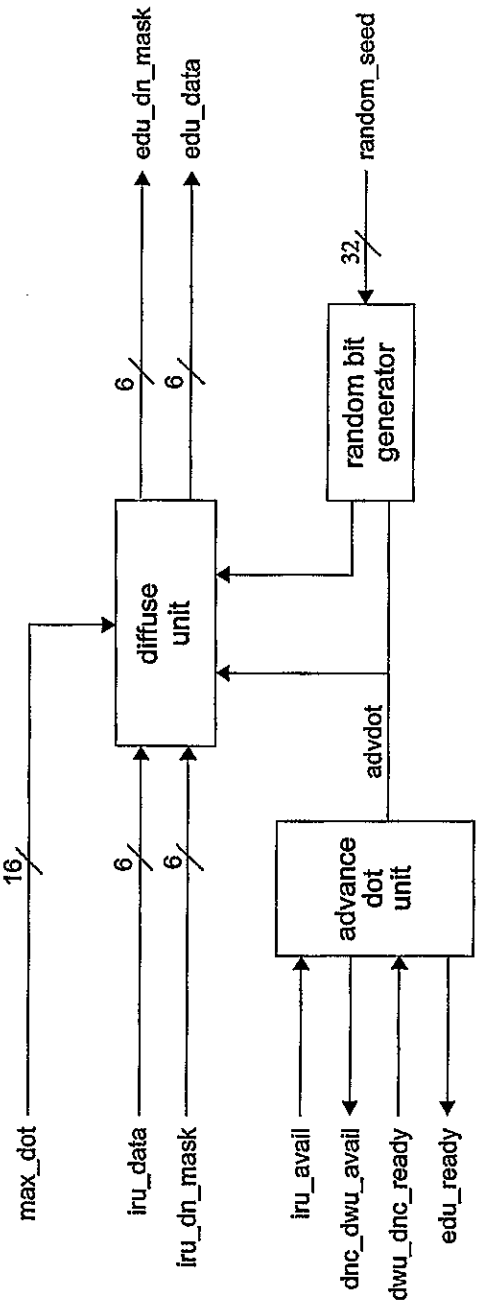

edu_dn_mask
edu_data
random_seed
32
6
6
random bit generator
diffuse unit
advdot
16
6
6
advance dot unit
max_dot
iru_data
iru_dn_mask
iru_avail
dnc_dwu_avail
dwu_dnc_ready
edu_ready

FIG. 251

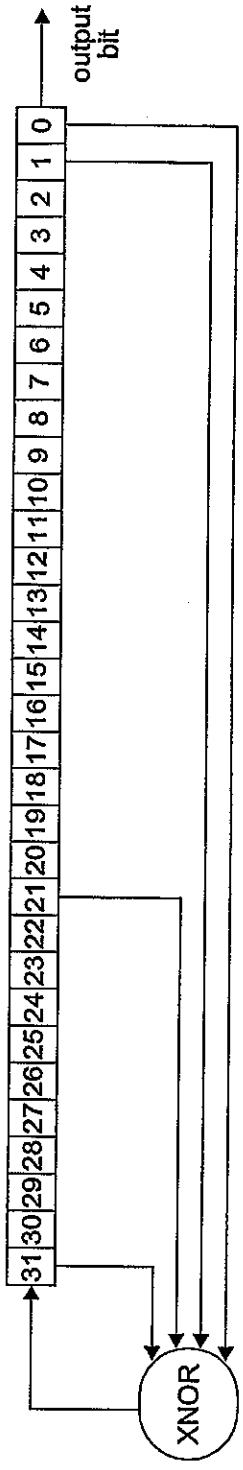


FIG. 252

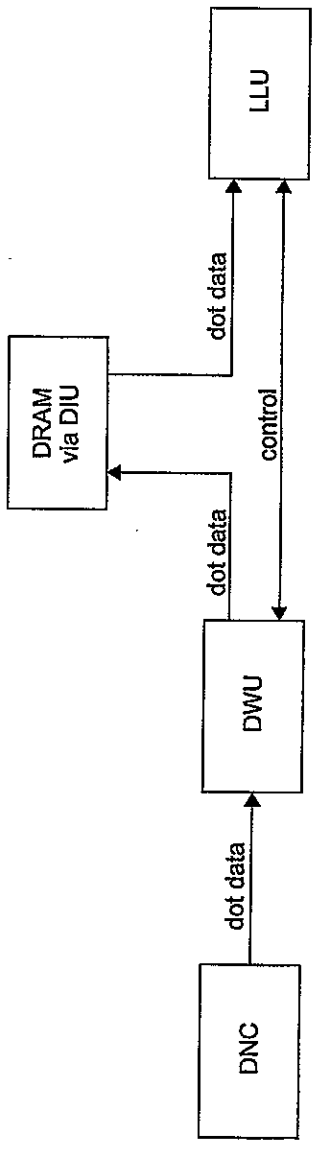


FIG. 253

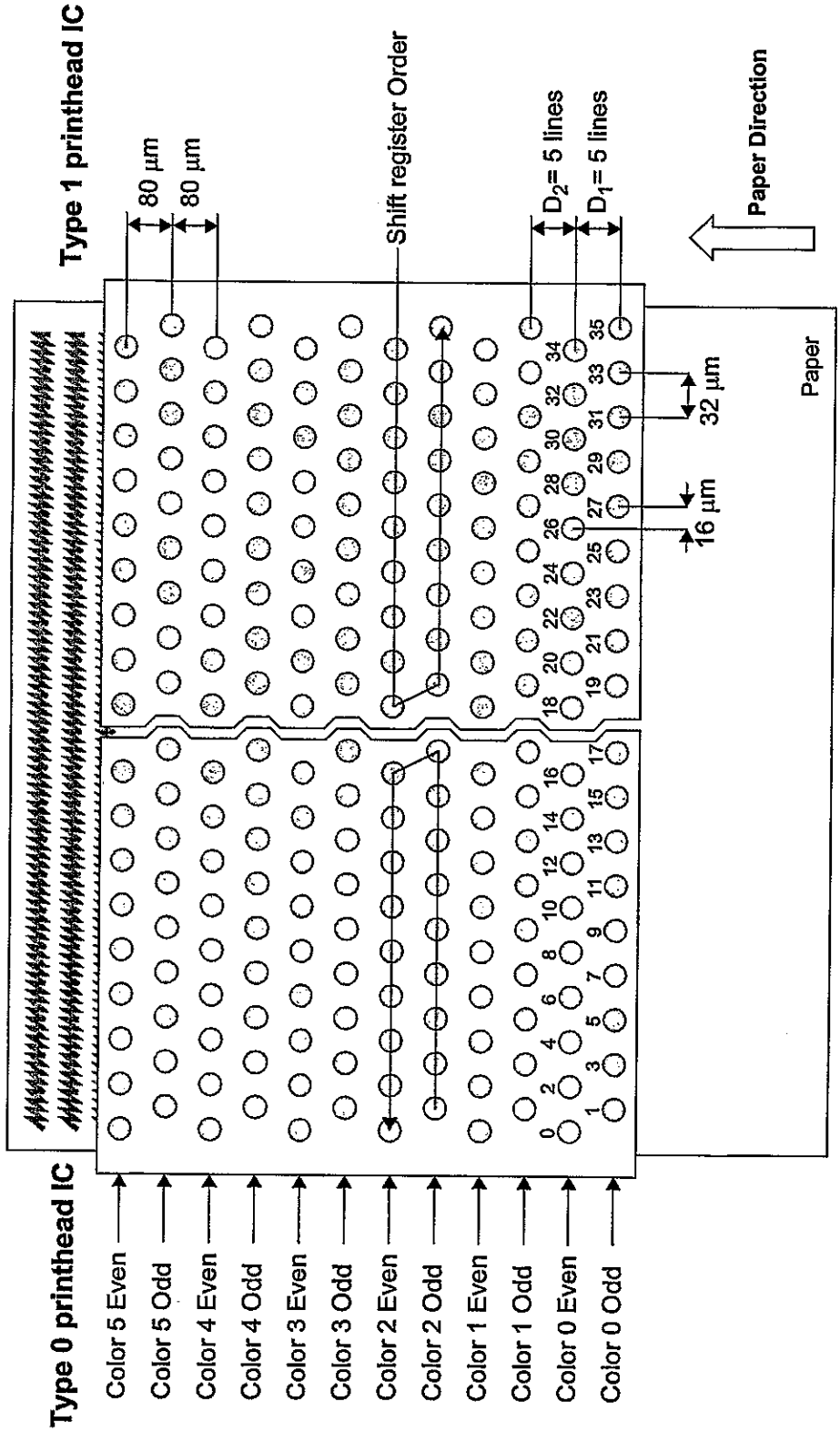

Type 0 printhead IC
Type 1 printhead IC
Color 5 Even
Color 5 Odd
Color 4 Even
Color 4 Odd
Color 3 Even
Color 3 Odd
Color 2 Even
Color 2 Odd
Color 1 Even
Color 1 Odd
Color 0 Even
Color 0 Odd
80 µm
80 µm
Shift register Order
D_2= 5 lines
D_1= 5 lines
32 µm
16 µm
Paper Direction
Paper
Note: Paper passes under printhead

FIG. 254

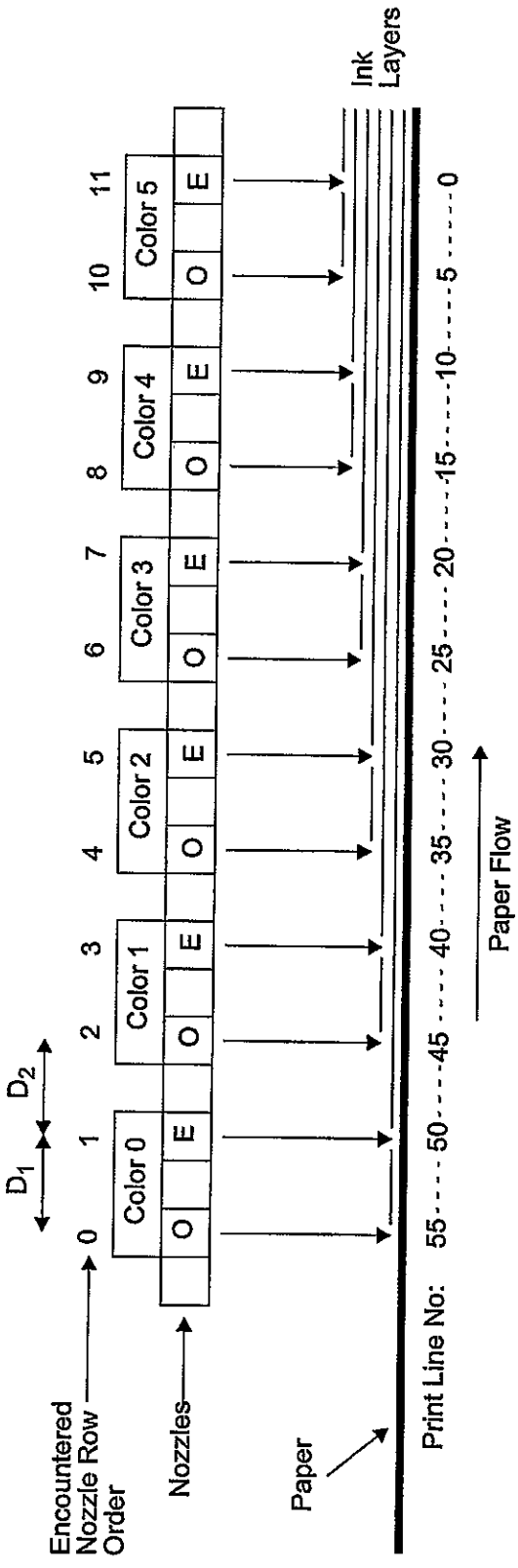

Ink Layers
Color 5
Color 4
Color 3
Color 2
Color 1
Color 0
O
E
D1
D2
0 1 2 3 4 5 6 7 8 9 10 11
Encountered Nozzle Row Order
Nozzles
Paper
Print Line No: 55 50 45 40 35 30 25 20 15 10 5 0
Paper Flow

FIG. 255

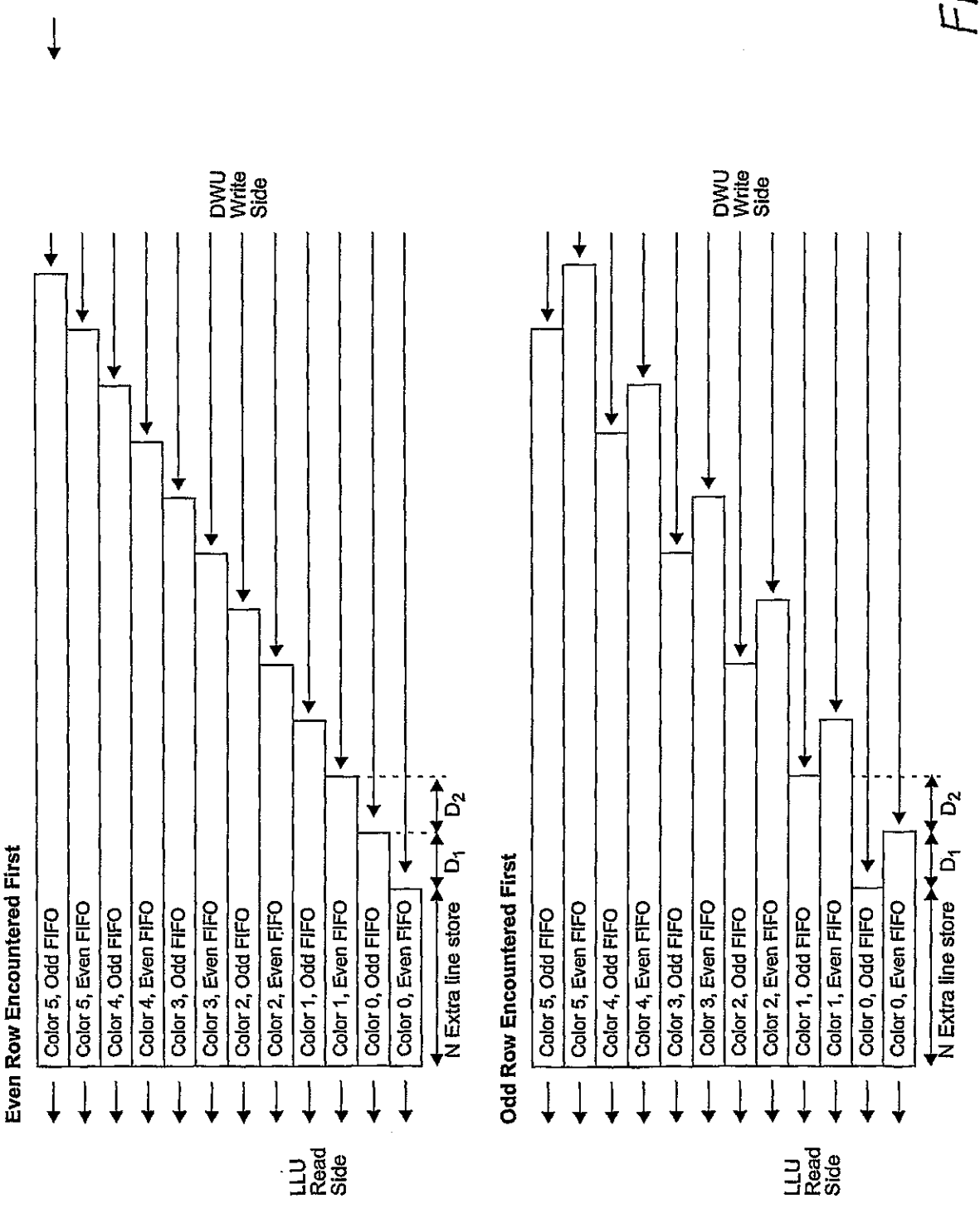

Even Row Encountered First
Color 5, Odd FIFO
Color 5, Even FIFO
Color 4, Odd FIFO
Color 4, Even FIFO
Color 3, Odd FIFO
Color 3, Even FIFO
Color 2, Odd FIFO
Color 2, Even FIFO
Color 1, Odd FIFO
Color 1, Even FIFO
Color 0, Odd FIFO
Color 0, Even FIFO
N Extra line store
D1
D2
DWU Write Side
LLU Read Side
Odd Row Encountered First
Color 5, Odd FIFO
Color 5, Even FIFO
Color 4, Odd FIFO
Color 4, Even FIFO
Color 3, Odd FIFO
Color 3, Even FIFO
Color 2, Odd FIFO
Color 2, Even FIFO
Color 1, Odd FIFO
Color 1, Even FIFO
Color 0, Odd FIFO
Color 0, Even FIFO
N Extra line store
D1
D2
DWU Write Side
LLU Read Side

FIG. 256

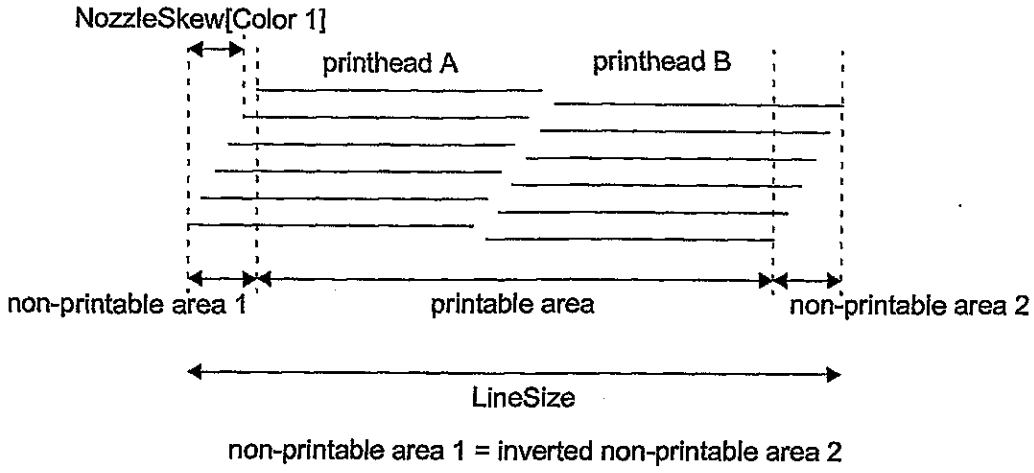


FIG. 257

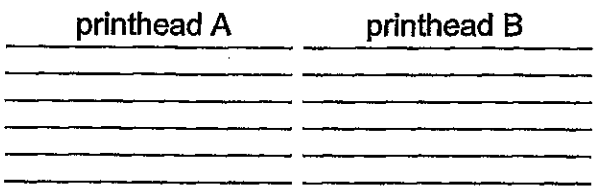


FIG. 258

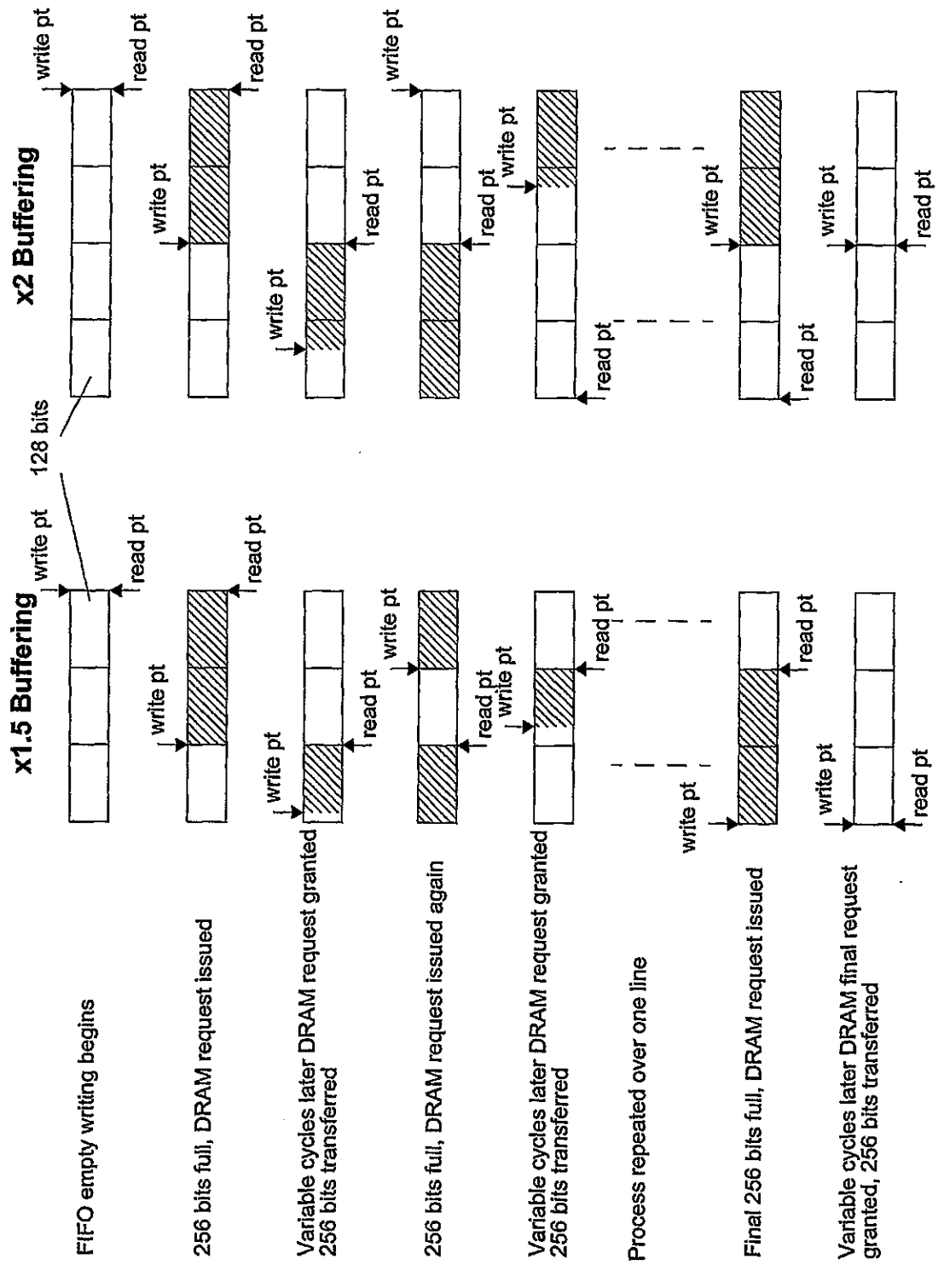

x1.5 Buffering
x2 Buffering
128 bits
write pt
read pt
FIFO empty writing begins
256 bits full, DRAM request issued
Variable cycles later DRAM request granted
256 bits transferred
256 bits full, DRAM request issued again
Variable cycles later DRAM request granted
256 bits transferred
Process repeated over one line
Final 256 bits full, DRAM request issued
Variable cycles later DRAM final request
granted, 256 bits transferred

FIG. 259

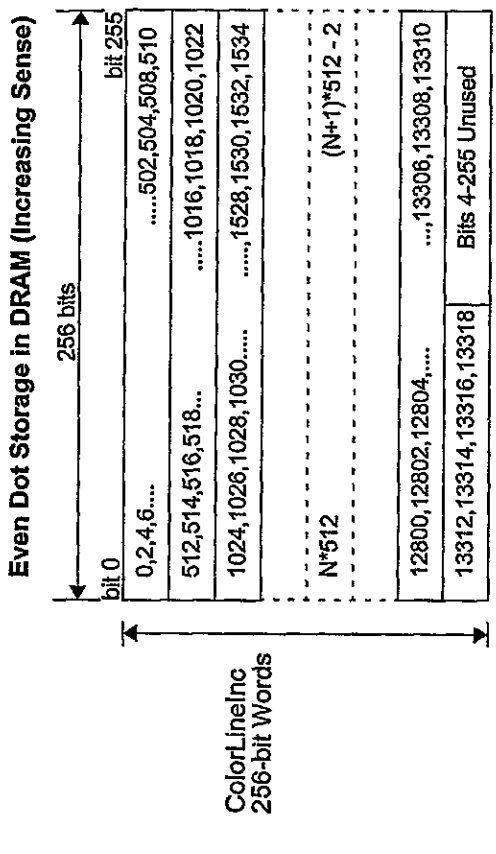


FIG. 260

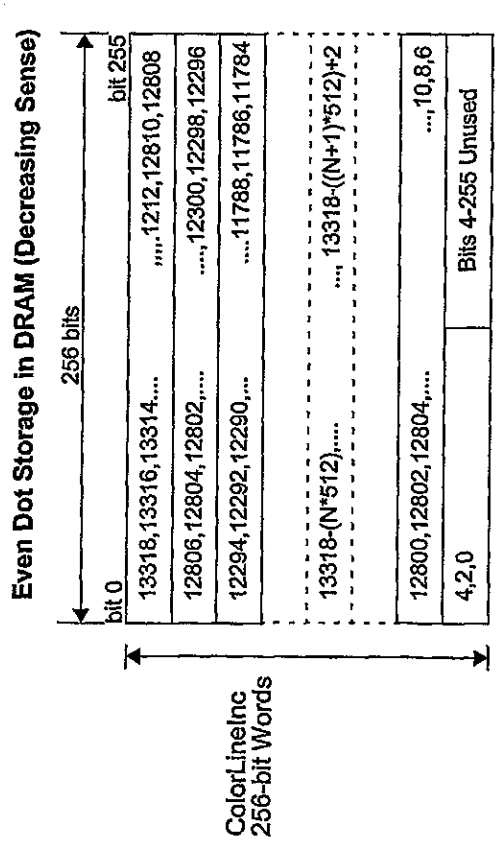


FIG. 261

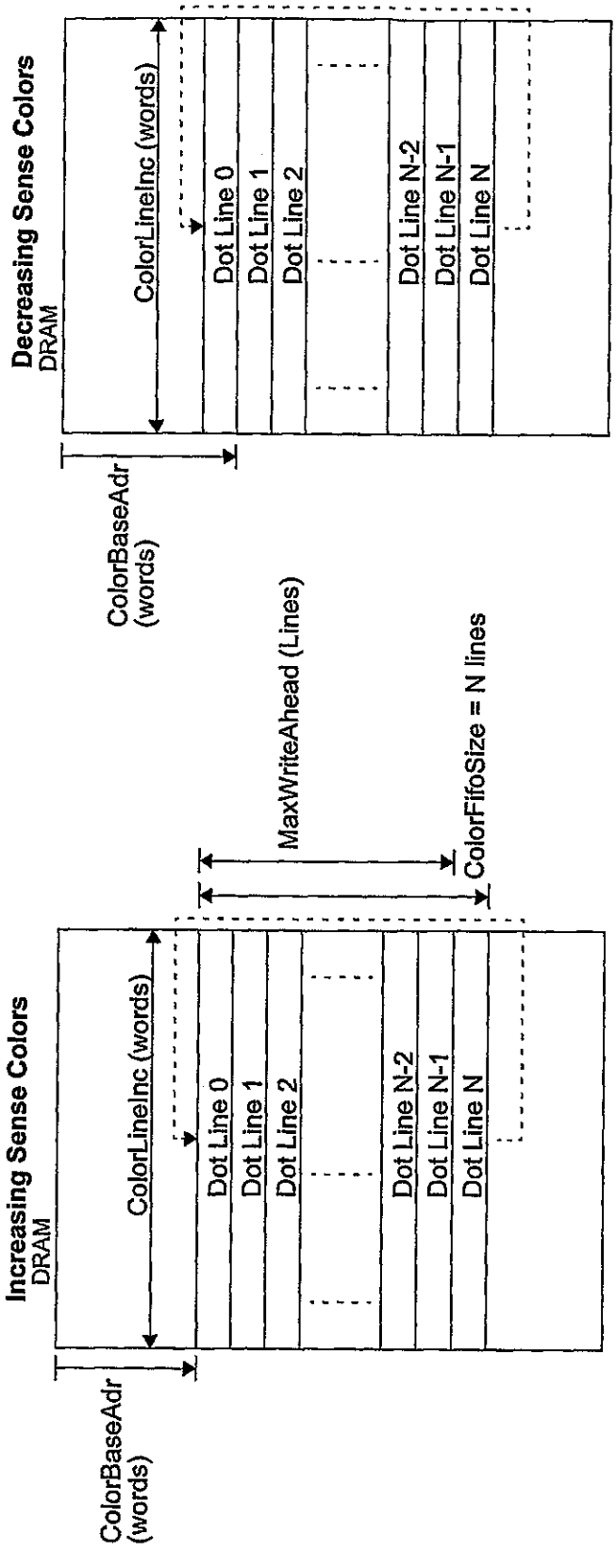

Decreasing Sense Colors
DRAM
ColorLineInc (words)
Dot Line 0
Dot Line 1
Dot Line 2
Dot Line N-2
Dot Line N-1
Dot Line N
ColorBaseAdr (words)
MaxWriteAhead (Lines)
ColorFifoSize = N lines
Increasing Sense Colors
DRAM
ColorLineInc (words)
Dot Line 0
Dot Line 1
Dot Line 2
Dot Line N-2
Dot Line N-1
Dot Line N
ColorBaseAdr (words)

FIG. 262

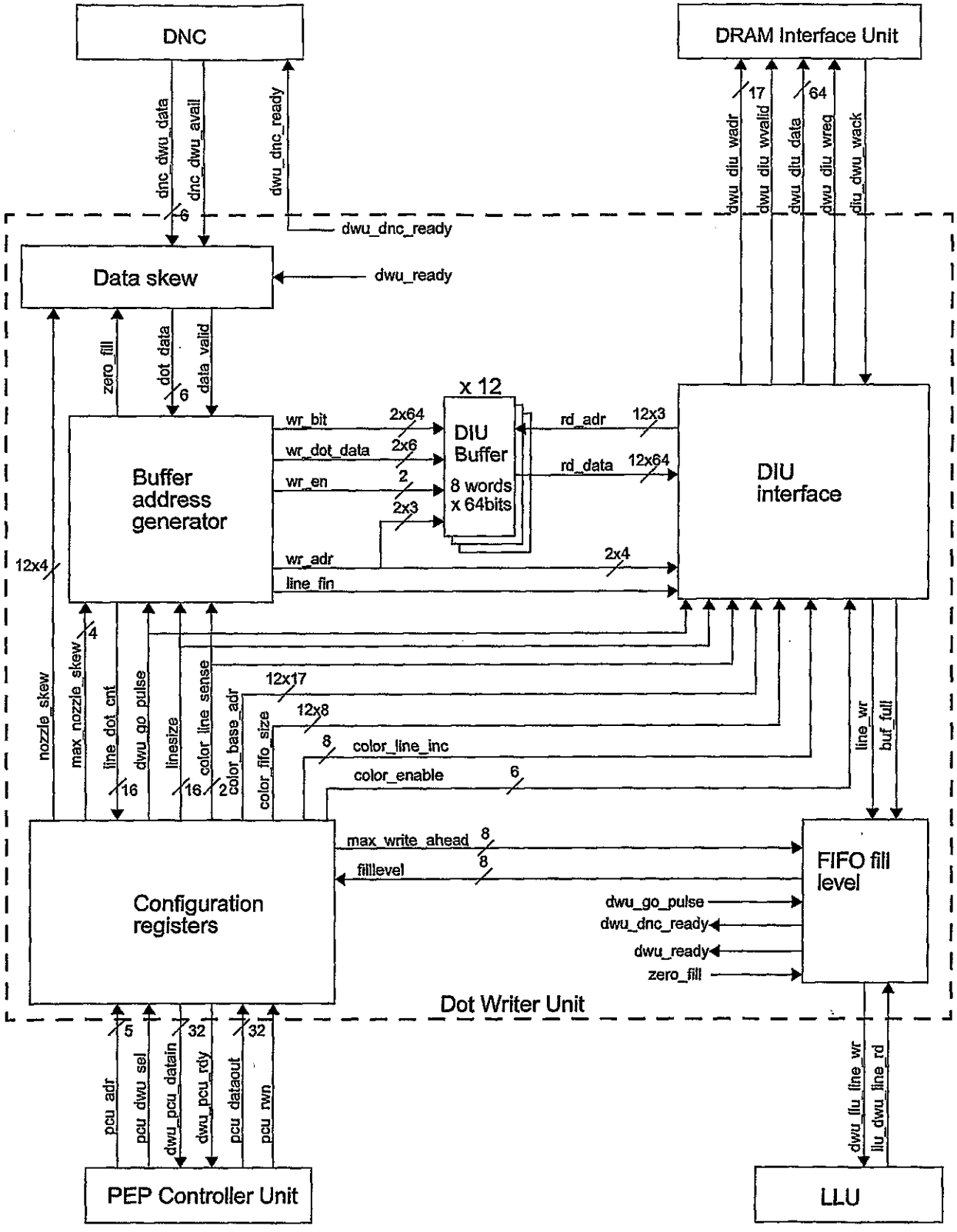

DNC
DRAM Interface Unit
dnc_dwu_data
dnc_dwu_avail
dwu_dnc_ready
dwu_diu_wadr
dwu_diu_wvalid
dwu_diu_data
dwu_diu_wreq
diu_dwu_wack
17
64
6
dwu_dnc_ready
Data skew
dwu_ready
zero_fill
dot_data
data_valid
x 12
wr_bit
wr_dot_data
wr_en
2x64
2x6
2
2x3
DIU Buffer
8 words x 64bits
rd_adr
rd_data
12x3
12x64
DIU interface
Buffer address generator
12x4
wr_adr
line_fin
2x4
nozzle_skew
max_nozzle_skew
line_dot_cnt
dwu_go_pulse
linesize
color_line_sense
color_base_adr
color_fifo_size
12x17
12x8
8
color_line_inc
color_enable
6
line_wr
buf_full
max_write_ahead
filllevel
8
FIFO fill level
Configuration registers
dwu_go_pulse
dwu_dnc_ready
dwu_ready
zero_fill
Dot Writer Unit
pcu_adr
pcu_dwu_sel
dwu_pcu_datain
dwu_pcu_rdy
pcu_dataout
pcu_rwn
5
32
32
dwu_llu_line_wr
llu_dwu_line_rd
PEP Controller Unit
LLU

FIG. 263

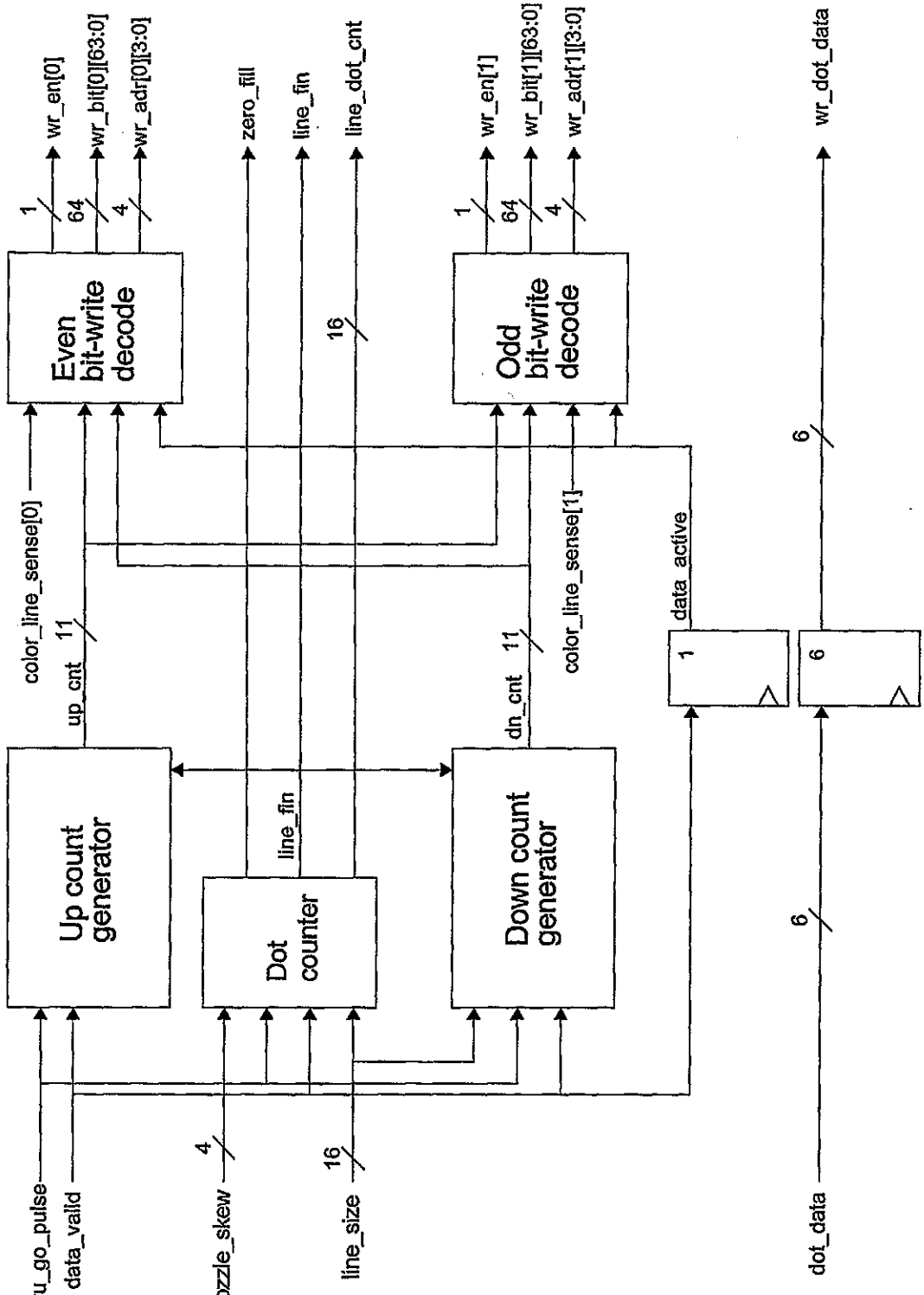

Even bit-write decode
Odd bit-write decode
Up count generator
Down count generator
Dot counter
wr_en[0]
wr_bit[0][63:0]
wr_adr[0][3:0]
wr_en[1]
wr_bit[1][63:0]
wr_adr[1][3:0]
zero_fill
line_fin
line_dot_cnt
color_line_sense[0]
color_line_sense[1]
up_cnt
dn_cnt
data_active
wr_dot_data
dot_data
dwu_go_pulse
data_valid
max_nozzle_skew
line_size
1
64
4
16
11
6

FIG. 264

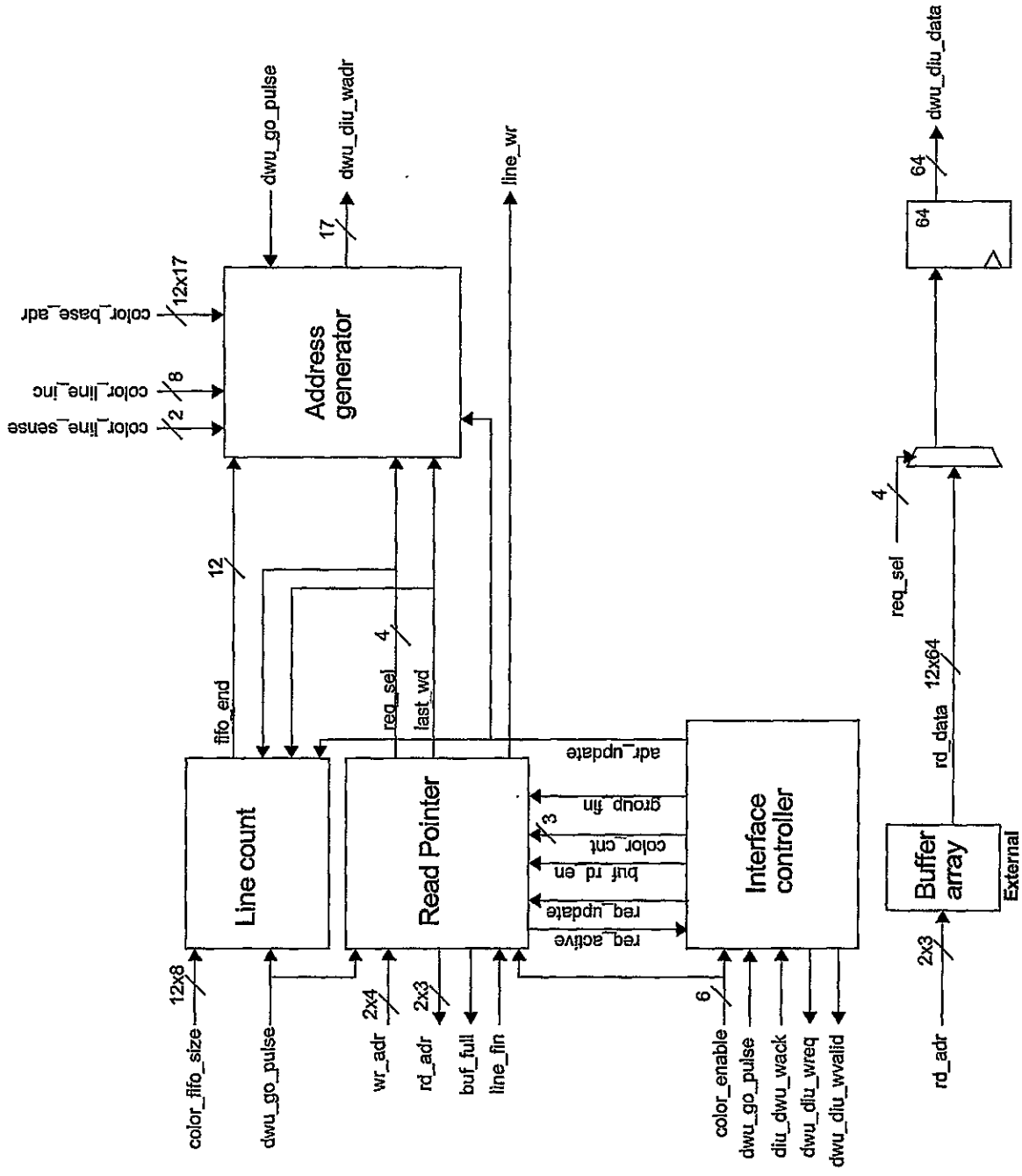

color_base_adr
12x17
color_line_inc
8
color_line_sense
2
Address generator
dwu_go_pulse
dwu_diu_wadr
17
line_wr
12
fifo_end
req_sel
4
last_wd
Line count
Read Pointer
adr_update
group_fin
3
color_cnt
buf_rd_en
req_update
req_active
Interface controller
12x8
color_fifo_size
dwu_go_pulse
2x4
wr_adr
2x3
rd_adr
buf_full
line_fin
6
color_enable
dwu_go_pulse
diu_dwu_wack
dwu_diu_wreq
dwu_diu_wvalid
Buffer array
External
rd_data
12x64
req_sel
4
64
64
dwu_diu_data
2x3
rd_adr

FIG. 265

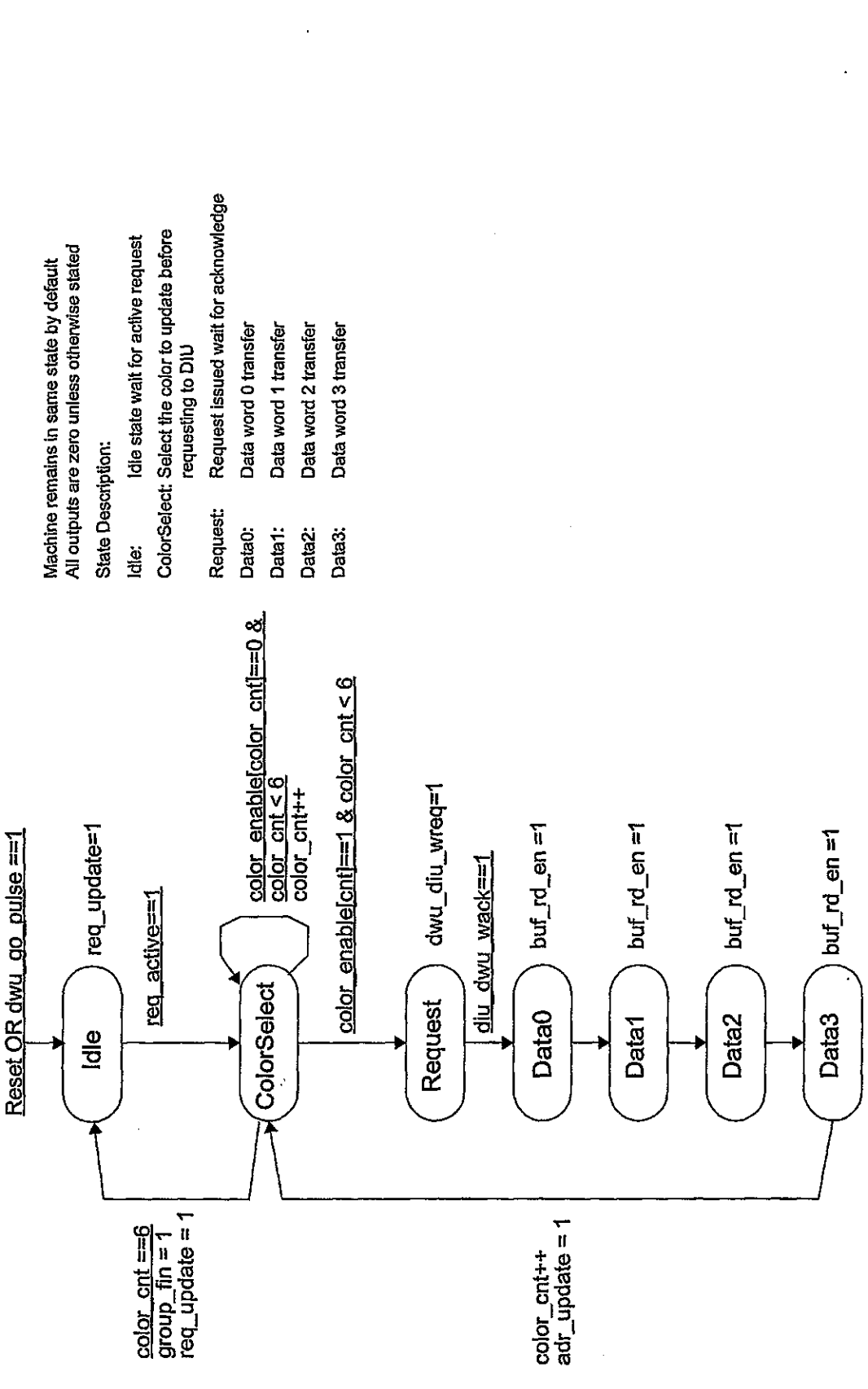

Machine remains in same state by default
All outputs are zero unless otherwise stated
State Description:
Idle: Idle state wait for active request
ColorSelect: Select the color to update before requesting to DIU
Request: Request issued wait for acknowledge
Data0: Data word 0 transfer
Data1: Data word 1 transfer
Data2: Data word 2 transfer
Data3: Data word 3 transfer
Reset OR dwu_go_pulse ==1
Idle
req_update=1
req_active==1
ColorSelect
color_enable[color_cnt[=0 &
color_cnt < 6
color_cnt++
color_enable[cnt]==1 & color_cnt < 6
Request
dwu_diu_wreq=1
diu_dwu_wack==1
Data0
buf_rd_en =1
Data1
buf_rd_en =1
Data2
buf_rd_en =1
Data3
buf_rd_en =1
color_cnt ==6
group_fin = 1
req_update = 1
color_cnt++
adr_update = 1

FIG. 266

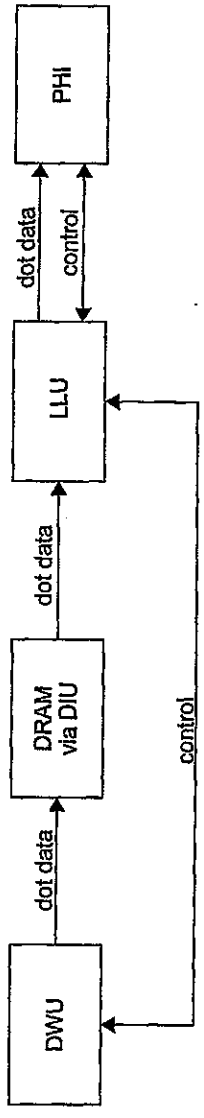


FIG. 267

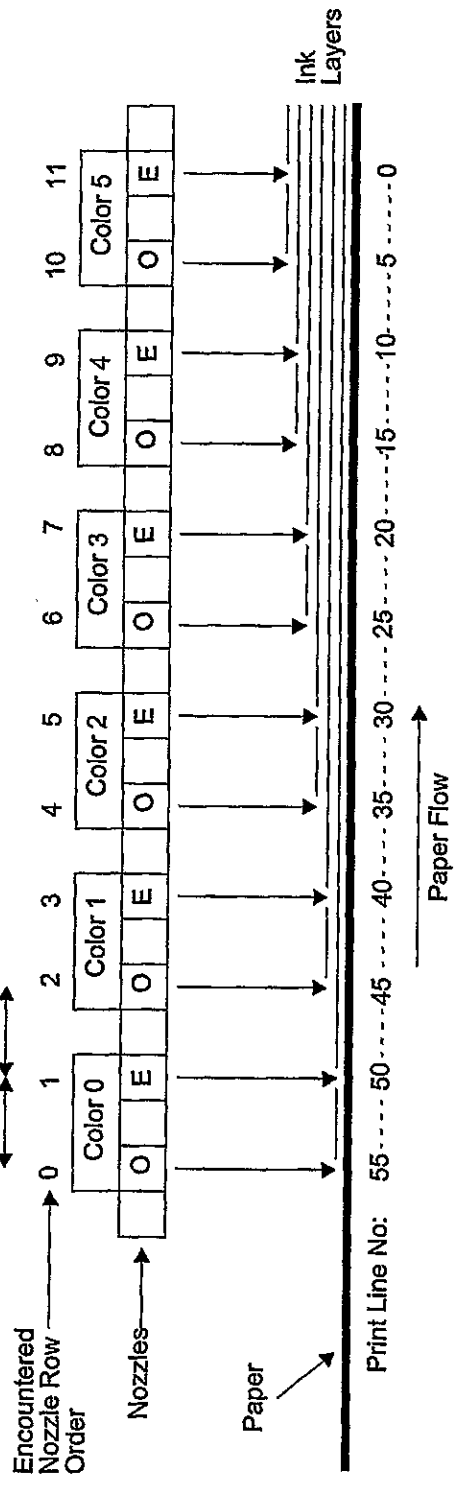


FIG. 268

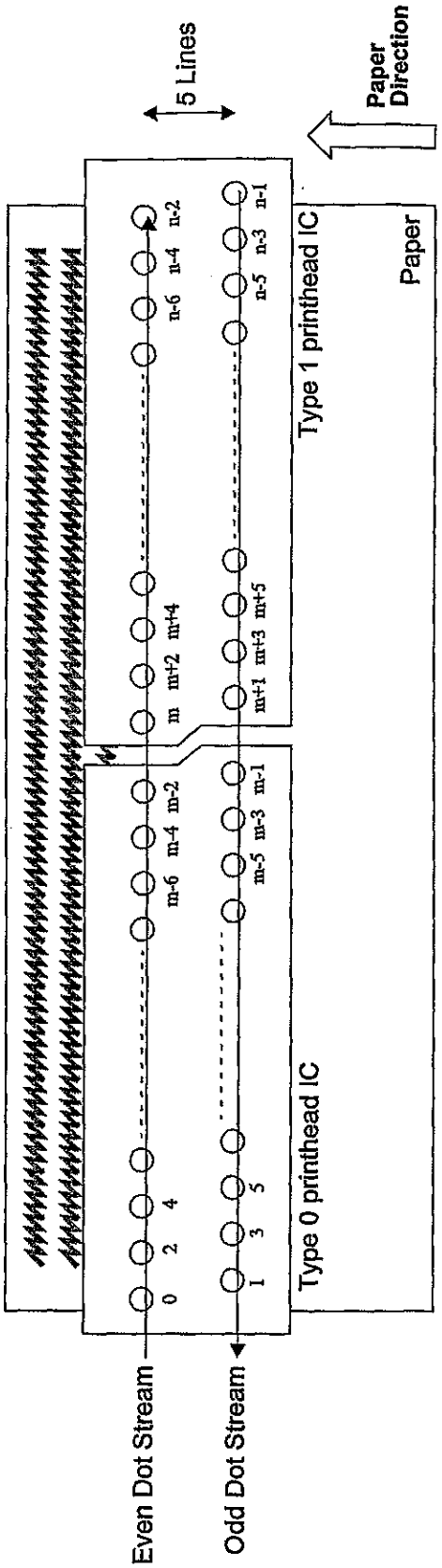

5 Lines
Paper Direction
Type 1 printhead IC
Paper
Type 0 printhead IC
Even Dot Stream
Odd Dot Stream
0 2 4
1 3 5
m-6 m-4 m-2
m-5 m-3 m-1
m m+2 m+4
m+1 m+3 m+5
n-6 n-4 n-2
n-5 n-3 n-1
M - Midway point in dots
N - Number of dots in a line
Note: Paper passing under printhead

FIG. 269

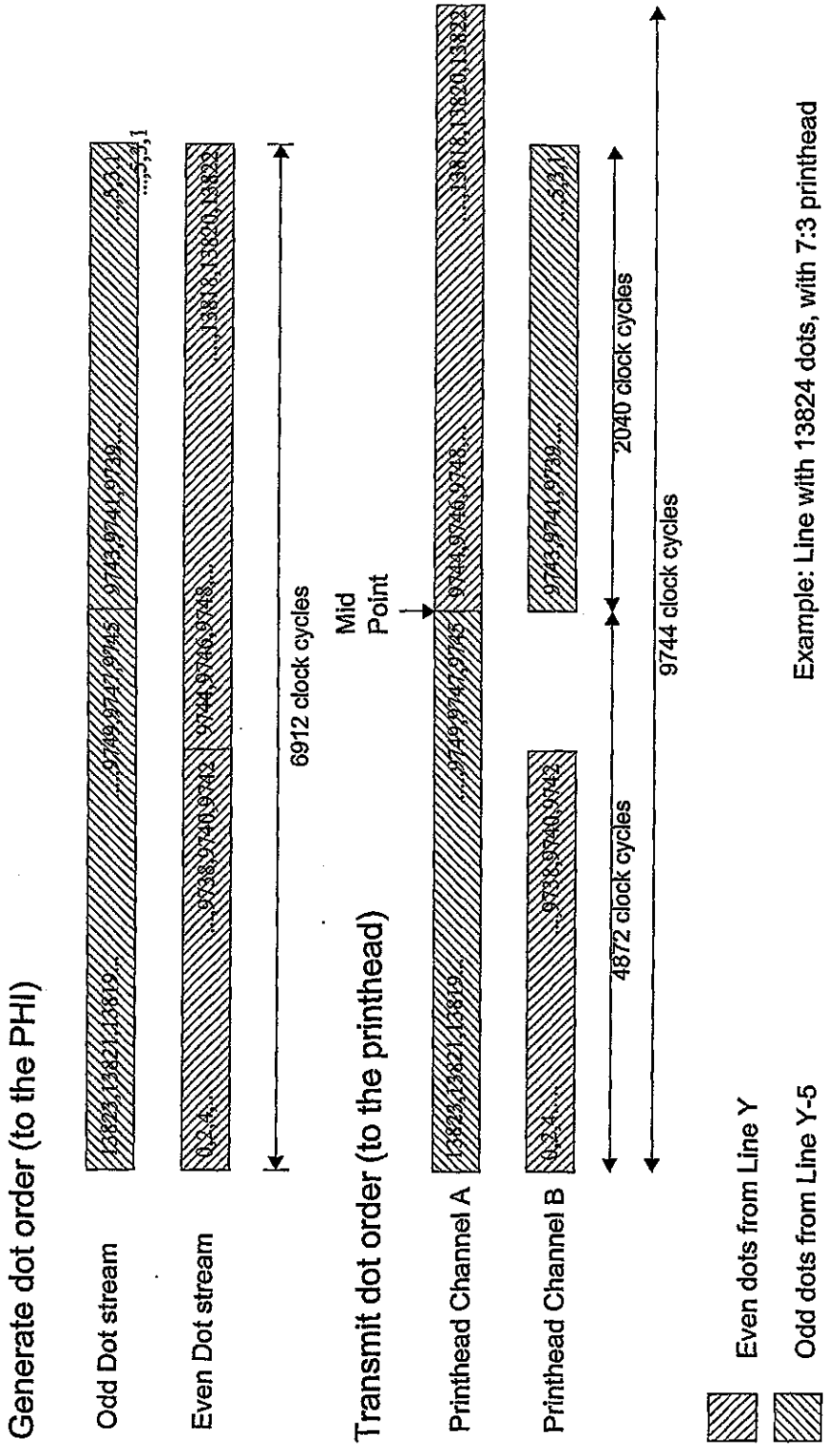

Generate dot order (to the PHI)
Odd Dot stream
Even Dot stream
6912 clock cycles
Transmit dot order (to the printhead)
Mid Point
Printhead Channel A
Printhead Channel B
4872 clock cycles
2040 clock cycles
9744 clock cycles
Even dots from Line Y
Odd dots from Line Y-5
Example: Line with 13824 dots, with 7:3 printhead

FIG. 270

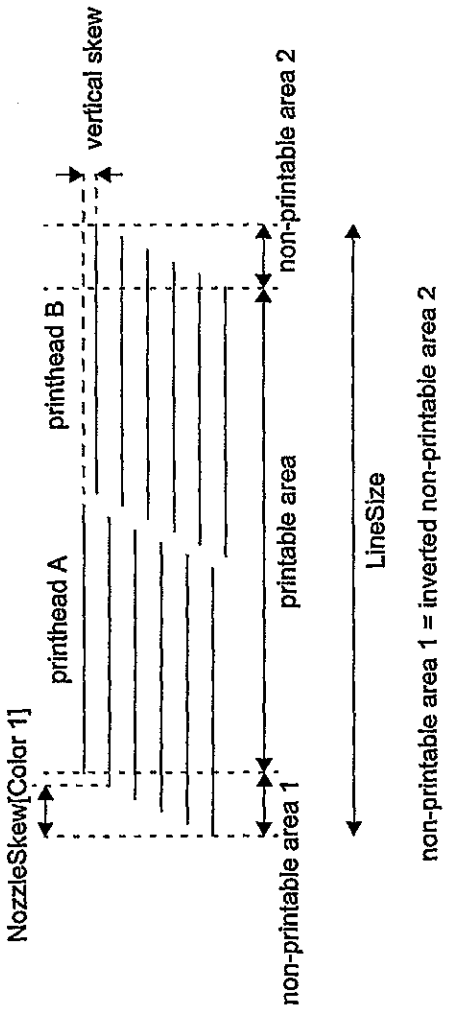

NozzleSkew[Color 1]
printhead A
printhead B
verfical skew
non-printable area 1
printable area
non-printable area 2
LineSize
non-printable area 1 = inverted non-printable area 2

FIG. 271

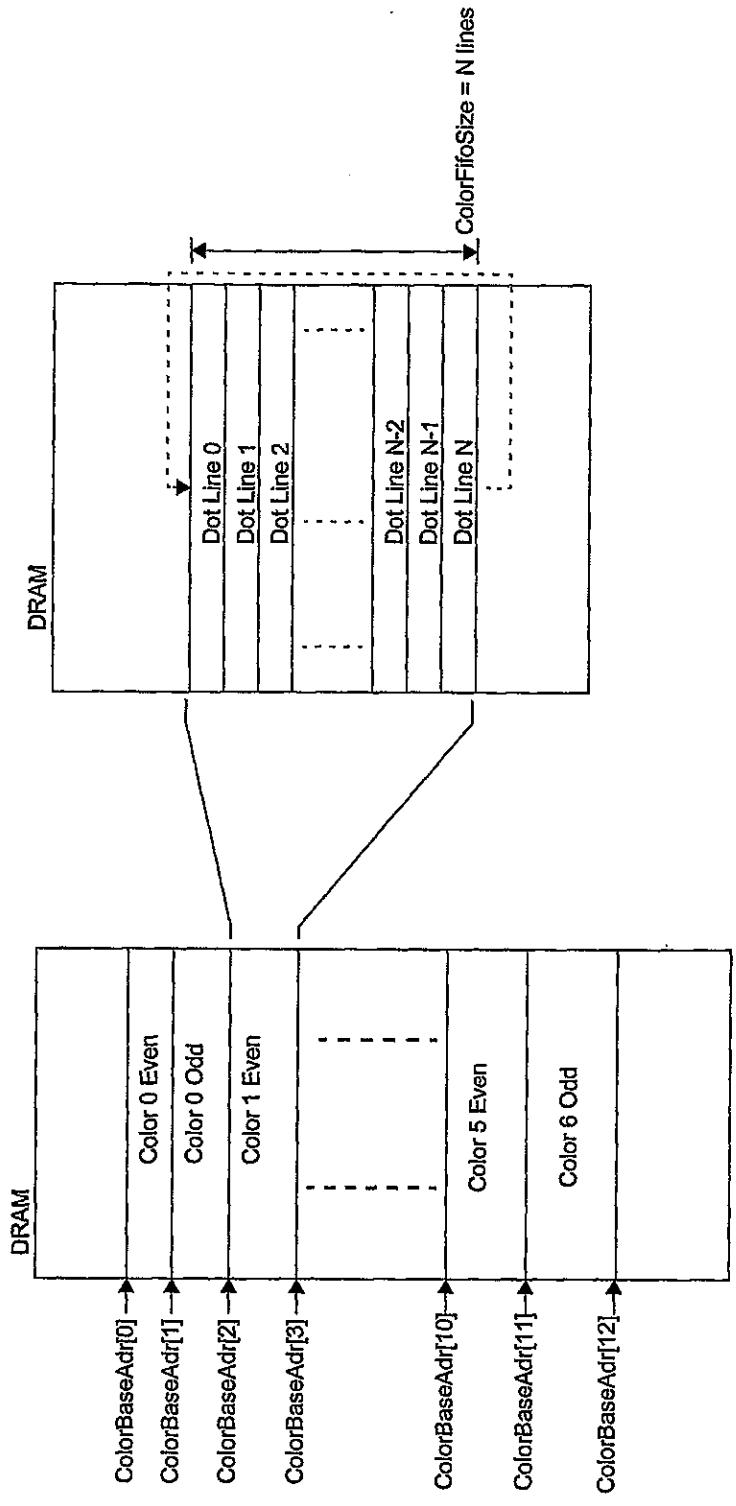

DRAM
Color 0 Even
Color 0 Odd
Color 1 Even
Color 5 Even
Color 6 Odd
ColorBaseAdr[0]
ColorBaseAdr[1]
ColorBaseAdr[2]
ColorBaseAdr[3]
ColorBaseAdr[10]
ColorBaseAdr[11]
ColorBaseAdr[12]
DRAM
Dot Line 0
Dot Line 1
Dot Line 2
Dot Line N-2
Dot Line N-1
Dot Line N
ColorFifoSize = N lines

FIG. 272

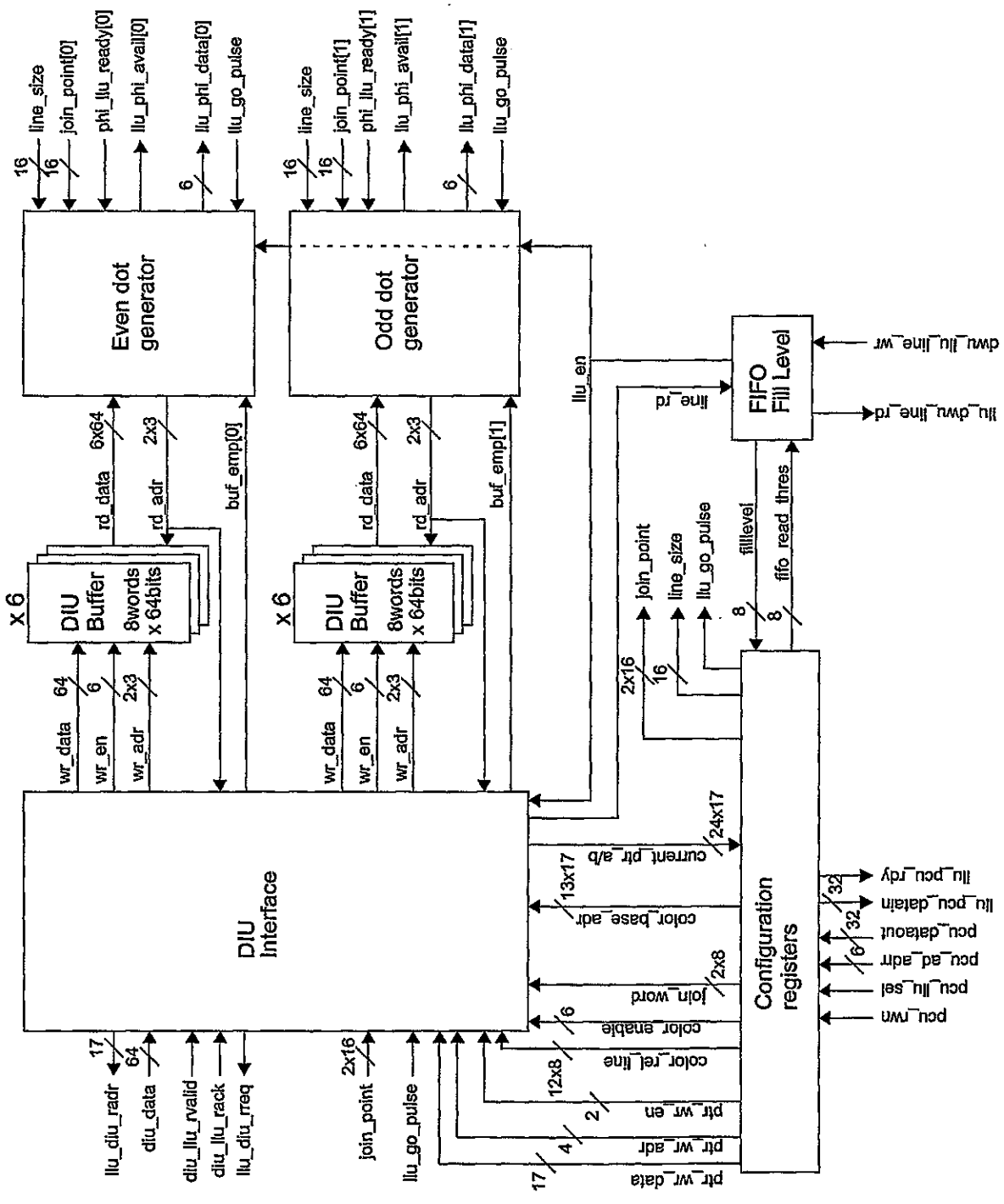

Even dot generator
Odd dot generator
DIU Buffer
8words x 64bits
x 6
DIU Interface
Configuration registers
FIFO Fill Level
line_size
join_point[0]
phi_llu_ready[0]
llu_phi_avail[0]
llu_phi_data[0]
llu_go_pulse
join_point[1]
phi_llu_ready[1]
llu_phi_avail[1]
llu_phi_data[1]
rd_data
rd_adr
buf_emp[0]
buf_emp[1]
llu_en
line_rd
dwu_llu_line_wr
llu_dwu_line_rd
wr_data
wr_en
wr_adr
join_point
line_size
llu_go_pulse
fifolevel
fifo_read_thres
current_ptr_a/b
color_base_adr
join_word
color_enable
color_rel_line
ptr_wr_en
ptr_wr_adr
ptr_wr_data
pcu_rwn
pcu_llu_sel
pcu_ad_adr
pcu_datout
llu_pcu_datain
llu_pcu_rdy
llu_diu_radr
diu_data
diu_llu_rvalid
diu_llu_rack
llu_diu_rreq

FIG. 273

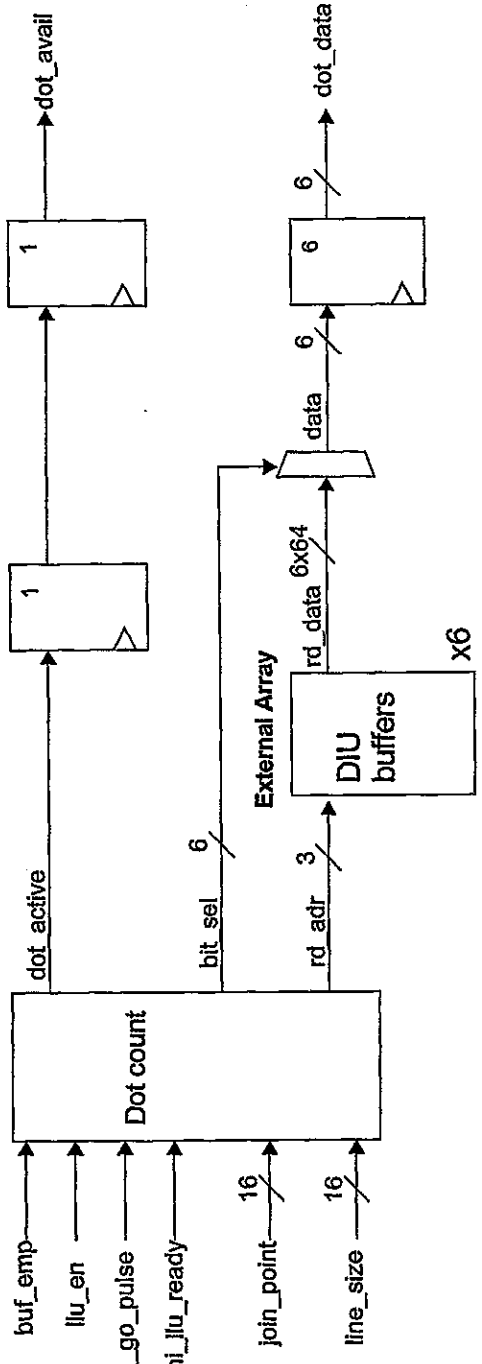

dot_avail
dot_data
1
6
6
6
data
1
6x64
rd_data
External Array
DIU buffers
x6
dot_active
bit_sel
6
rd_adr
3
Dot count
buf_emp
llu_en
llu_go_pulse
phi_llu_ready
join_point
16
line_size
16

FIG. 274

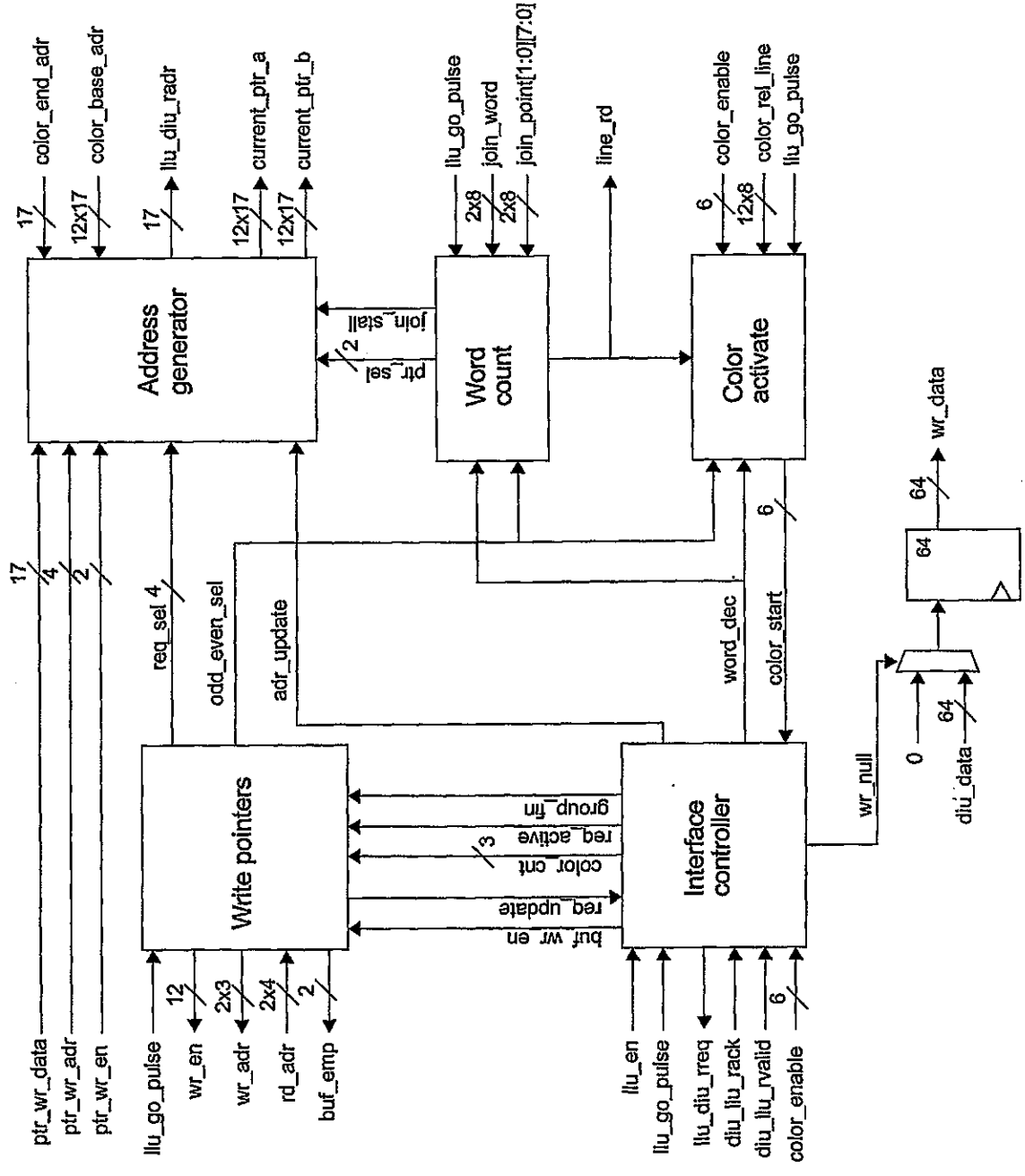
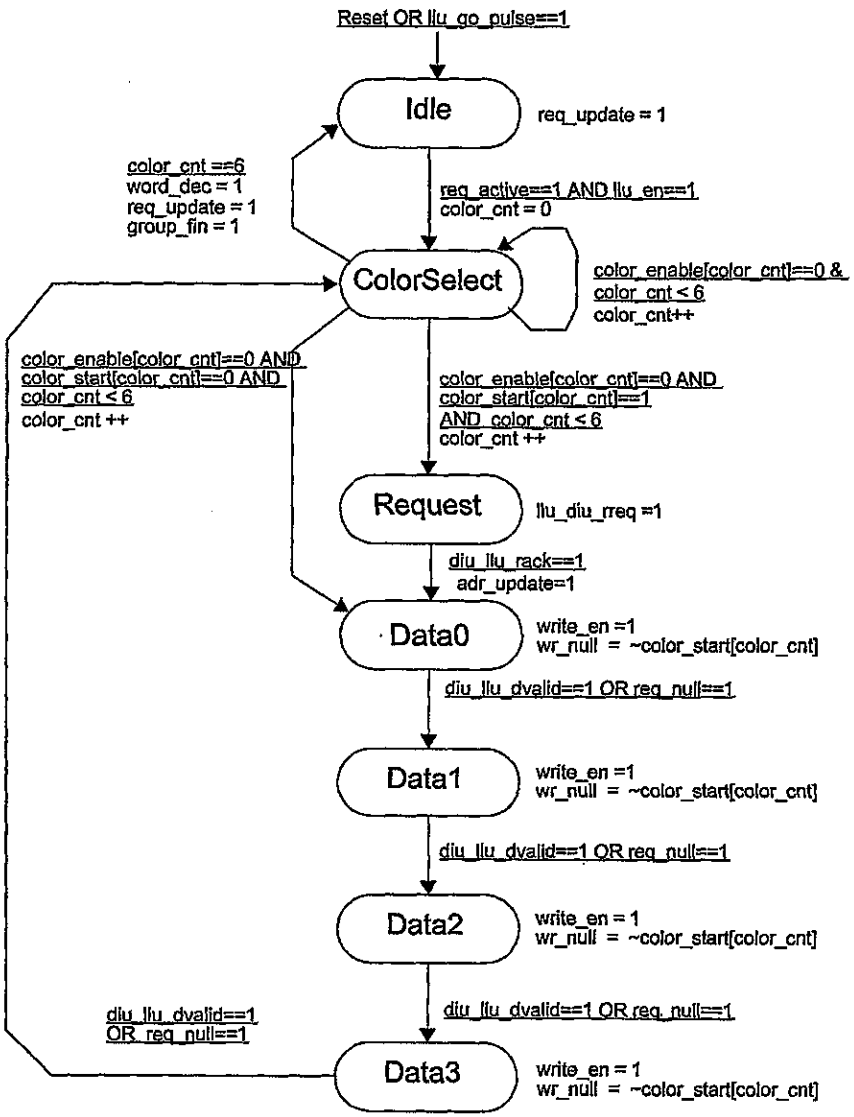

Address generator
Word count
Color activate
Write pointers
Interface controller
color_end_adr
color_base_adr
llu_diu_radr
current_ptr_a
current_ptr_b
llu_go_pulse
join_word
join_point[1:0][7:0]
line_rd
color_enable
color_rel_line
llu_go_pulse
join_stall
ptr_sel
req_sel
odd_even_sel
adr_update
word_dec
color_start
wr_data
wr_null
diu_data
ptr_wr_data
ptr_wr_adr
ptr_wr_en
llu_go_pulse
wr_en
wr_adr
rd_adr
buf_emp
buf_wr_en
req_update
color_cnt
req_active
group_fin
llu_en
llu_go_pulse
llu_diu_rreq
diu_llu_rack
diu_llu_rvalid
color_enable

FIG. 275



Machine remains in same state by default
All outputs are zero unless otherwise stated

State Description:

Idle:	Idle state wait for active request
ColorSelect:	Select the color to update before requesting to DIU
Request:	Request issued wait for acknowledge
Data0:	Data word 0 transfer
Data1:	Data word 1 transfer
Data2:	Data word 2 transfer
Data3:	Data word 3 transfer

FIG. 276

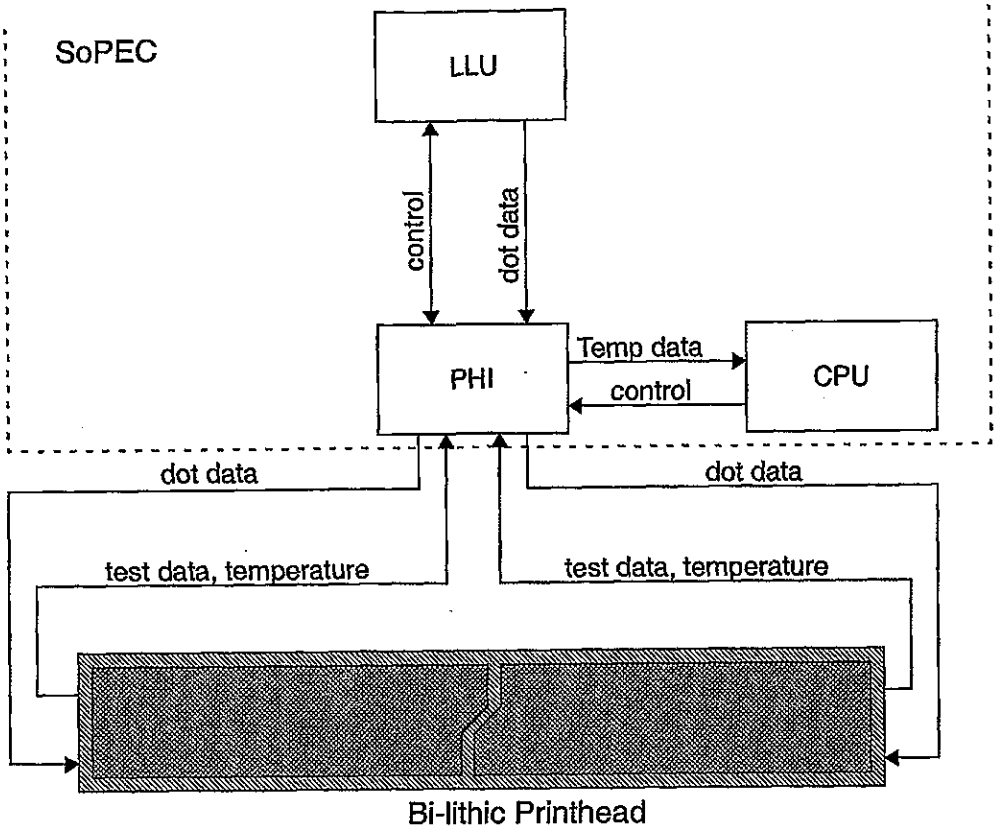


FIG. 277

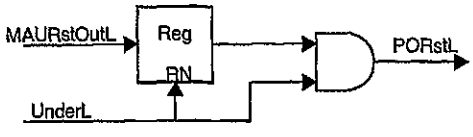


FIG. 278

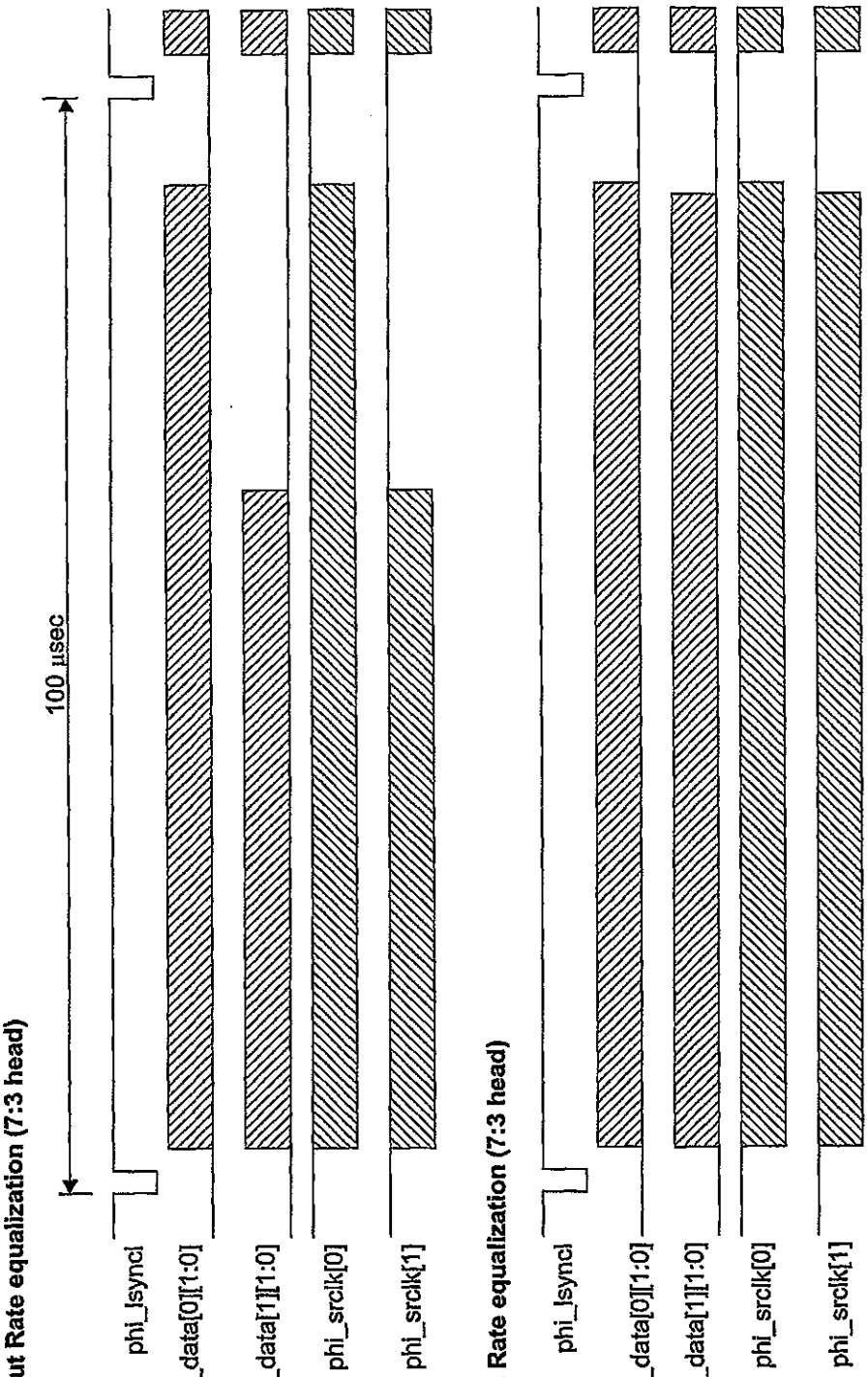

Without Rate equalization (7:3 head)
100 µsec
phi_lsyncl
phi_ph_data[0][1:0]
phi_ph_data[1][1:0]
phi_srclk[0]
phi_srclk[1]
With Rate equalization (7:3 head)
phi_lsyncl
phi_ph_data[0][1:0]
phi_ph_data[1][1:0]
phi_srclk[0]
phi_srclk[1]

FIG. 279

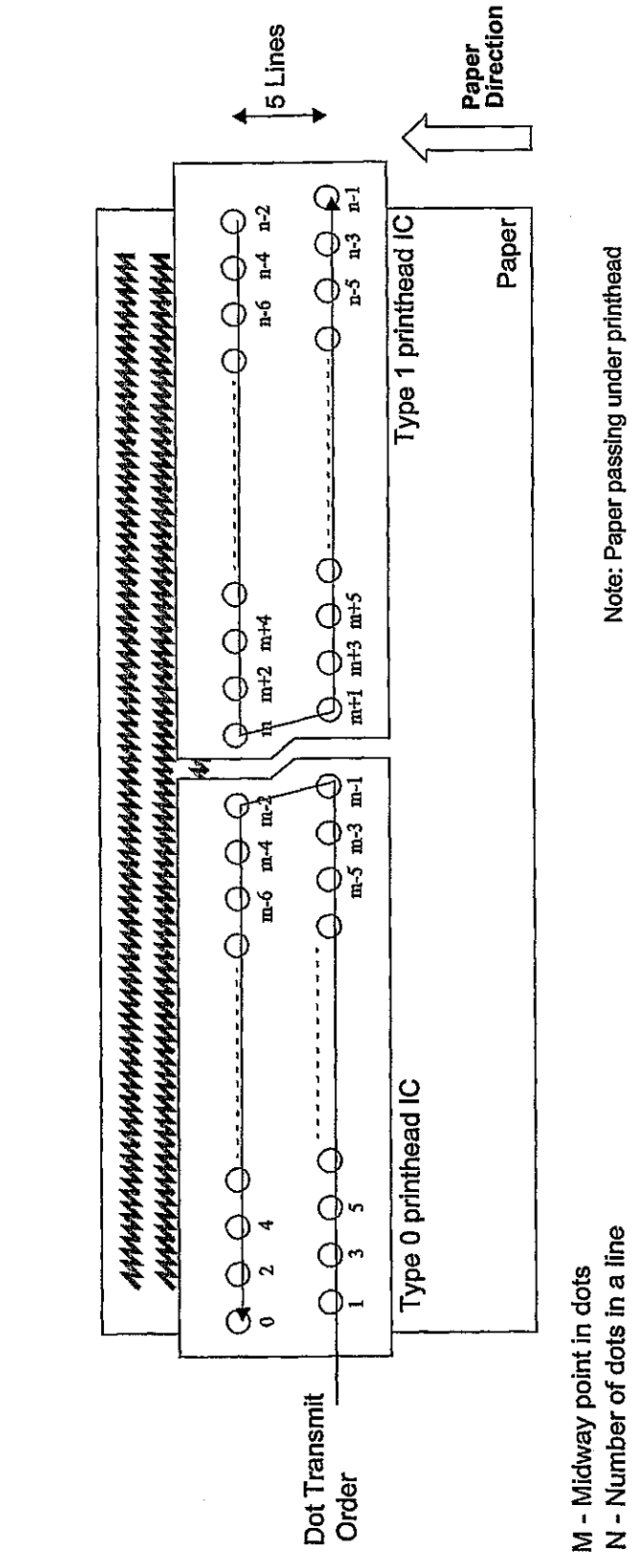

5 Lines
Paper Direction
Type 1 printhead IC
Paper
Type 0 printhead IC
Dot Transmit Order
0 2 4
1 3 5
m-6 m-4 m-2
m-5 m-3 m-1
m m+2 m+4
m+1 m+3 m+5
n-6 n-4 n-2
n-5 n-3 n-1
Note: Paper passing under printhead
M - Midway point in dots
N - Number of dots in a line

FIG. 280

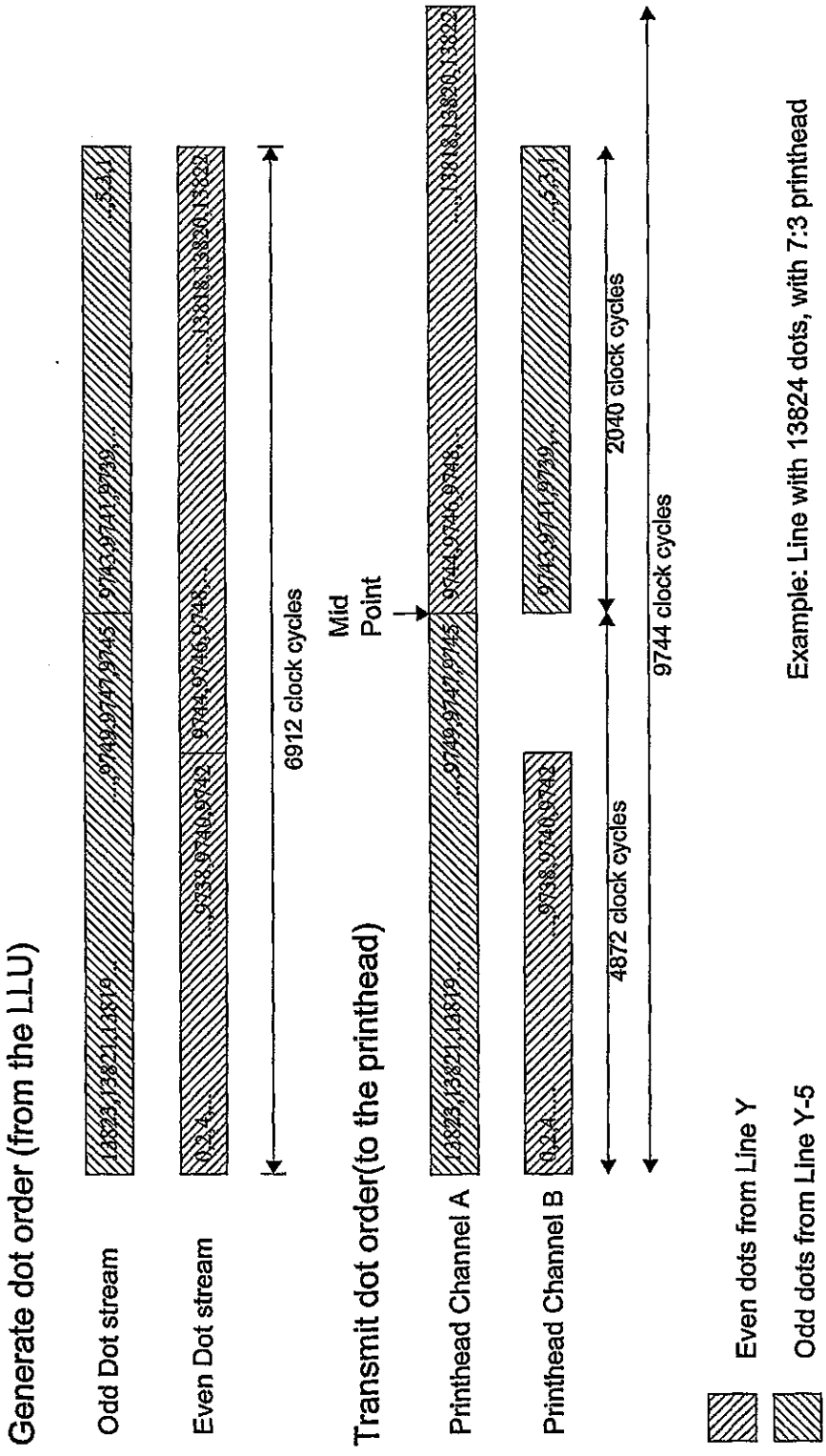

Generate dot order (from the LLU)
Odd Dot stream
Even Dot stream
6912 clock cycles
Transmit dot order(to the printhead)
Mid Point
Printhead Channel A
Printhead Channel B
4872 clock cycles
2040 clock cycles
9744 clock cycles
Even dots from Line Y
Odd dots from Line Y-5
Example: Line with 13824 dots, with 7:3 printhead

FIG. 281

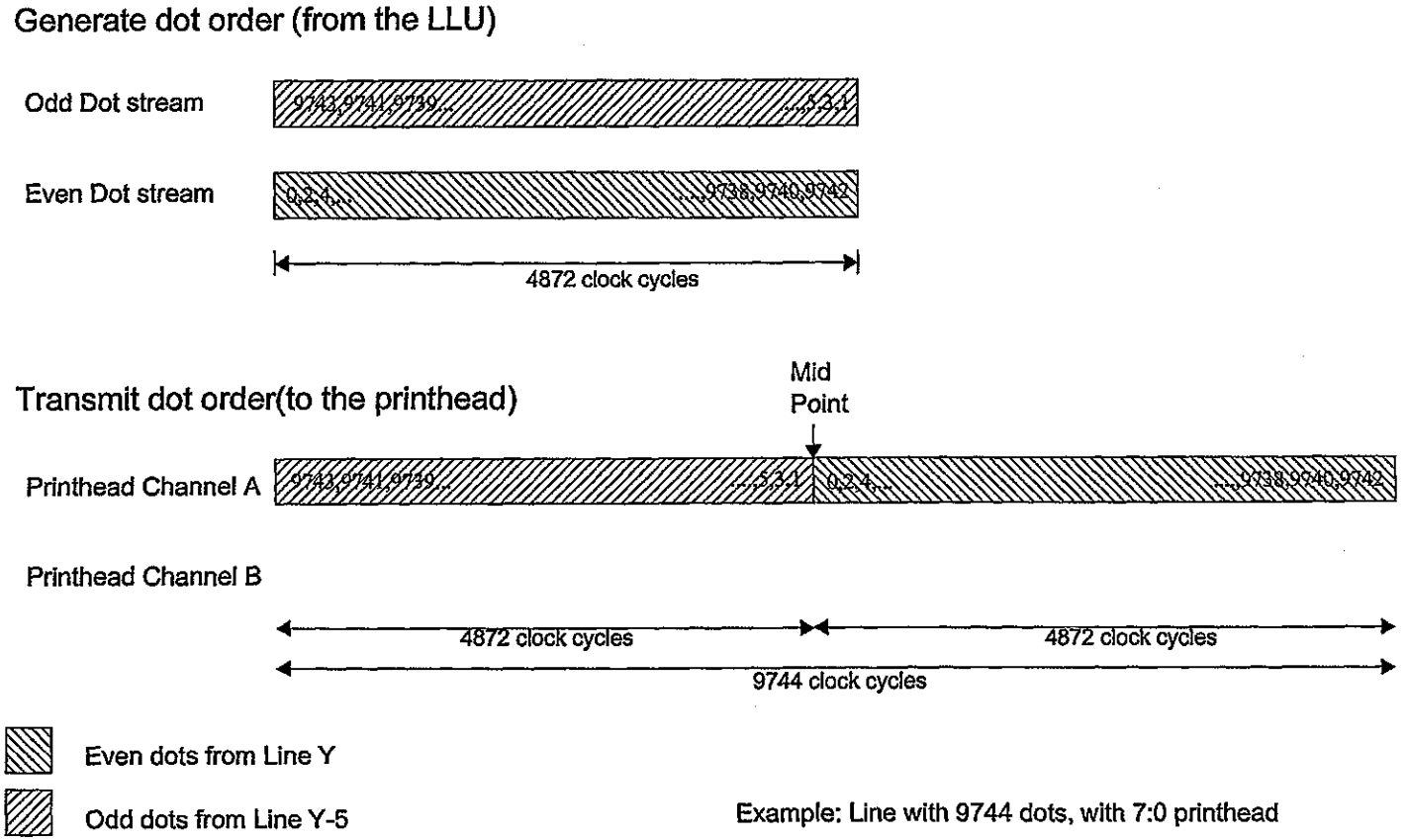

Generate dot order (from the LLU)
Odd Dot stream
9743,9741,9739...
...5,3,1
Even Dot stream
0,2,4...
...9738,9740,9742
4872 clock cycles
Transmit dot order(to the printhead)
Mid Point
Printhead Channel A
9743,9741,9739...
...5,3,1
0,2,4...
...9738,9740,9742
Printhead Channel B
4872 clock cycles
4872 clock cycles
9744 clock cycles
Even dots from Line Y
Odd dots from Line Y-5
Example: Line with 9744 dots, with 7:0 printhead

FIG. 282

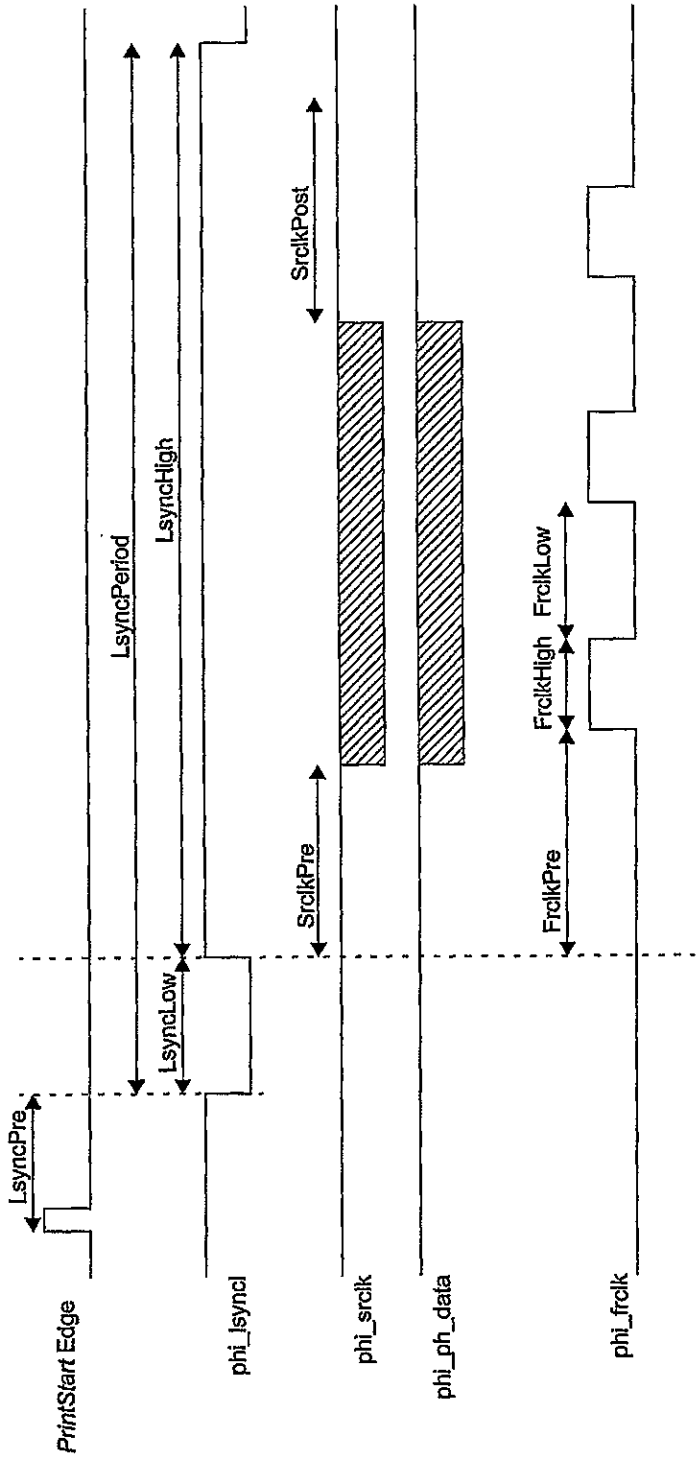

PrintStart Edge
phi_lsyncl
phi_srclk
phi_ph_data
phi_frclk
LsyncPre
LsyncPeriod
LsyncHigh
LsyncLow
SrclkPre
SrclkPost
FrclkPre
FrclkHigh
FrclkLow

FIG. 283

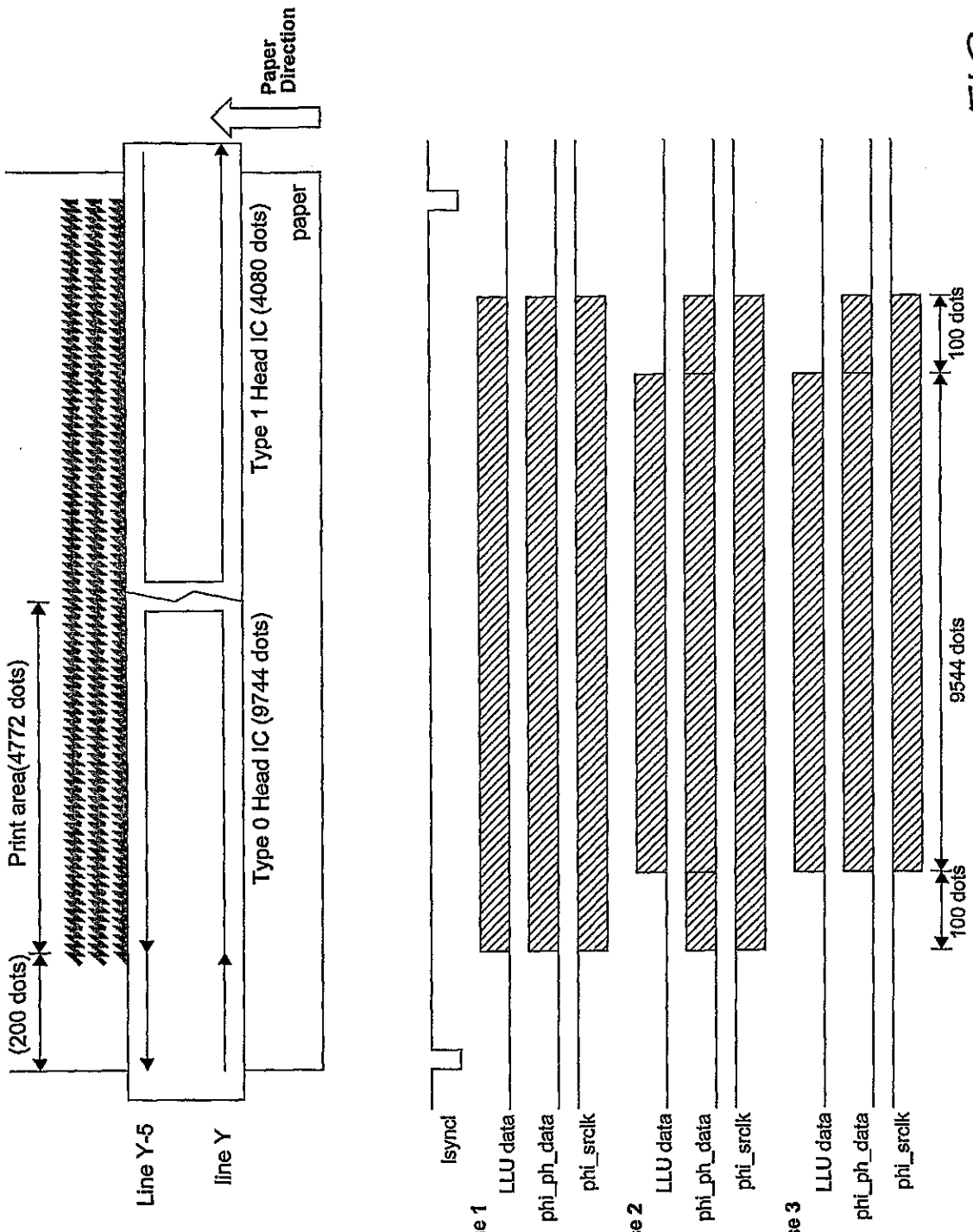

Paper Direction
Print area(4772 dots)
(200 dots)
Type 1 Head IC (4080 dots)
paper
Type 0 Head IC (9744 dots)
Line Y-5
line Y
lsyncl
Case 1
LLU data
phi_ph_data
phi_srclk
Case 2
LLU data
phi_ph_data
phi_srclk
Case 3
LLU data
phi_ph_data
phi_srclk
100 dots
9544 dots
100 dots

FIG. 284

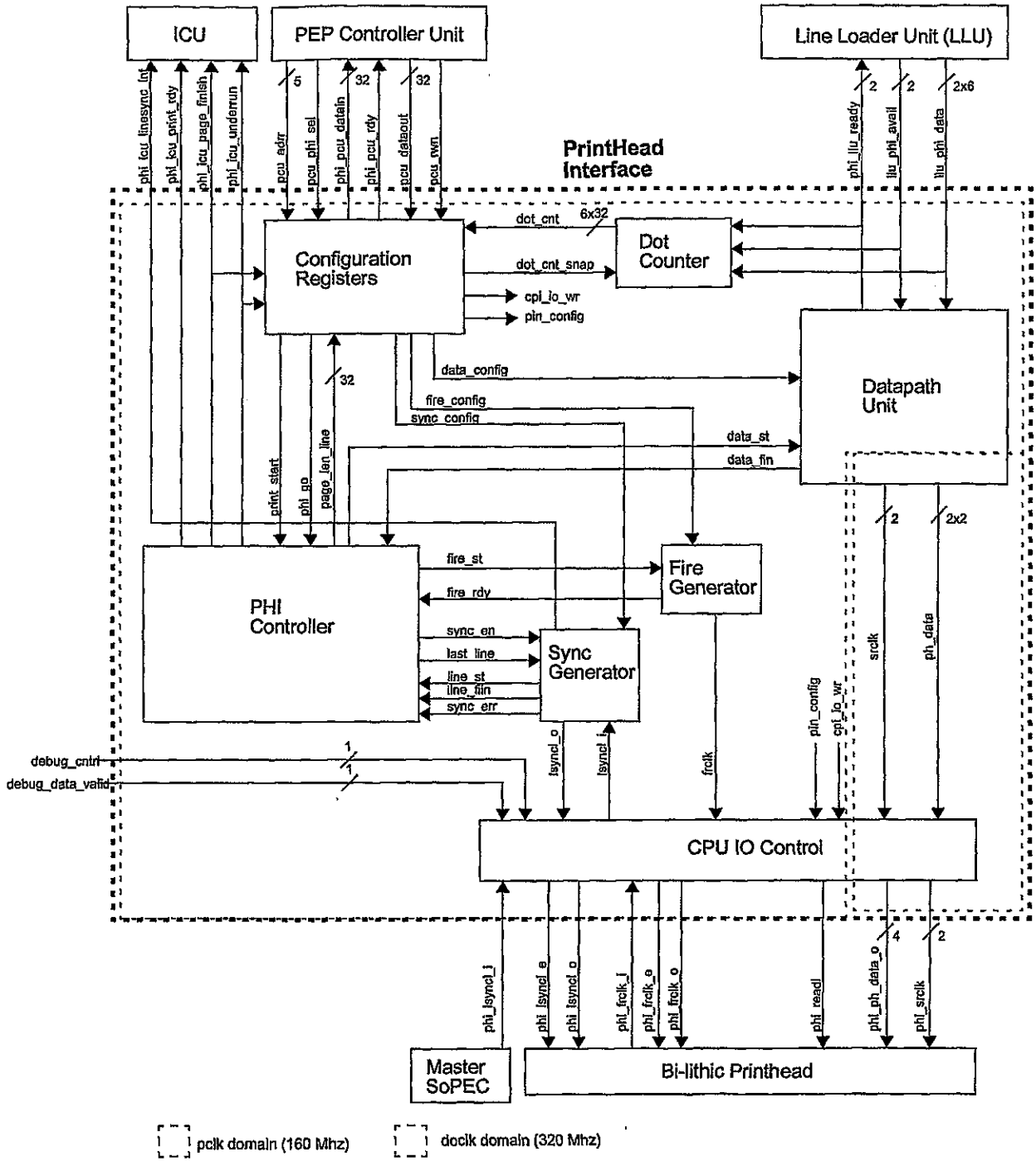

ICU
PEP Controller Unit
Line Loader Unit (LLU)
PrintHead Interface
Configuration Registers
Dot Counter
Datapath Unit
PHI Controller
Fire Generator
Sync Generator
CPU IO Control
Master SoPEC
Bi-lithic Printhead
phi_icu_linesync_int
phi_icu_print_rdy
phi_icu_page_finish
phi_icu_underrun
pcu_adr
pcu_phi_sel
phi_pcu_datain
phi_pcu_rdy
pcu_dataout
pcu_rwn
phi_llu_ready
llu_phi_avail
llu_phi_data
dot_cnt
dot_cnt_snap
cpi_io_wr
pin_config
data_config
fire_config
sync_config
data_st
data_fin
print_start
phi_go
page_len_line
fire_st
fire_rdy
sync_en
last_line
line_st
line_fin
sync_err
lsyncl_o
lsyncl_i
frclk
srclk
ph_data
debug_cntrl
debug_data_valid
phi_lsyncl_i
phi_lsyncl_e
phi_lsyncl_o
phi_frclk_i
phi_frclk_e
phi_frclk_o
phi_readl
phi_ph_data_o
phi_srclk
pclk domain (160 Mhz)
dockk domain (320 Mhz)

FIG. 285

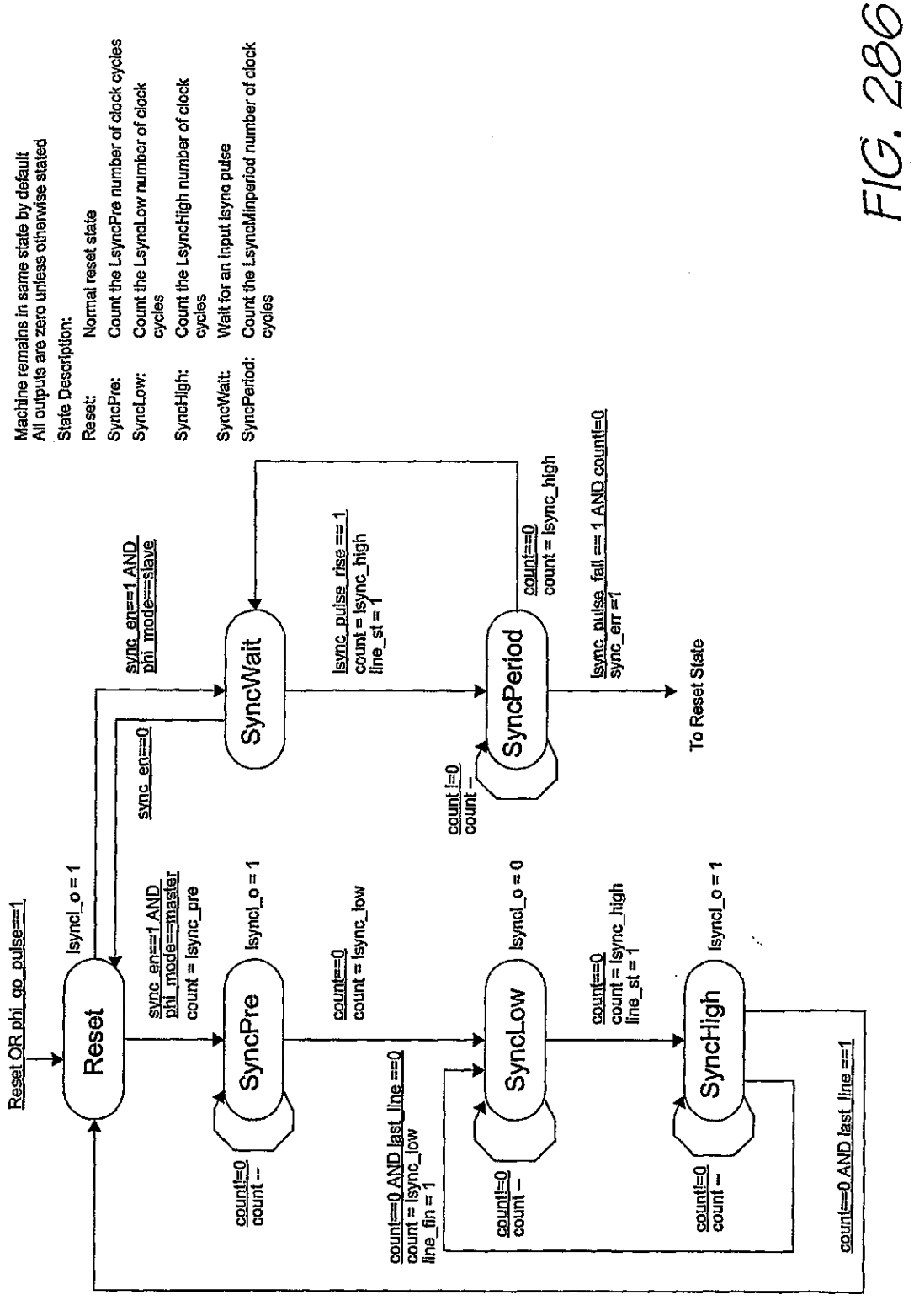

Machine remains in same state by default
All outputs are zero unless otherwise stated
State Description:
Reset: Normal reset state
SyncPre: Count the LsyncPre number of clock cycles
SyncLow: Count the LsyncLow number of clock cycles
SyncHigh: Count the LsyncHigh number of clock cycles
SyncWait: Wait for an input lsync pulse
SyncPeriod: Count the LsyncMinperiod number of clock cycles
Reset OR phi_go_pulse==1
lsyncl_o = 1
Reset
sync_en==1 AND phi_mode==master
count = lsync_pre
SyncPre
lsyncl_o = 1
count!=0
count --
count==0
count = lsync_low
SyncLow
lsyncl_o = 0
count!=0
count --
count==0
count = lsync_high
line_st = 1
SyncHigh
lsyncl_o = 1
count!=0
count --
count==0 AND last_line ==0
count = lsync_low
line_fin = 1
count==0 AND last_line ==1
sync_en==1 AND phi_mode==slave
sync_en==0
SyncWait
lsync_pulse_rise == 1
count = lsync_high
line_st = 1
count!=0
count --
SyncPeriod
count==0
count = lsync_high
lsync_pulse_fall == 1 AND count!=0
sync_err =1
To Reset State

FIG. 286

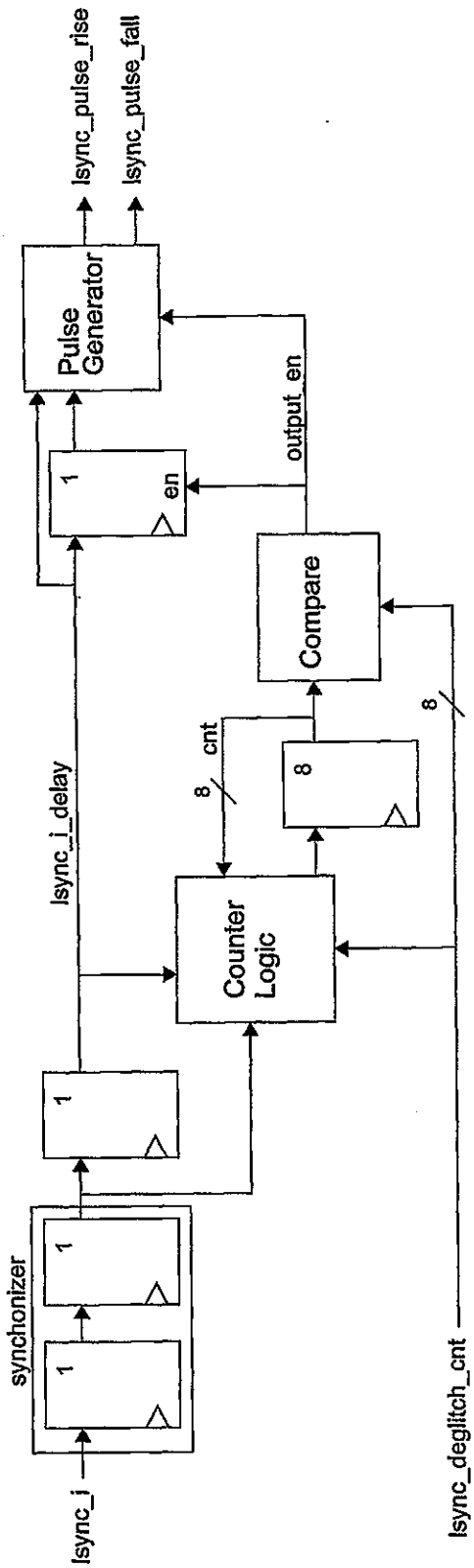
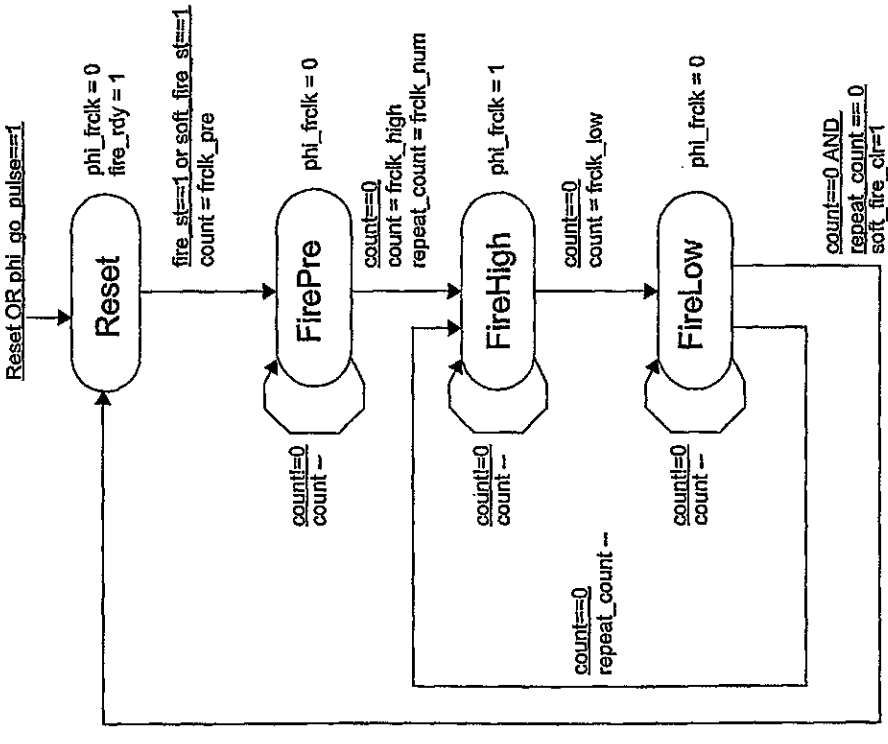

lsync_pulse_rise
lsync_pulse_fall
Pulse Generator
output_en
1
en
Compare
8
cnt
8
lsync_i_delay
Counter Logic
1
synchonizer
1
1
lsync_i
lsync_deglitch_cnt

FIG. 287



Machine remains in same state by default
All outputs are zero unless otherwise stated

State Description:

Reset: Normal reset state

FirePre: Count the FrclkPre number of clock cycles, repeat count set to FrclkNum

FireHigh: Count the FrclkHigh number of clock cycles

FireLow: Count the FrclkLow number of clock cycles

FIG. 288

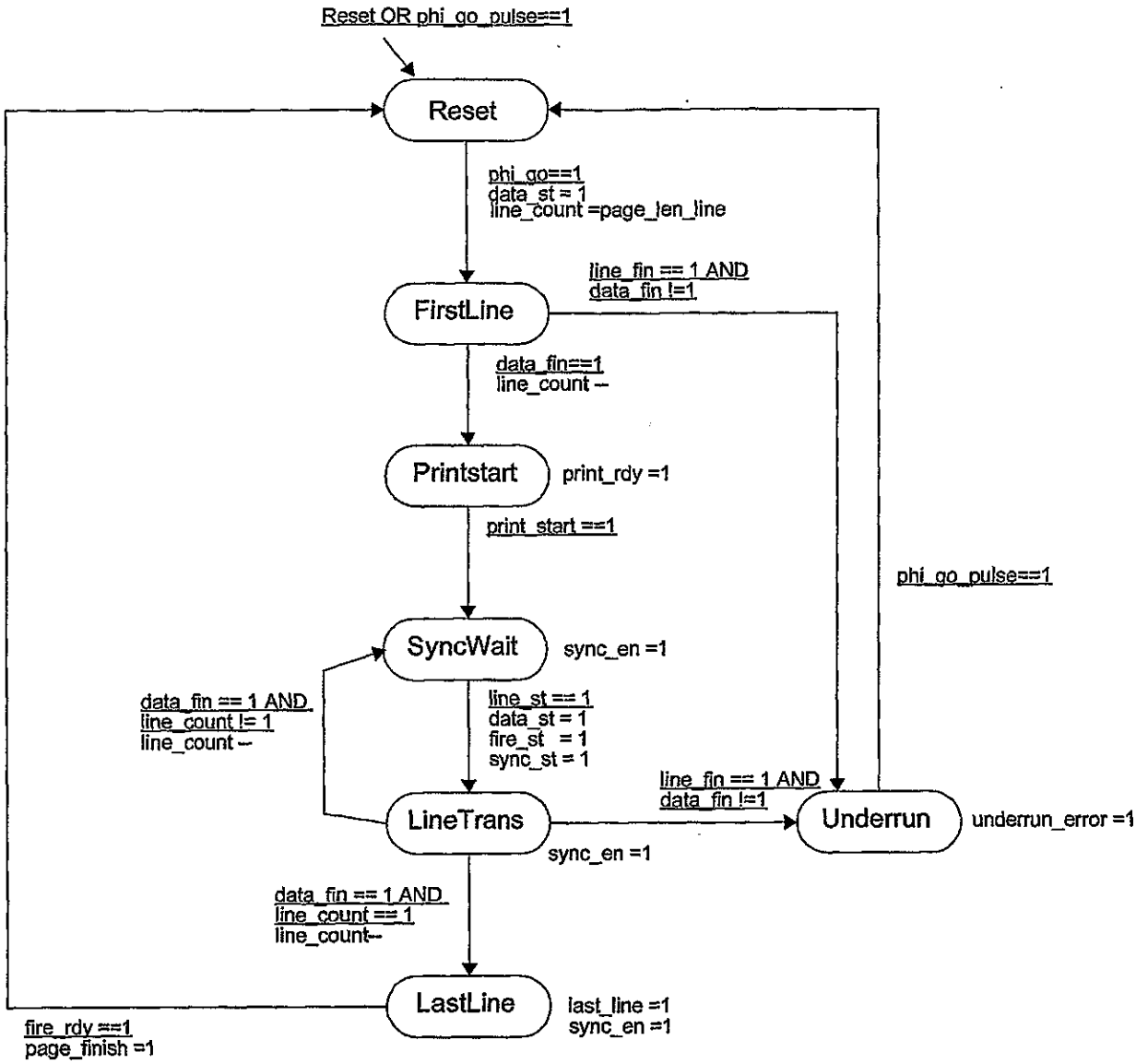

Reset OR phi_go_pulse==1
Reset
phi_go==1
data_st = 1
line_count =page_len_line
line_fin == 1 AND
data_fin !=1
FirstLine
data_fin==1
line_count --
Printstart
print_rdy =1
print_start ==1
phi_go_pulse==1
SyncWait
sync_en =1
data_fin == 1 AND
line_count != 1
line_count --
line_st == 1
data_st = 1
fire_st = 1
sync_st = 1
line_fin == 1 AND
data_fin !=1
LineTrans
Underrun
underrun_error =1
sync_en =1
data_fin == 1 AND
line_count == 1
line_count--
LastLine
last_line =1
sync_en =1
fire_rdy ==1
page_finish =1

FIG. 289

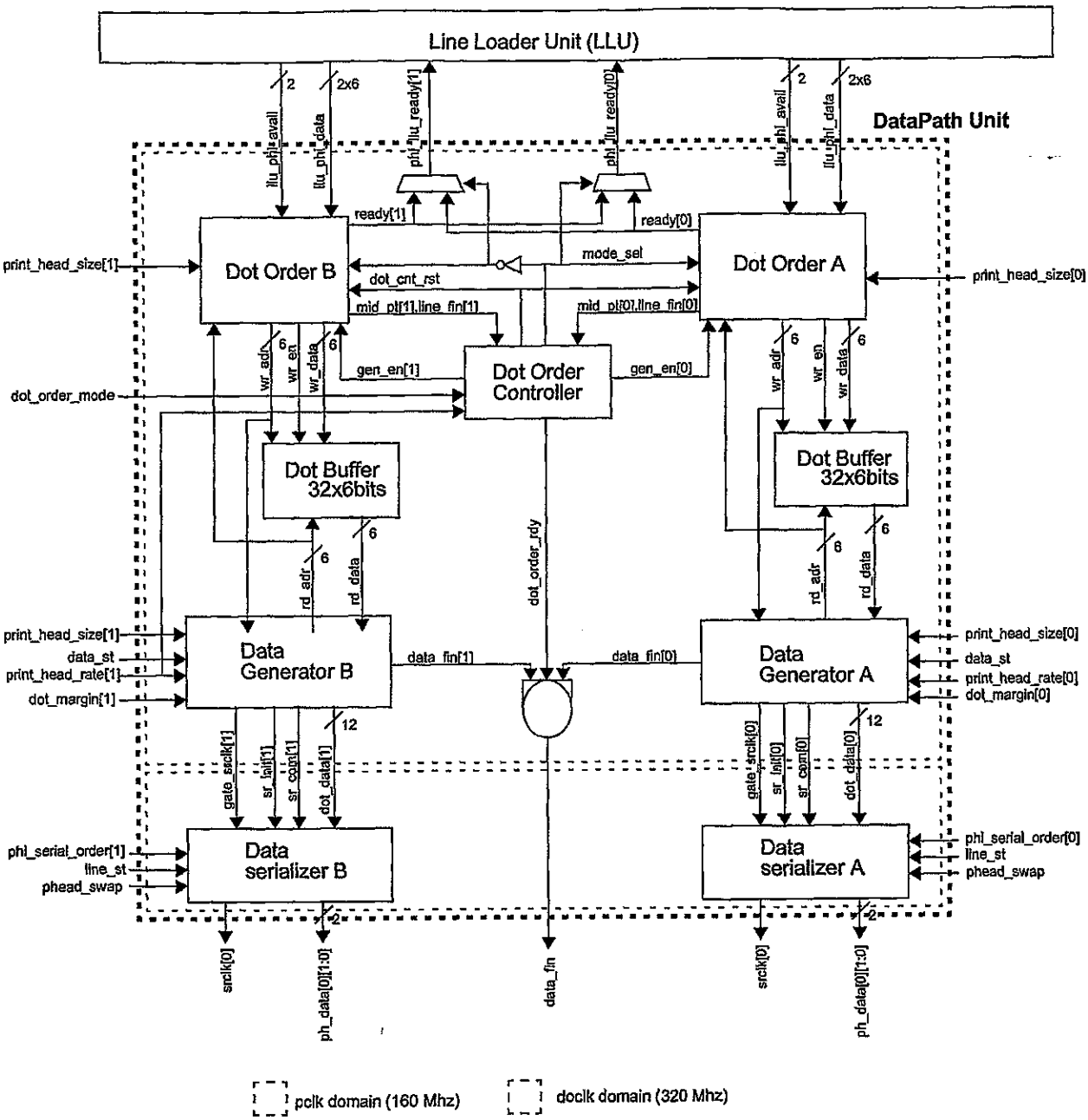

Line Loader Unit (LLU)
DataPath Unit
Dot Order B
Dot Order A
Dot Order Controller
Dot Buffer 32x6bits
Dot Buffer 32x6bits
Data Generator B
Data Generator A
Data serializer B
Data serializer A
print_head_size[1]
print_head_size[0]
dot_order_mode
data_st
print_head_rate[1]
print_head_rate[0]
dot_margin[1]
dot_margin[0]
phl_serial_order[1]
phl_serial_order[0]
line_st
phead_swap
ready[1]
ready[0]
mode_sel
dot_cnt_rst
gen_en[1]
gen_en[0]
data_fin[1]
data_fin[0]
dot_order_rdy
srclk[0]
data_fin
pclk domain (160 Mhz)
doclk domain (320 Mhz)

FIG. 290

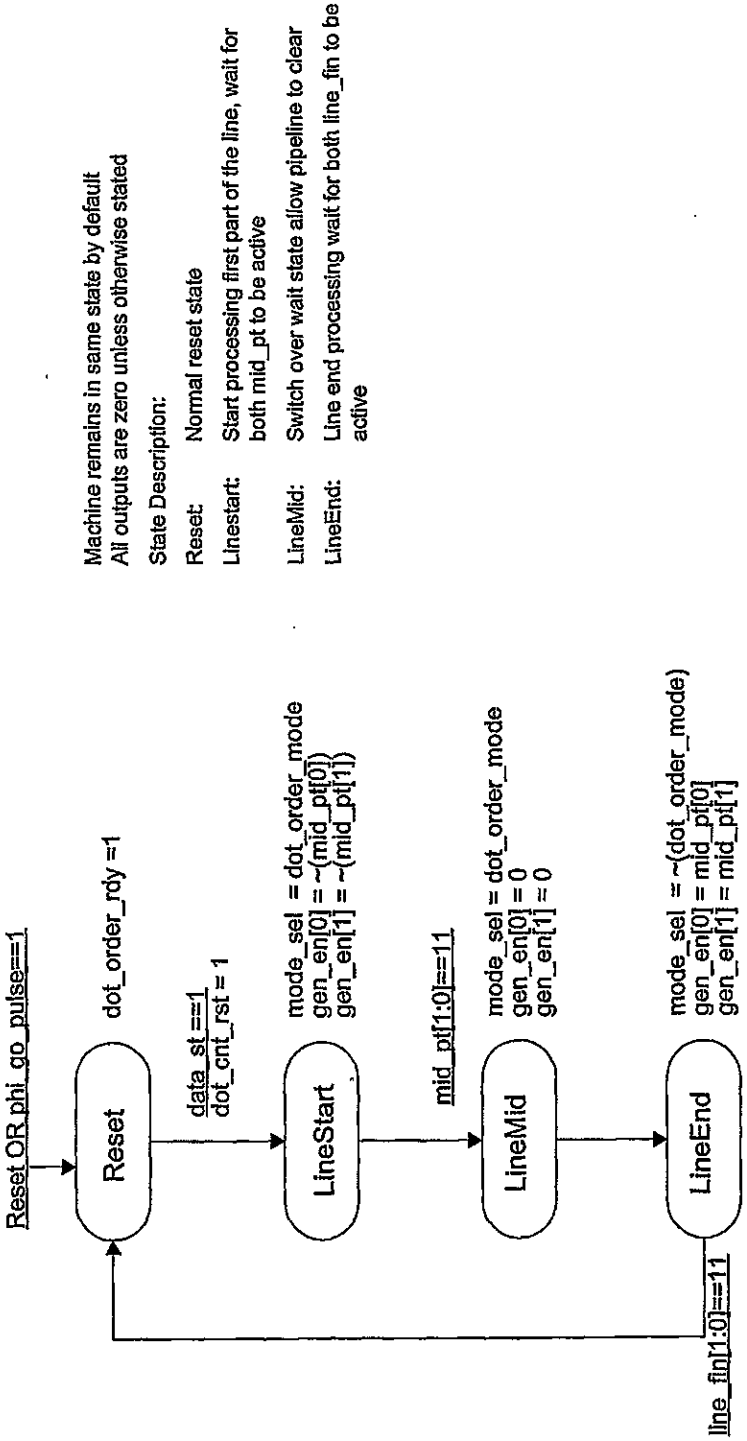

Machine remains in same state by default
All outputs are zero unless otherwise stated
State Description:
Reset: Normal reset state
Linestart: Start processing first part of the line, wait for both mid_pt to be active
LineMid: Switch over wait state allow pipeline to clear
LineEnd: Line end processing wait for both line_fin to be active
Reset OR phi_go_pulse==1
Reset
dot_order_rdy =1
data_st ==1
dot_cnt_rst = 1
LineStart
mode_sel = dot_order_mode
gen_en[0] = ~(mid_pt[0])
gen_en[1] = ~(mid_pt[1])
mid_pt[1:0]==11
LineMid
mode_sel = dot_order_mode
gen_en[0] = 0
gen_en[1] = 0
LineEnd
mode_sel = ~(dot_order_mode)
gen_en[0] = mid_pt[0]
gen_en[1] = mid_pt[1]
line_fin[1:0]==11

FIG. 291

Machine remains in same state by default
All outputs are zero unless otherwise stated

State Description:

Reset: Normal reset state

SrclkPre: Count the SrclkPre number of clock cycles

DataGen1: Read Line Dot data from buffer

DataGen2: Read Line Dot data from buffer

DataGen3: Data gen wait state

Wait1: Data rate control wait state

Wait2: Data rate control wait state

Wait3: Data rate control wait state

MarginGen1:Generate *DotMargin* number of dots

MarginGen2:Generate *DotMargin* number of dots

MarginGen3:Generate margin wait state

SrclkPost: Wait for SrclkPost number of cycles

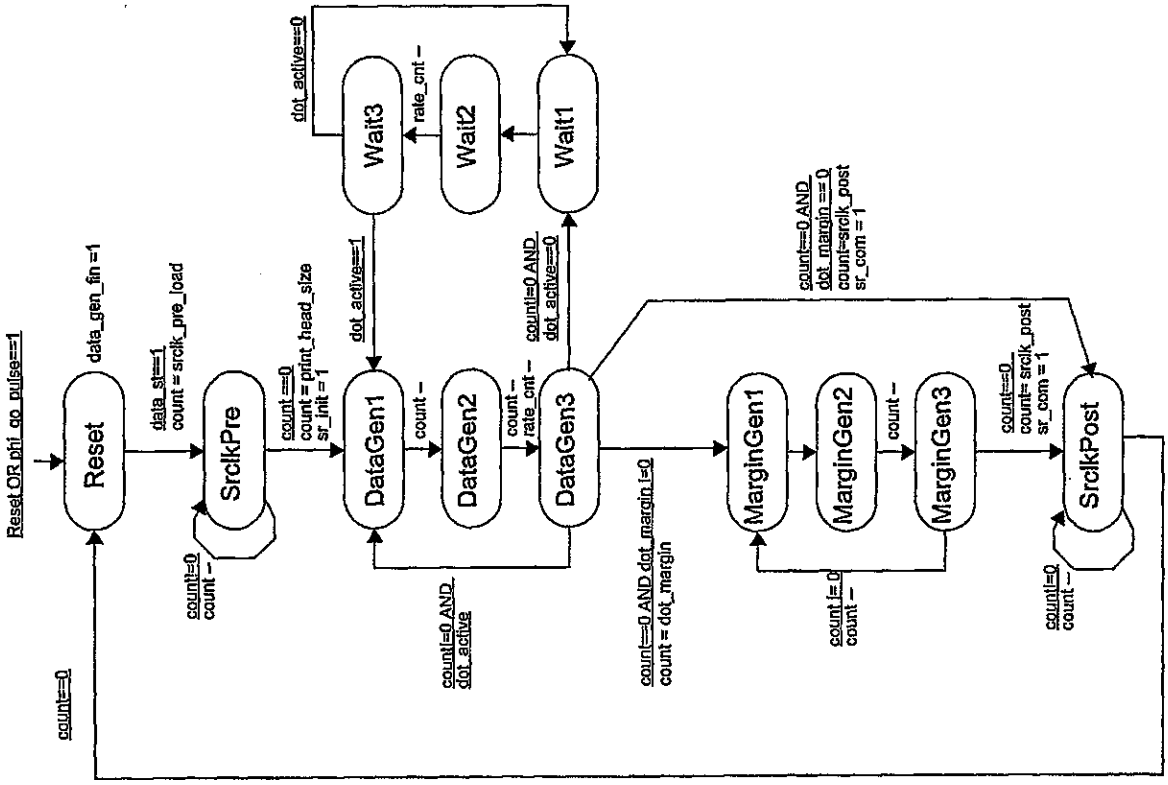


FIG. 292

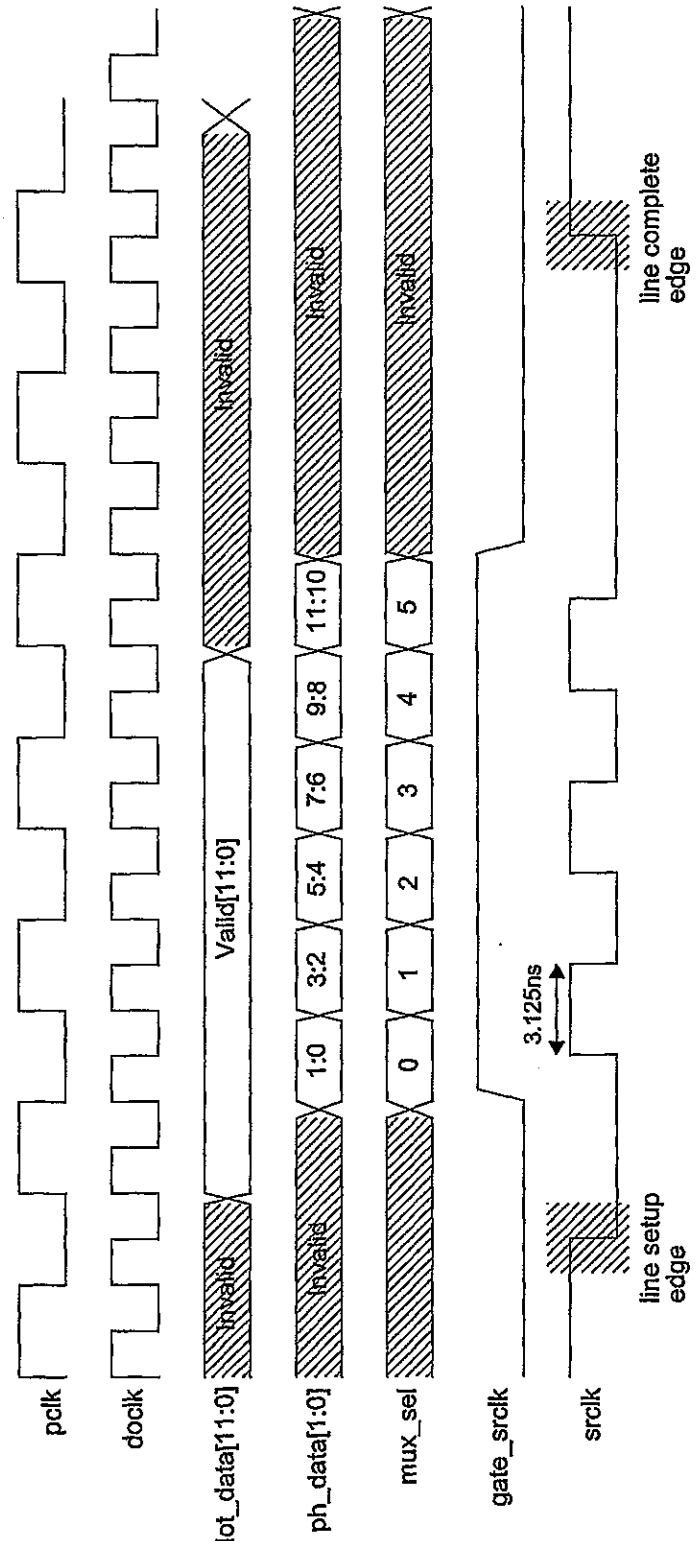

pclk
doclk
dot_data[11:0]
Invalid
Valid[11:0]
Invalid
ph_data[1:0]
Invalid
1:0
3:2
5:4
7:6
9:8
11:10
Invalid
mux_sel
Invalid
0
1
2
3
4
5
gate_srclk
3.125ns
srclk
line setup edge
line complete edge

FIG. 293

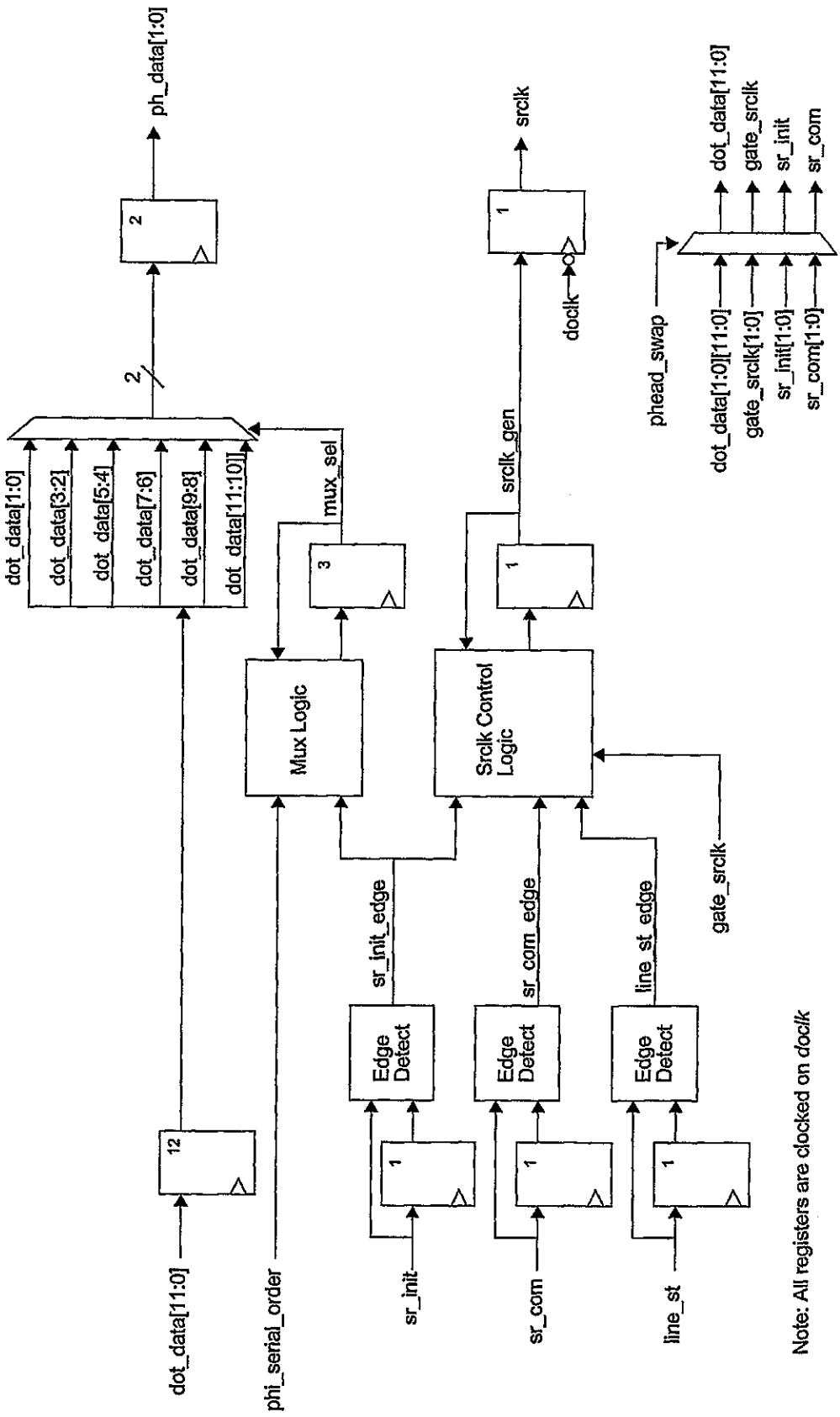

ph_data[1:0]
2
2
dot_data[1:0]
dot_data[3:2]
dot_data[5:4]
dot_data[7:6]
dot_data[9:8]
dot_data[11:10]
mux_sel
3
Mux Logic
srclk
1
dotclk
srclk_gen
1
Srclk Control Logic
phead_swap
dot_data[11:0]
gate_srclk
sr_init
sr_com
dot_data[1:0][11:0]
gate_srclk[1:0]
sr_init[1:0]
sr_com[1:0]
sr_init_edge
sr_com_edge
line_st_edge
gate_srclk
Edge Detect
Edge Defect
Edge Defect
1
1
1
12
dot_data[11:0]
phi_serial_order
sr_init
sr_com
line_st
Note: All registers are clocked on dotclk

FIG. 294

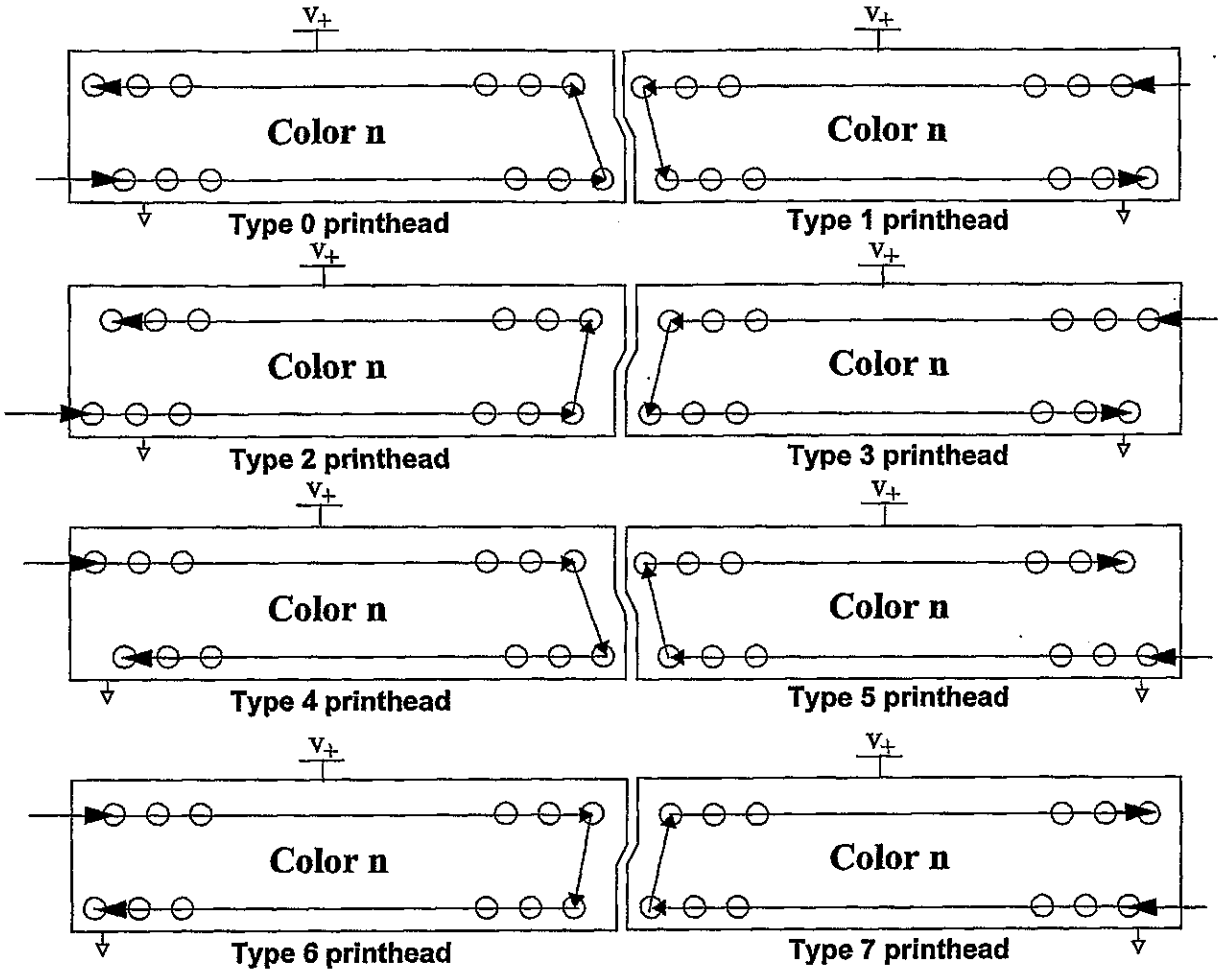

V+
Color n
Type 0 printhead
V+
Color n
Type 1 printhead
V+
Color n
Type 2 printhead
V+
Color n
Type 3 printhead
V+
Color n
Type 4 printhead
V+
Color n
Type 5 printhead
V+
Color n
Type 6 printhead
V+
Color n
Type 7 printhead

FIG. 295

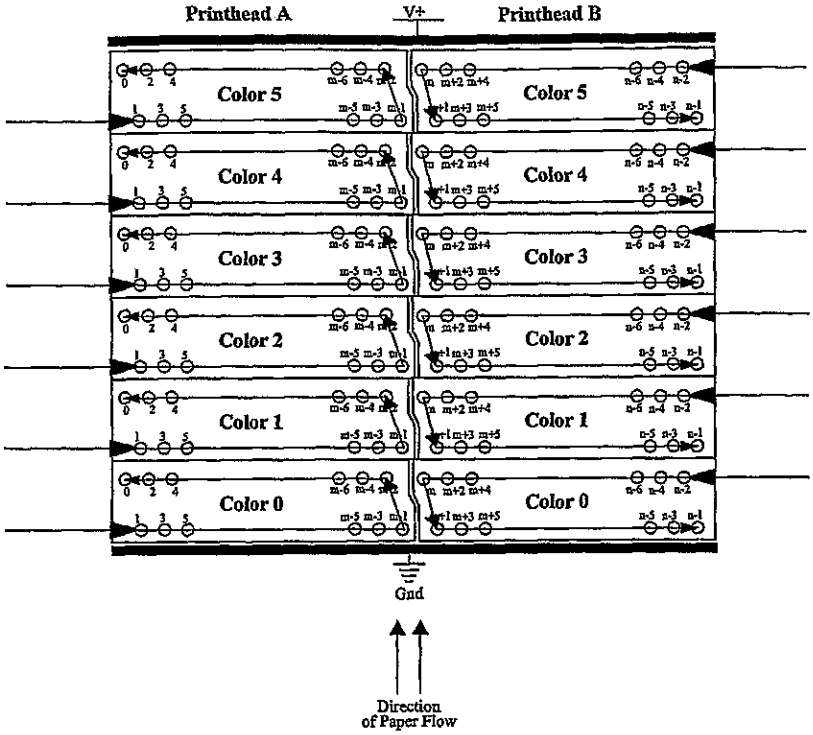

Printhead A
V+
Printhead B
Color 5
Color 4
Color 3
Color 2
Color 1
Color 0
Gnd
Direction
of Paper Flow

FIG. 296

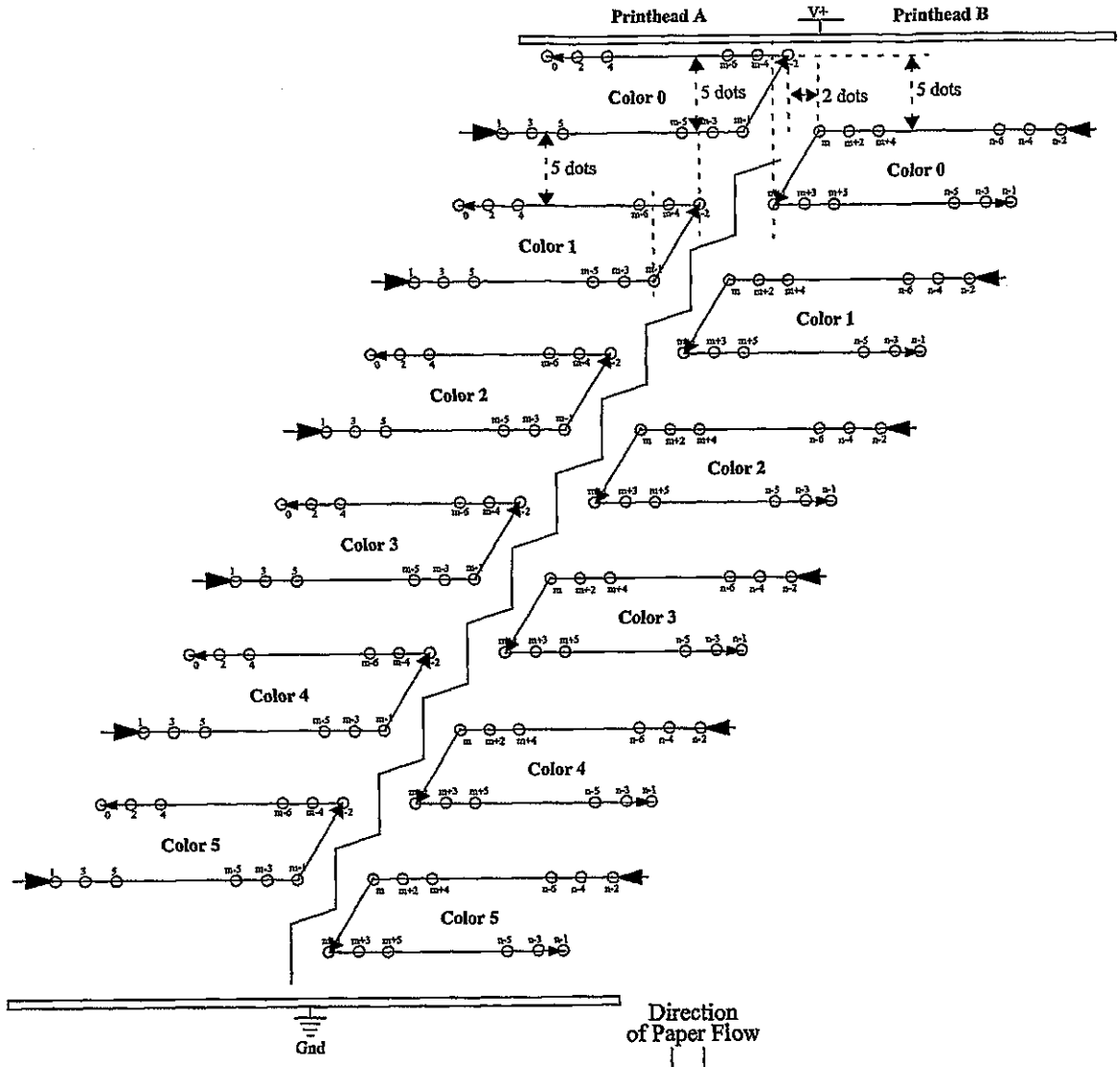

Printhead A
V+
Printhead B
Color 0
5 dots
2 dots
5 dots
5 dots
Color 0
Color 1
Color 1
Color 2
Color 2
Color 3
Color 3
Color 4
Color 4
Color 5
Color 5
Gnd
Direction of Paper Flow
The printheads are facing downwards.
The ink is being shot down onto the page.

FIG. 297

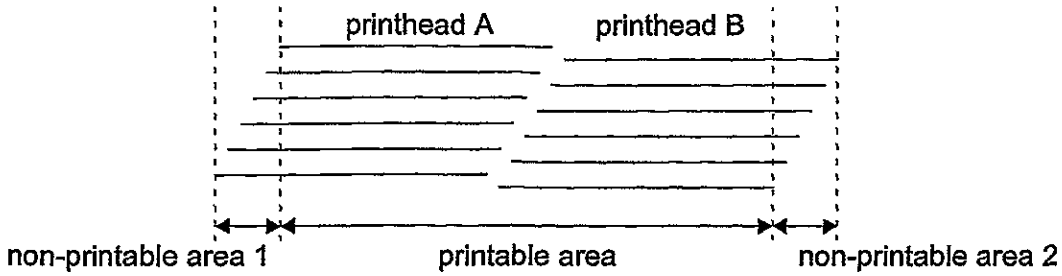

printhead A
printhead B
non-printable area 1
printable area
non-printable area 2

FIG. 298

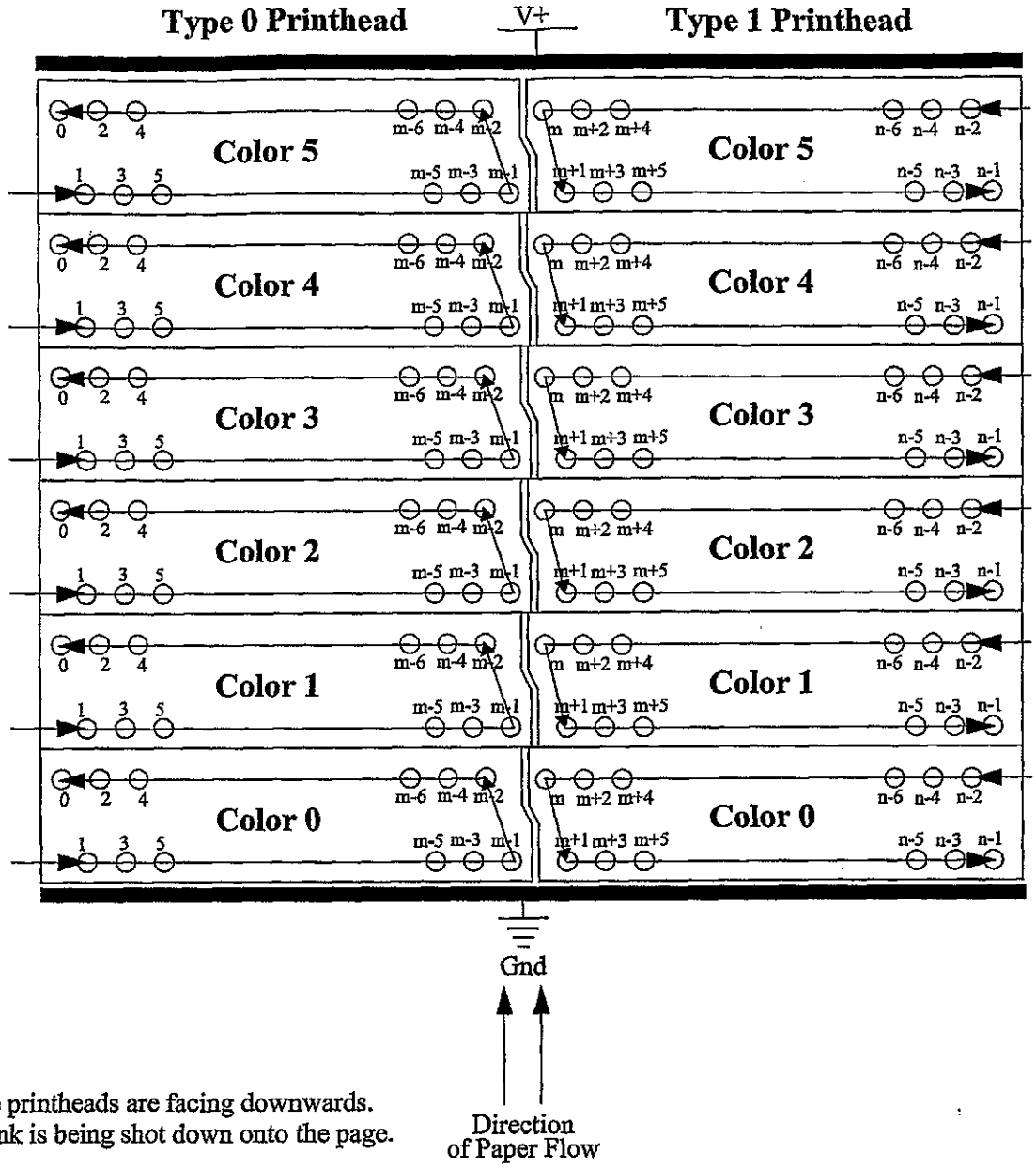

Type 0 Printhead
V+
Type 1 Printhead
Color 5
Color 4
Color 3
Color 2
Color 1
Color 0
0 2 4
1 3 5
m-6 m-4 m-2
m-5 m-3 m-1
m m+2 m+4
m+1 m+3 m+5
n-6 n-4 n-2
n-5 n-3 n-1
Gnd
The printheads are facing downwards.
The ink is being shot down onto the page.
Direction
of Paper Flow

FIG. 299

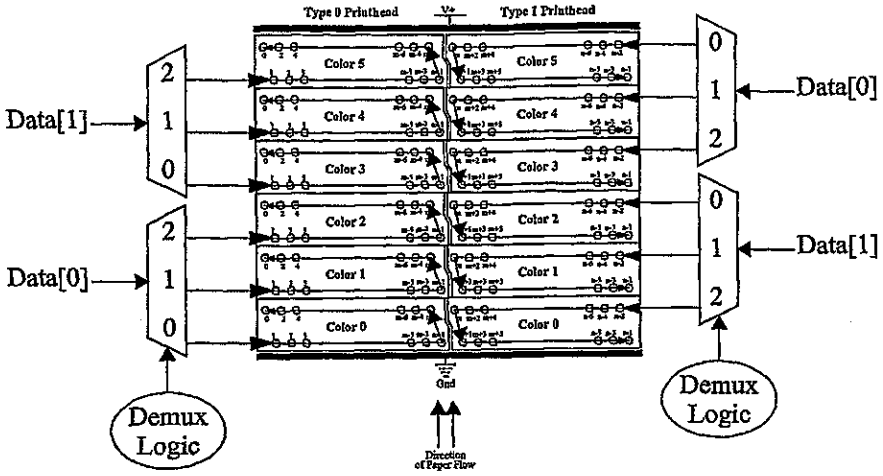


FIG. 300

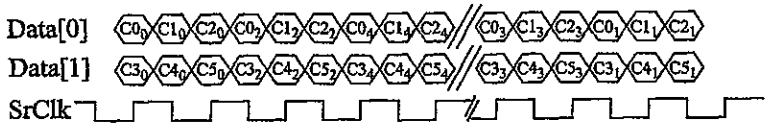


FIG. 301

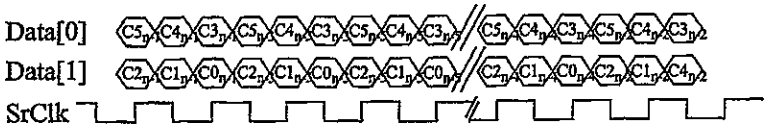


FIG. 302

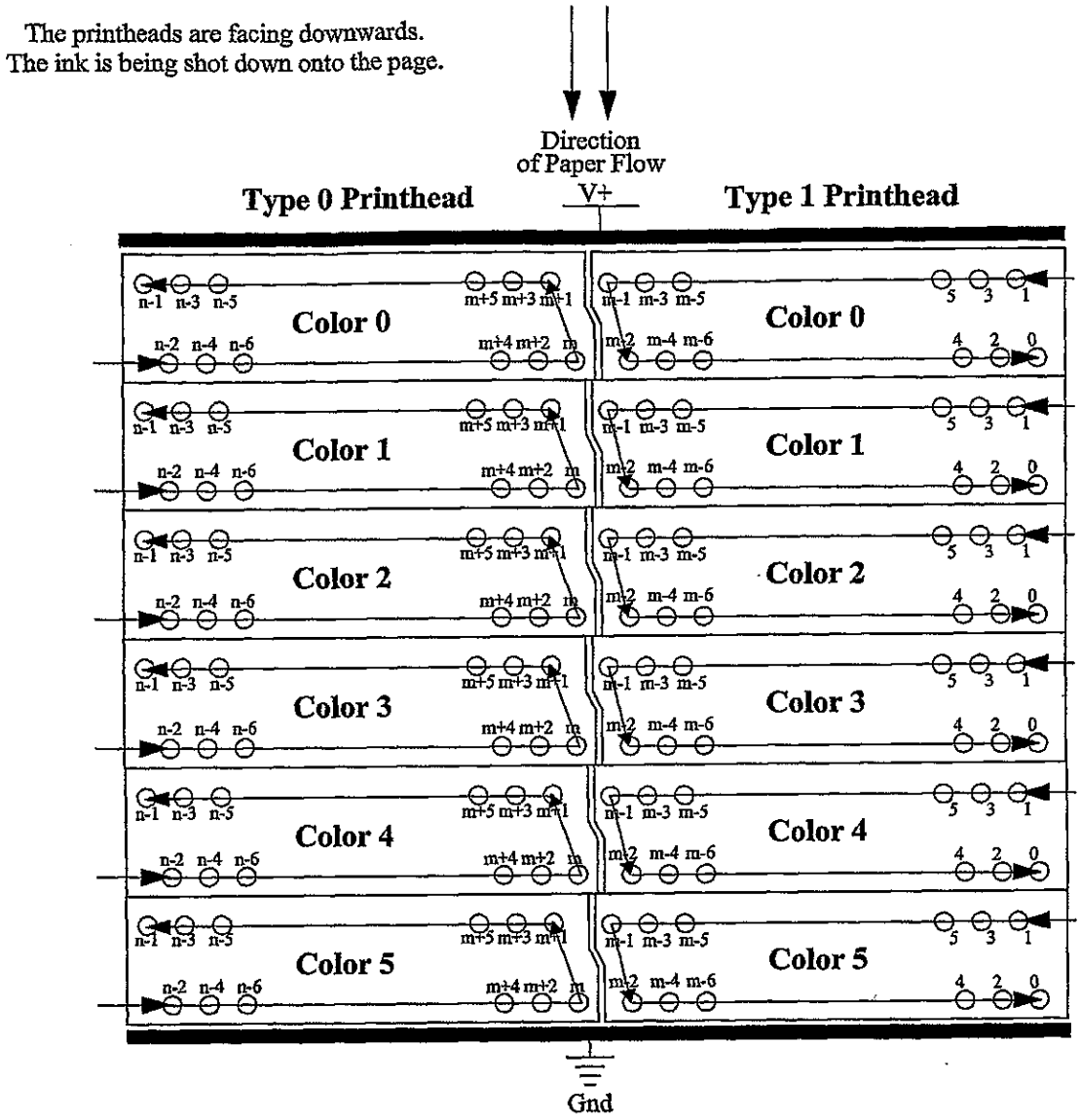

The printheads are facing downwards.
The ink is being shot down onto the page.
Direction of Paper Flow
V+
Type 0 Printhead
Type 1 Printhead
Color 0
Color 1
Color 2
Color 3
Color 4
Color 5
n-1 n-3 n-5
n-2 n-4 n-6
m+5 m+3 m+1
m+4 m+2 m
m-1 m-3 m-5
m-2 m-4 m-6
5 3 1
4 2 0
Gnd

FIG. 303

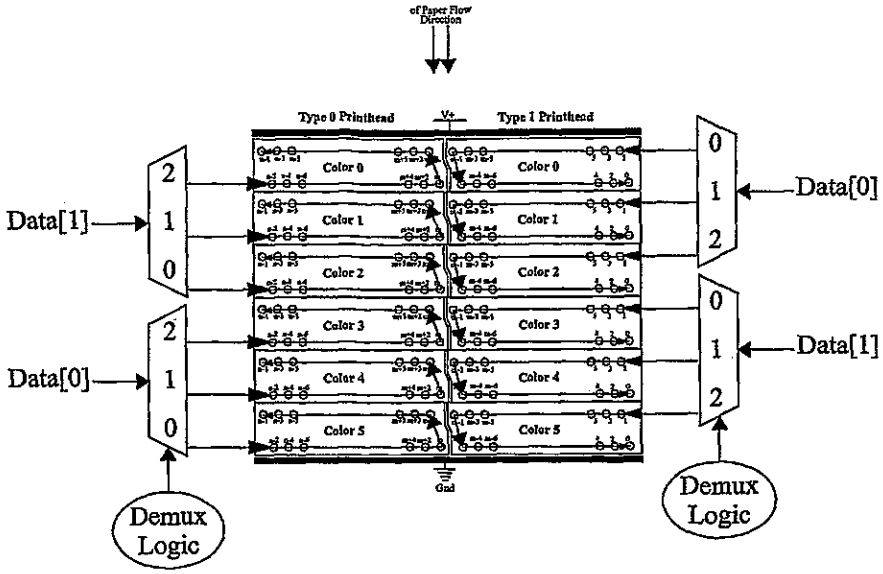


FIG. 304

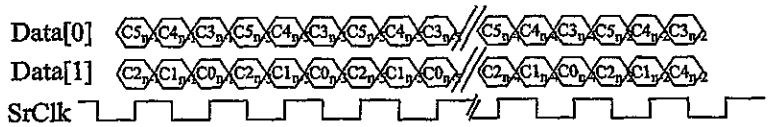


FIG. 305

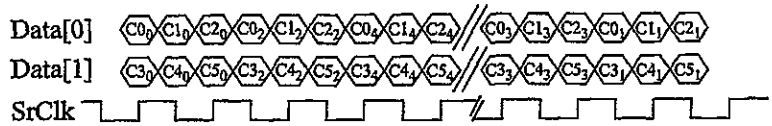


FIG. 306

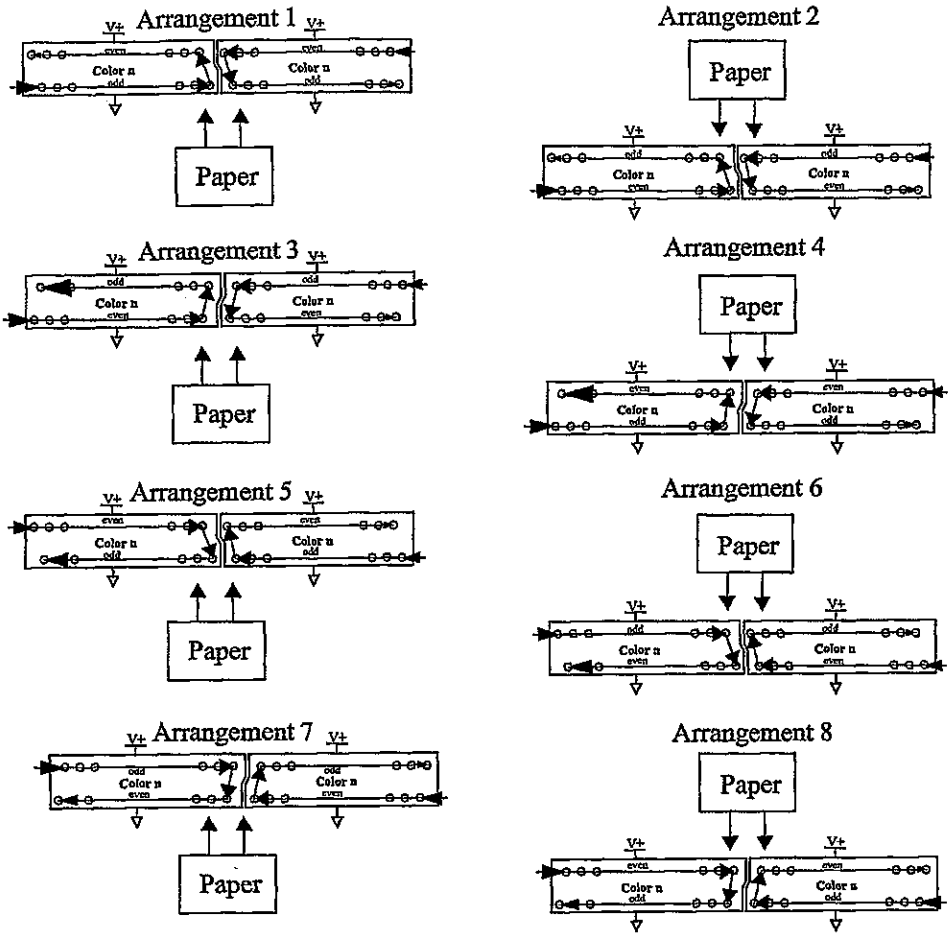

Arrangement 1
V+
V+
even
Color n
odd
even
Color n
odd
Paper
Arrangement 2
Paper
V+
V+
odd
Color n
even
odd
Color n
even
Arrangement 3
V+
V+
odd
Color n
even
odd
Color n
even
Paper
Arrangement 4
Paper
V+
V+
even
Color n
odd
even
Color n
odd
Arrangement 5
V+
V+
even
Color n
odd
even
Color n
odd
Paper
Arrangement 6
Paper
V+
V+
odd
Color n
even
odd
Color n
even
Arrangement 7
V+
V+
odd
Color n
even
odd
Color n
even
Paper
Arrangement 8
Paper
V+
V+
even
Color n
odd
even
Color n
odd

FIG. 307

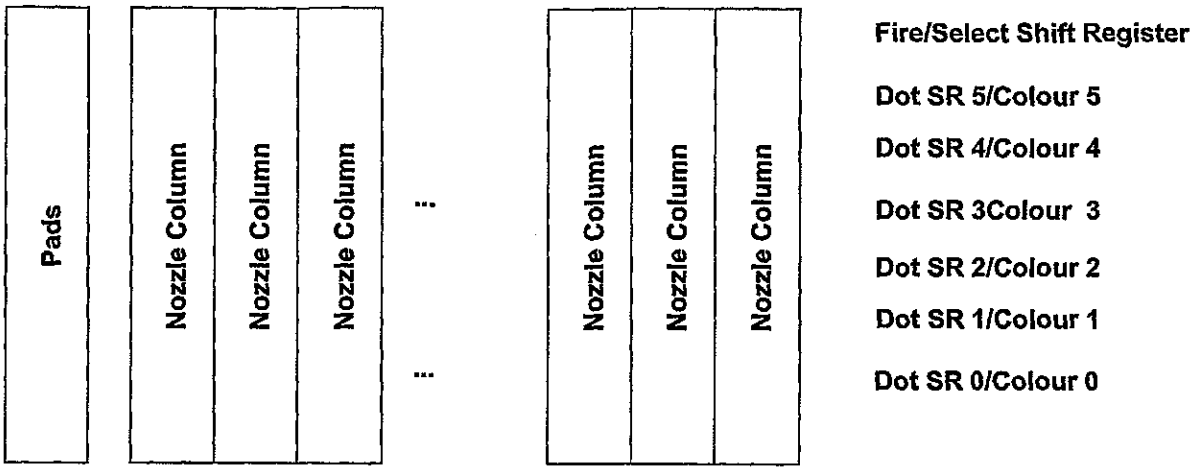

Pads
Nozzle Column
Nozzle Column
Nozzle Column
Nozzle Column
Nozzle Column
Nozzle Column
Fire/Select Shift Register
Dot SR 5/Colour 5
Dot SR 4/Colour 4
Dot SR 3Colour 3
Dot SR 2/Colour 2
Dot SR 1/Colour 1
Dot SR 0/Colour 0

FIG. 308

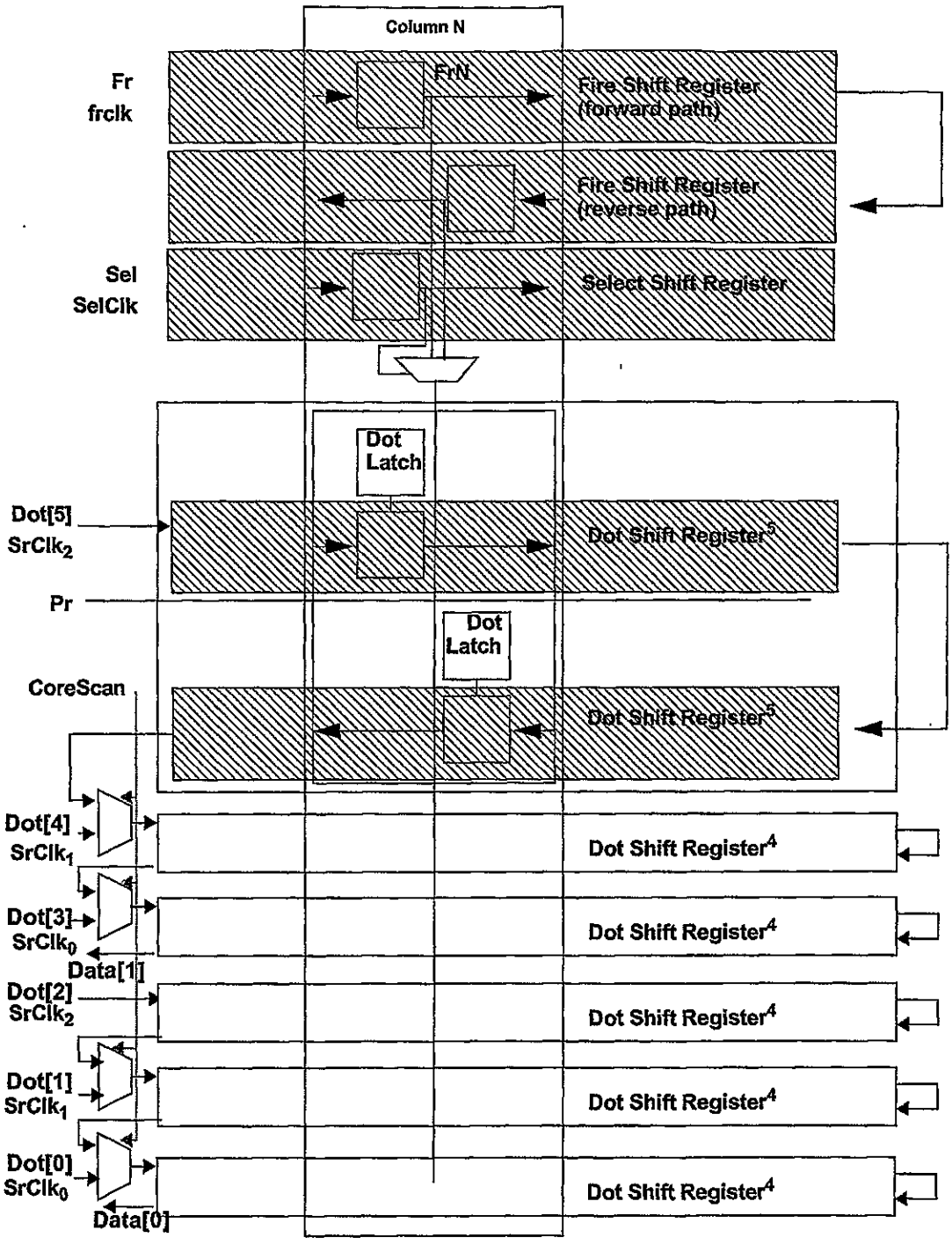

Column N
Fr
frclk
FrN
Fire Shift Register (forward path)
Fire Shift Register (reverse path)
Sel
SelClk
Select Shift Register
Dot Latch
Dot[5]
SrClk2
Dot Shift Register5
Pr
Dot Latch
CoreScan
Dot Shift Register5
Dot[4]
SrClk1
Dot Shift Register4
Dot[3]
SrClk0
Dot Shift Register4
Data[1]
Dot[2]
SrClk2
Dot Shift Register4
Dot[1]
SrClk1
Dot Shift Register4
Dot[0]
SrClk0
Dot Shift Register4
Data[0]

FIG. 309

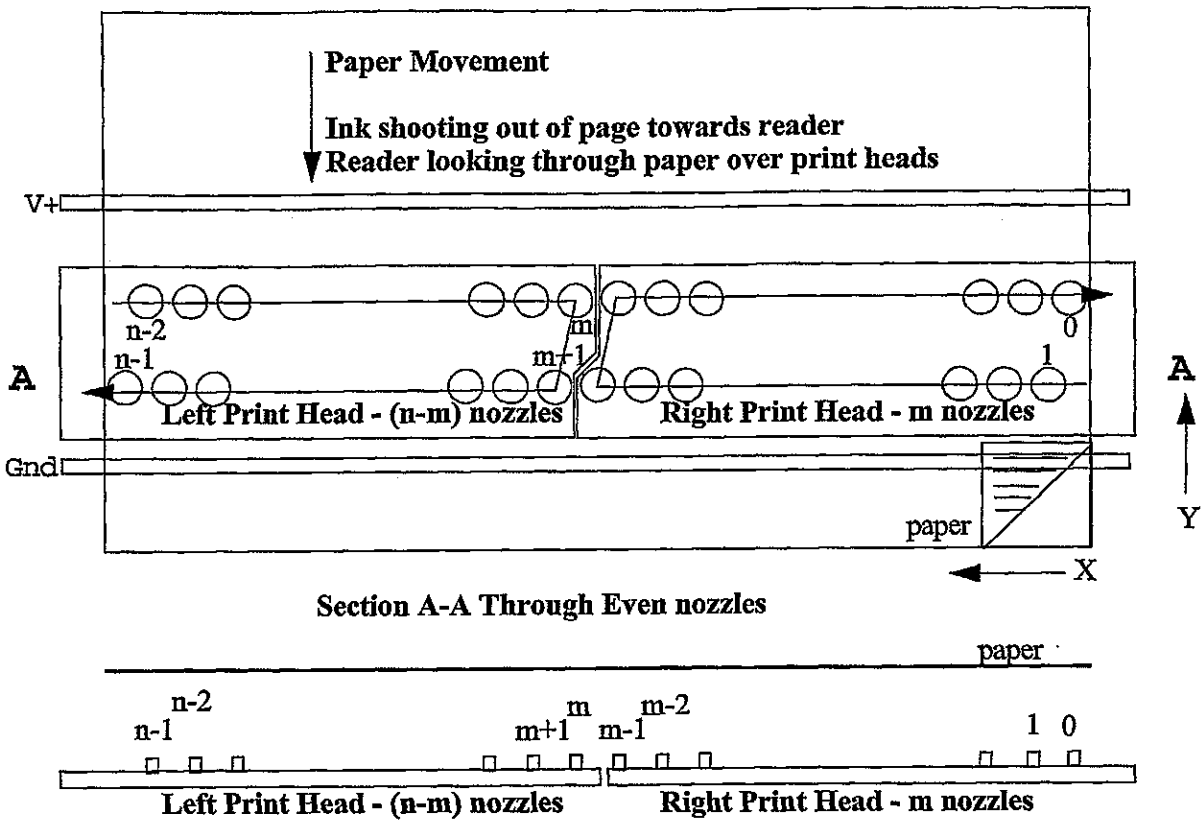

Paper Movement
Ink shooting out of page towards reader
Reader looking through paper over print heads
V+
Gnd
A
A
n-2
n-1
m
m+1
0
1
Left Print Head - (n-m) nozzles
Right Print Head - m nozzles
paper
X
Y
Section A-A Through Even nozzles
paper
n-2
n-1
m
m+1
m-2
m-1
1 0
Left Print Head - (n-m) nozzles
Right Print Head - m nozzles

FIG. 310

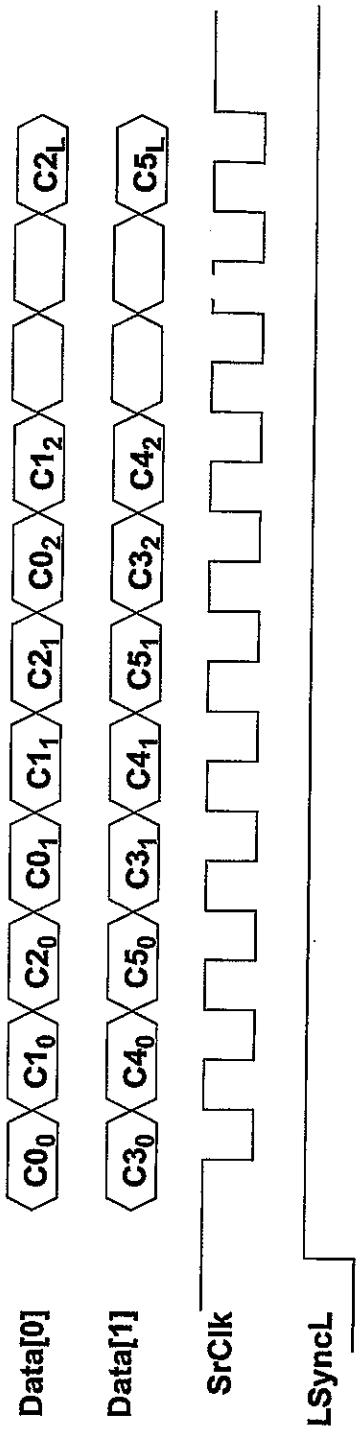
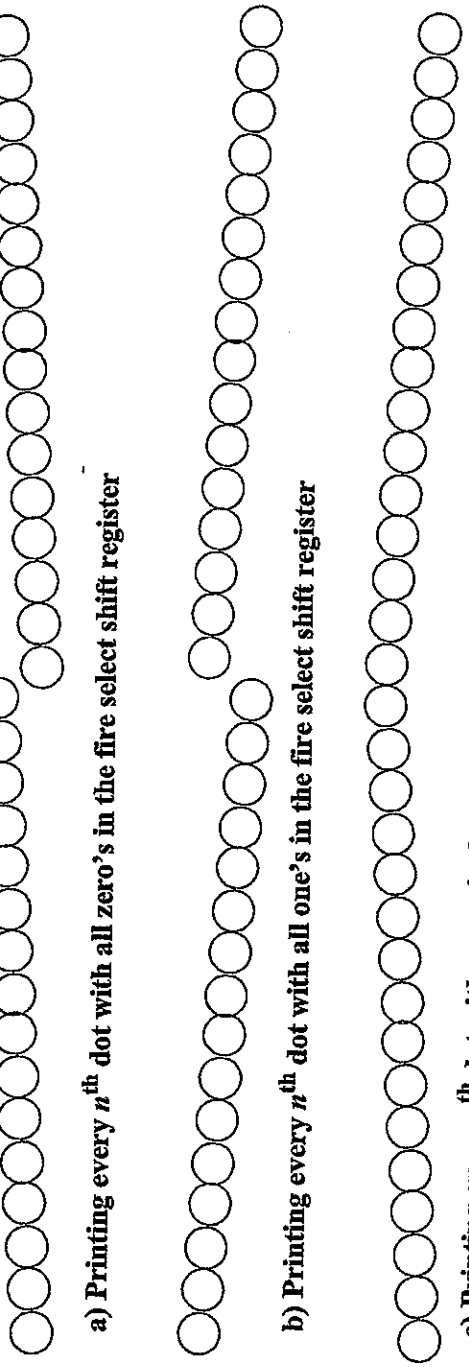

Data[0]
Data[1]
SrClk
LSyncL
C0$_0$ C1$_0$ C2$_0$ C0$_1$ C1$_1$ C2$_1$ C0$_2$ C1$_2$ C2$_L$
C3$_0$ C4$_0$ C5$_0$ C3$_1$ C4$_1$ C5$_1$ C3$_2$ C4$_2$ C5$_L$

FIG. 311



a) Printing every n^{th} dot with all zero's in the fire select shift register

b) Printing every n^{th} dot with all one's in the fire select shift register

c) Printing every n^{th} dot with n zero's then n one's in the fire select shift registers

FIG. 312

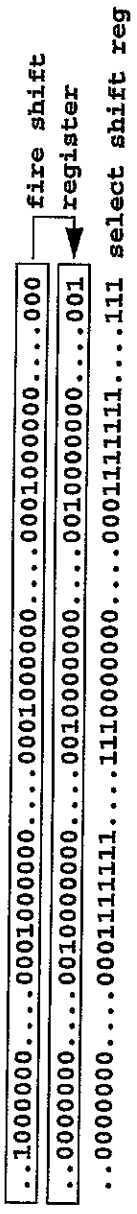


FIG. 313

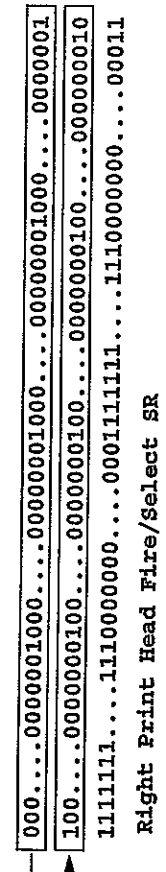


FIG. 314

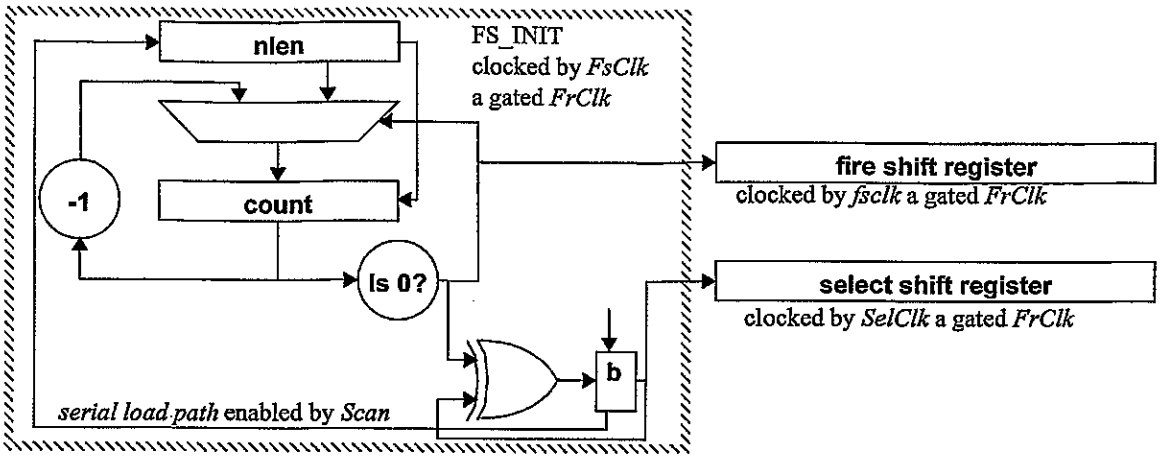


FIG. 315

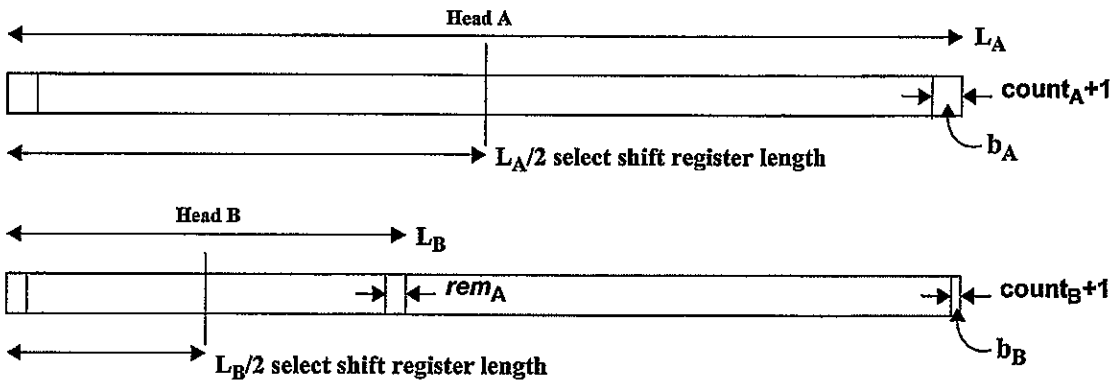


FIG. 316

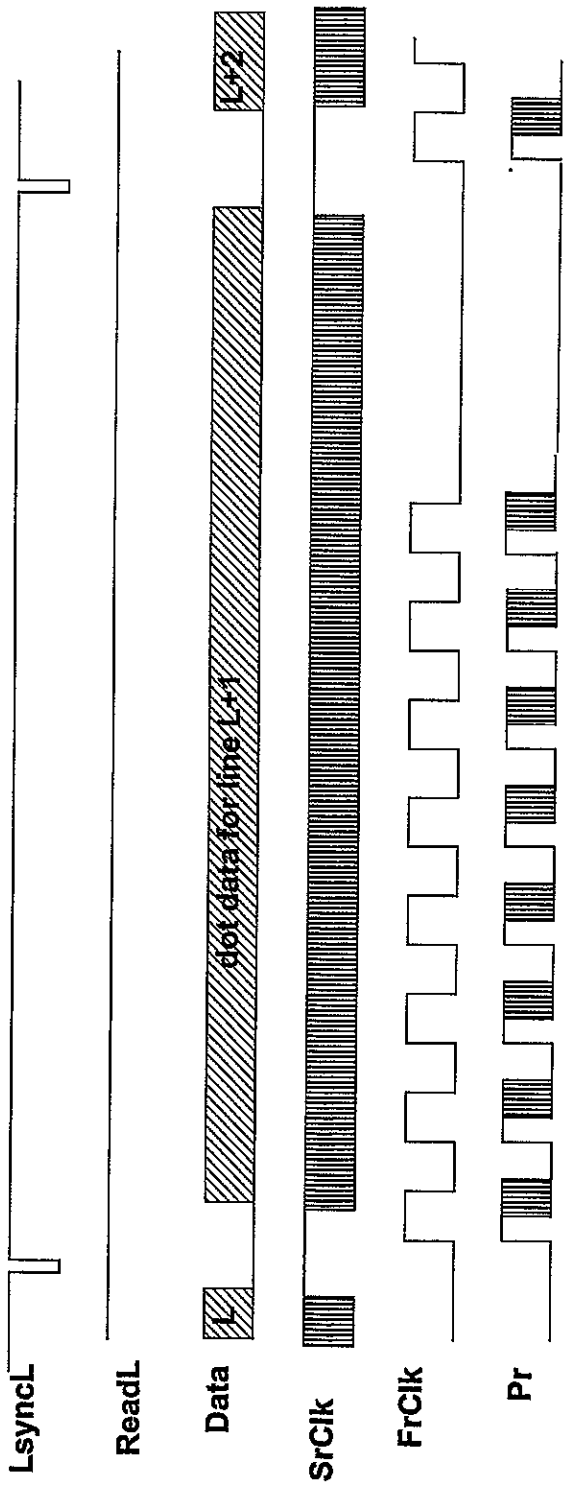


FIG. 317

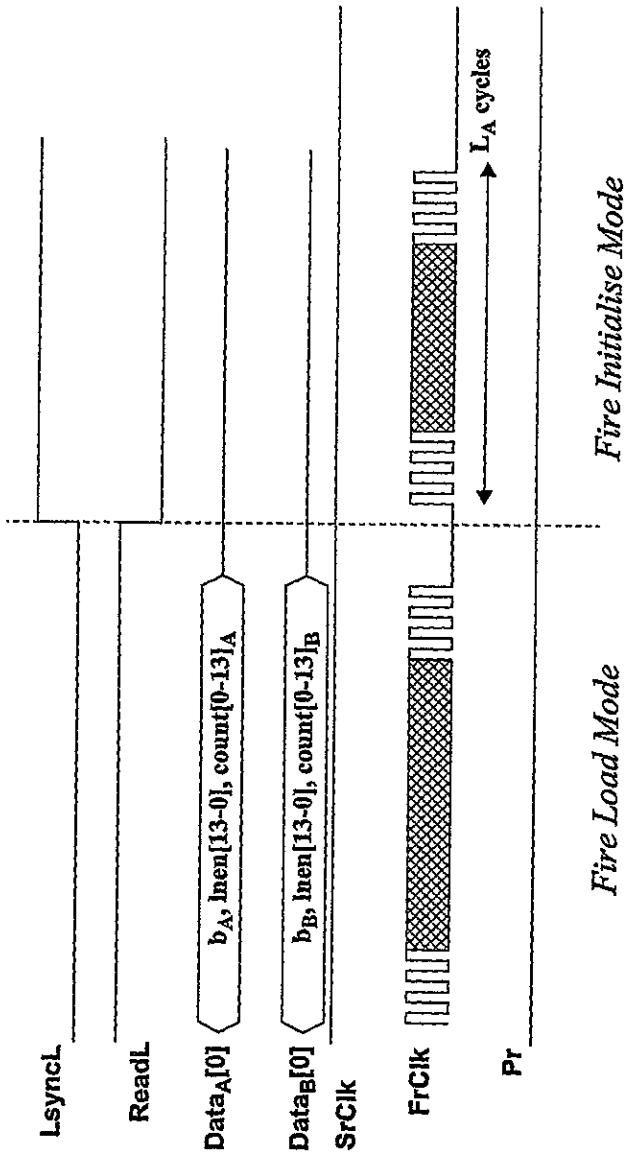

LsyncL
ReadL
Data$_A$[0]
Data$_B$[0]
SrClk
FrClk
Pr
b_A, lnen[13-0], count[0-13]$_A$
b_B, lnen[13-0], count[0-13]$_B$
L_A cycles
Fire Load Mode
Fire Initialise Mode

FIG. 318

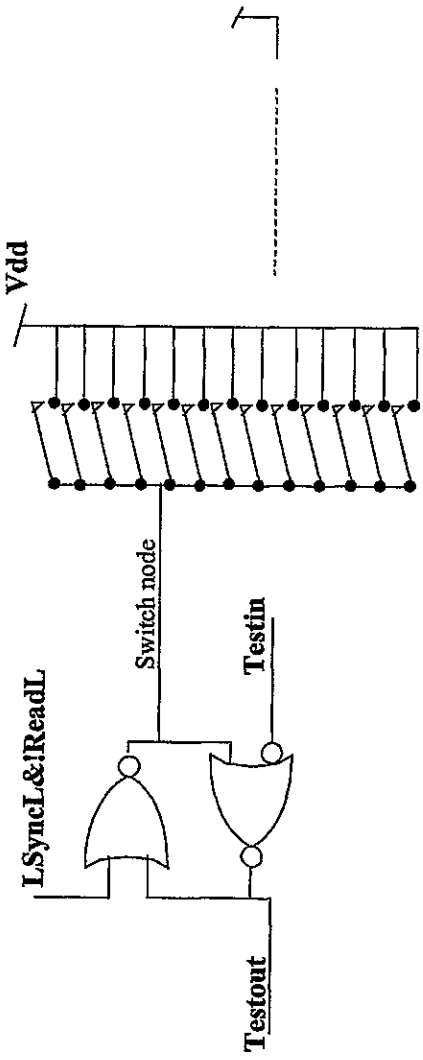


FIG. 319

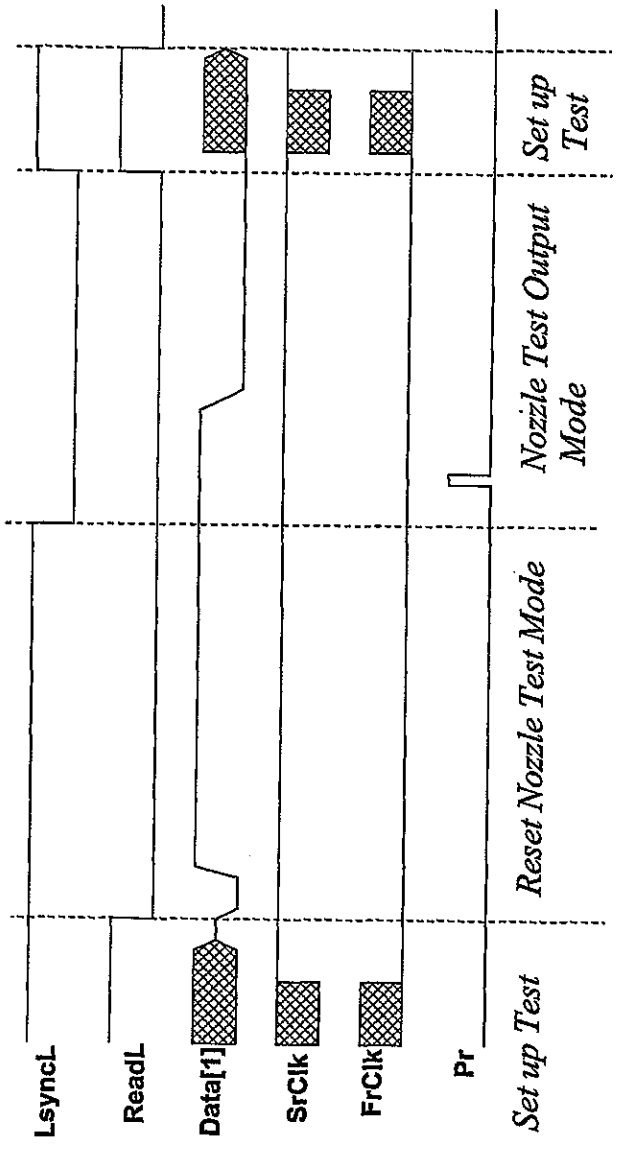

LsyncL
ReadL
Data[1]
SrClk
FrClk
Pr
Set up Test
Reset Nozzle Test Mode
Nozzle Test Output Mode
Set up Test

FIG. 320

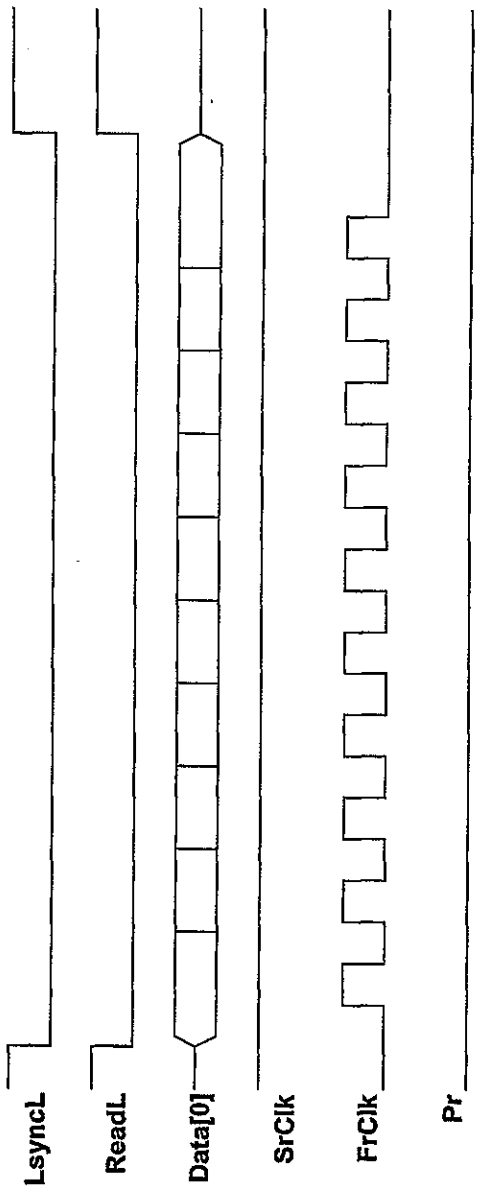


FIG. 321

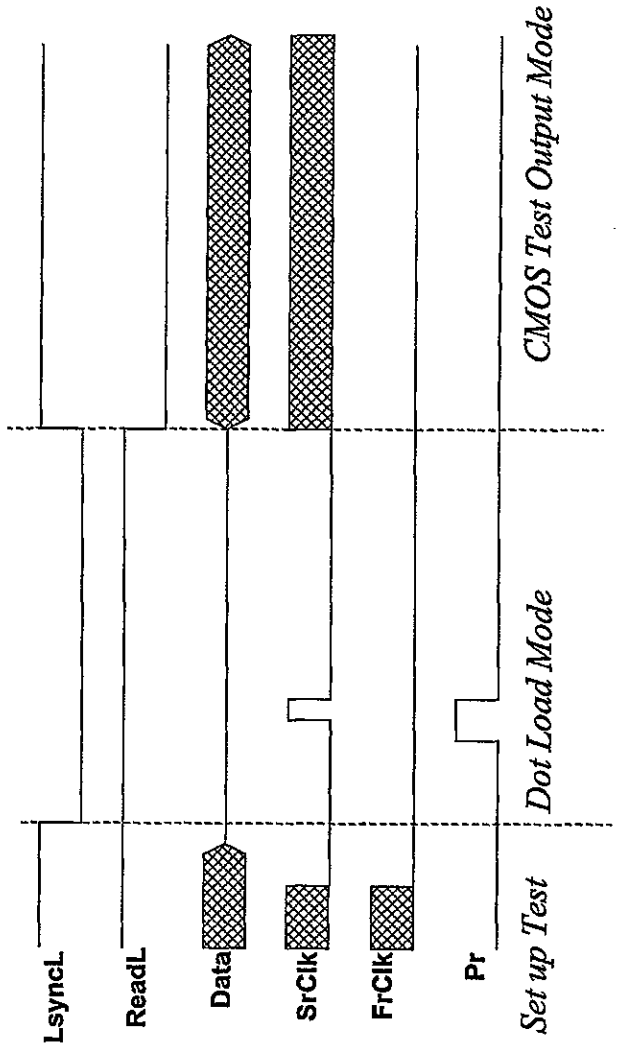

LsyncL
ReadL
Data
SrClk
FrClk
Pr
Set up Test
Dot Load Mode
CMOS Test Output Mode

FIG. 322

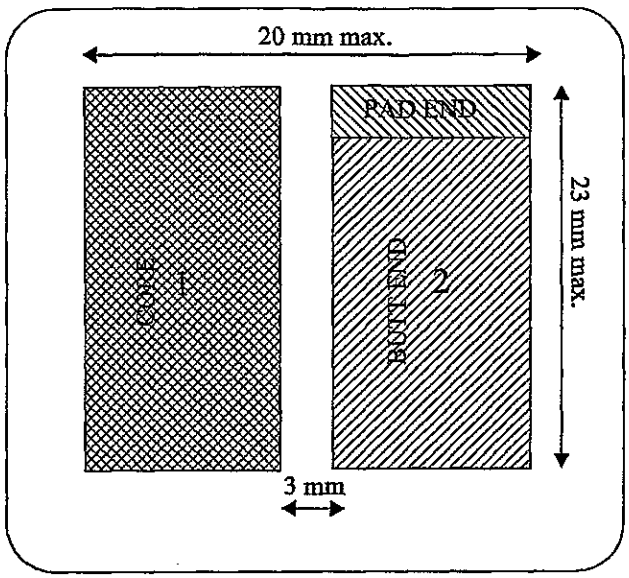


FIG. 323

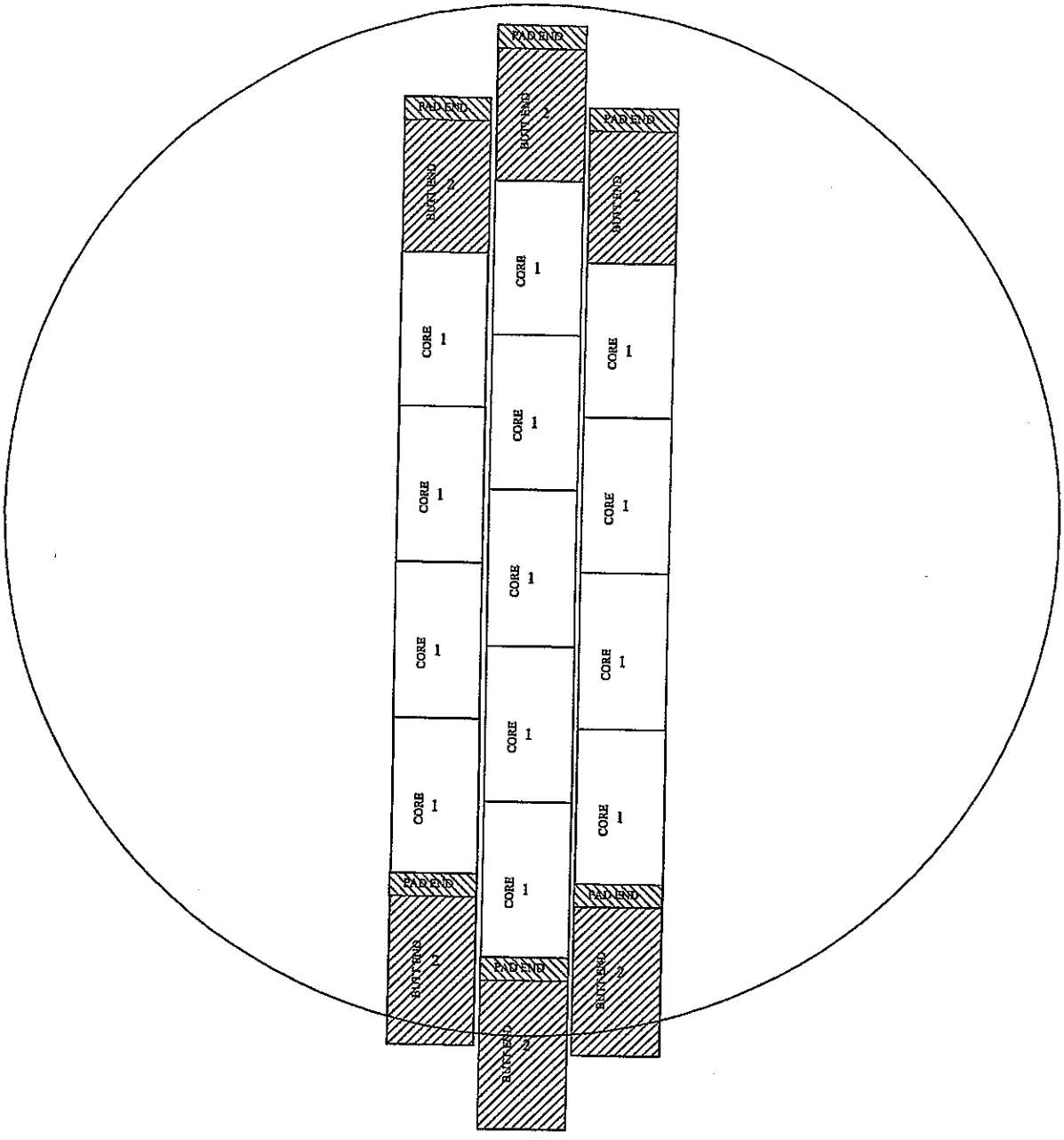


FIG. 324

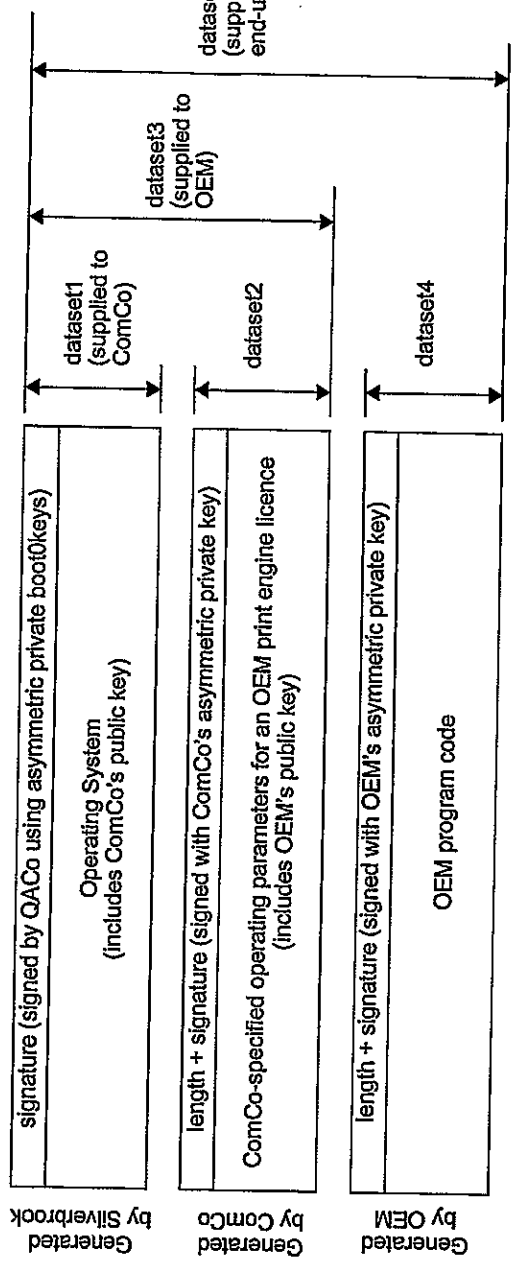


FIG. 325

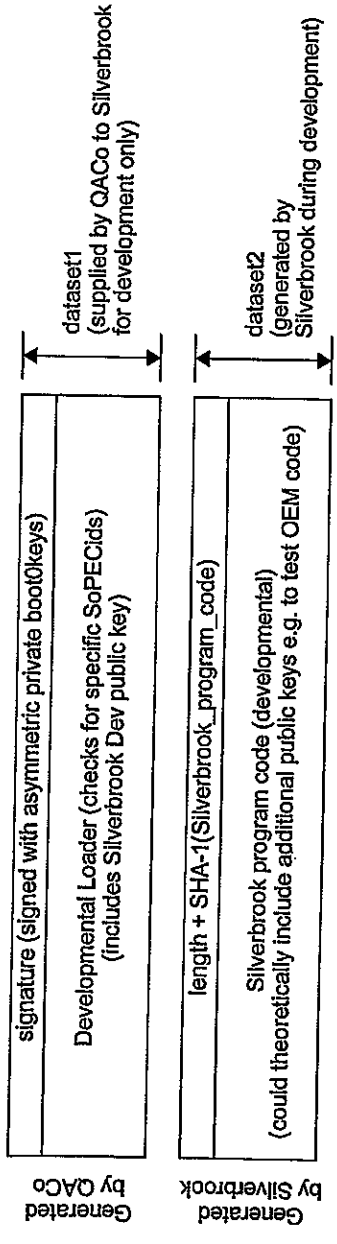


FIG. 327

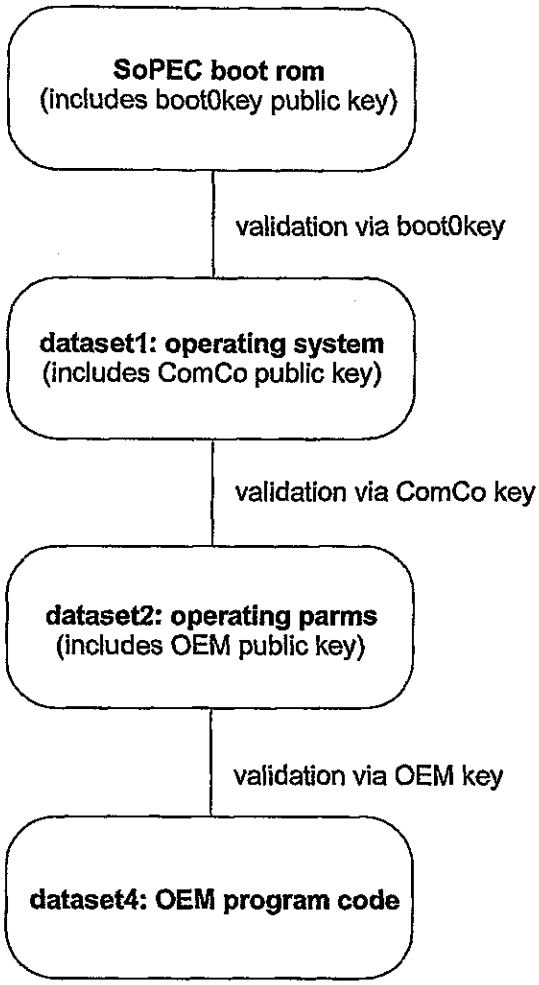

SoPEC boot rom
(includes boot0key public key)
validation via boot0key
dataset1: operating system
(includes ComCo public key)
validation via ComCo key
dataset2: operating parms
(includes OEM public key)
validation via OEM key
dataset4: OEM program code

FIG. 326

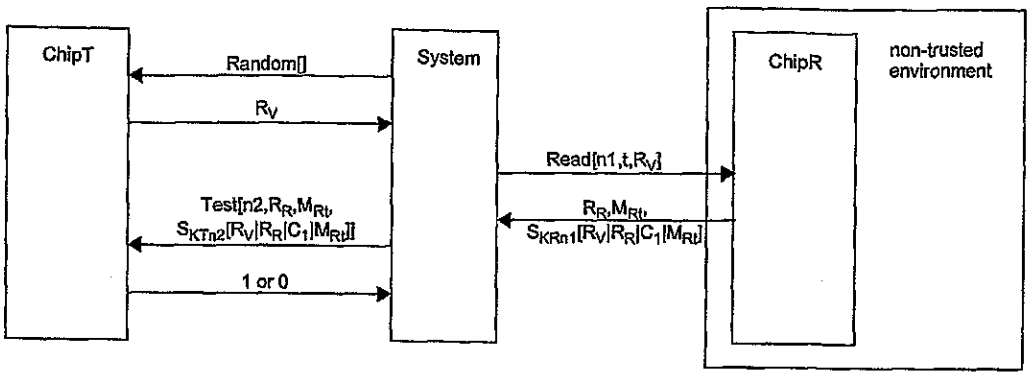


FIG. 328

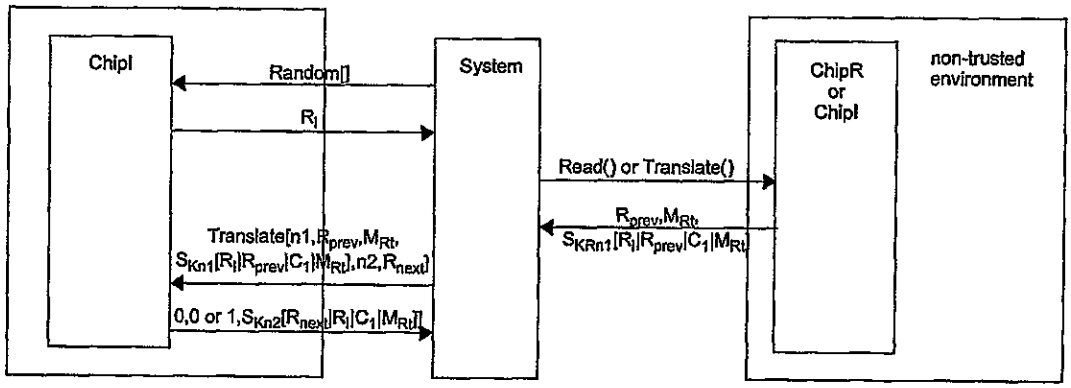


FIG. 329

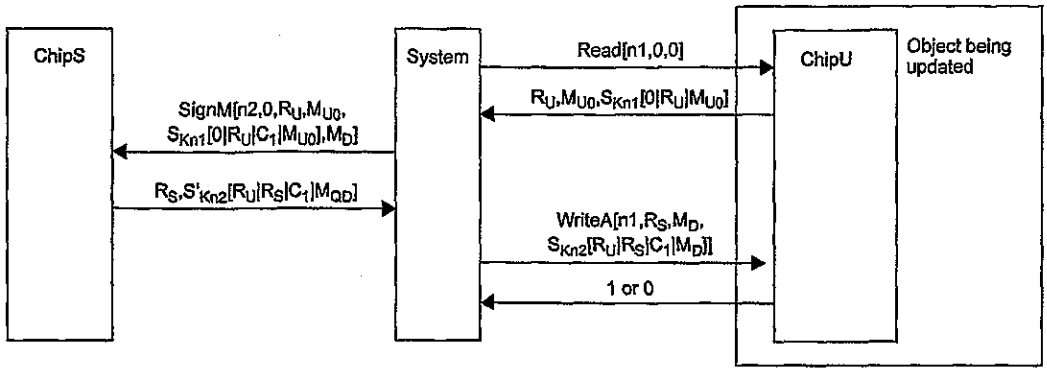


FIG. 330

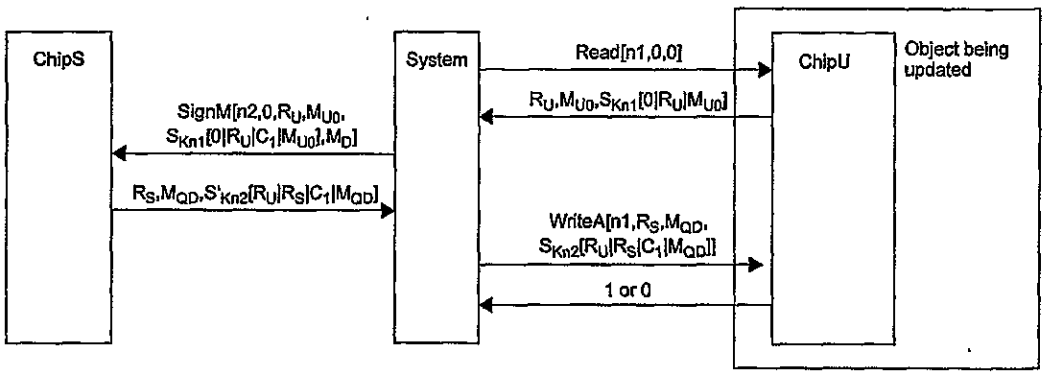


FIG. 331

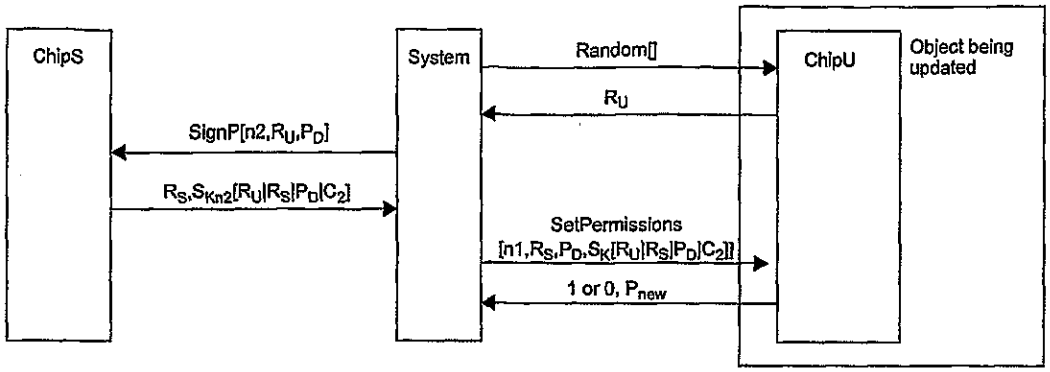


FIG. 332

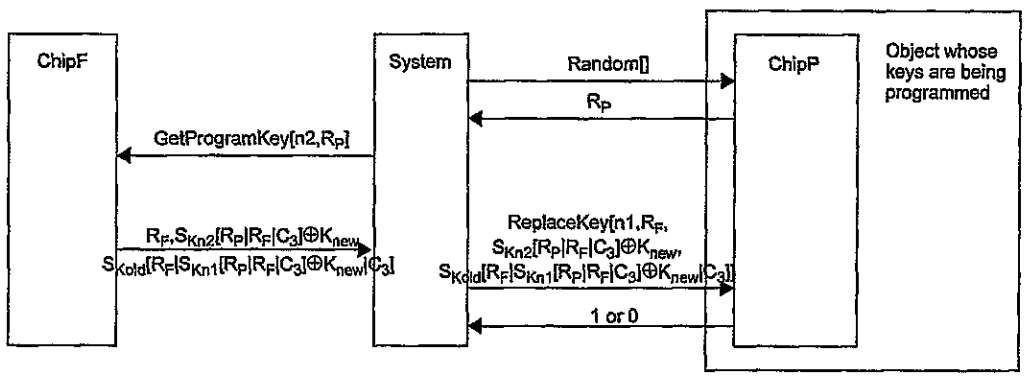


FIG. 333

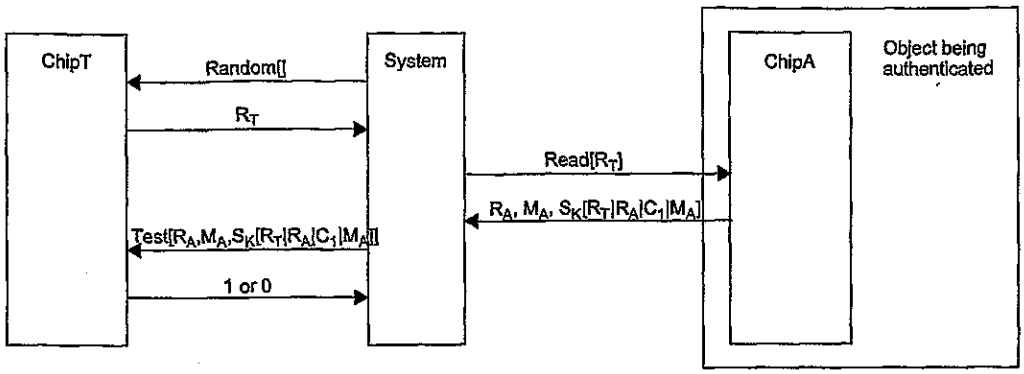


FIG. 334

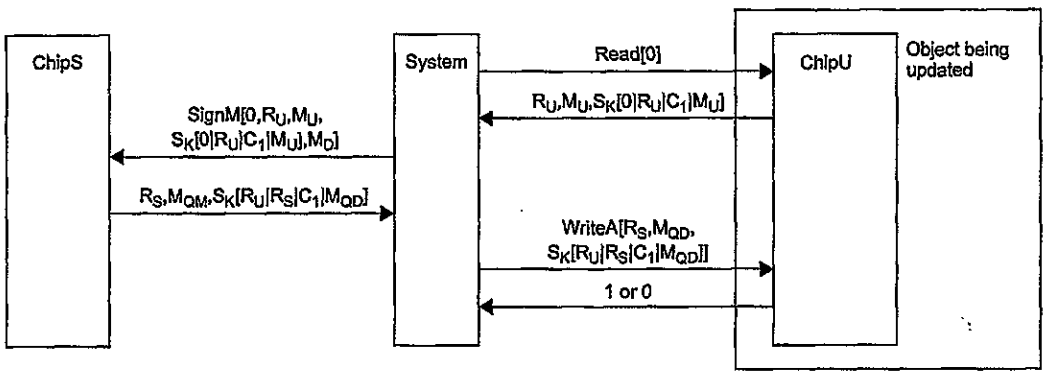


FIG. 335

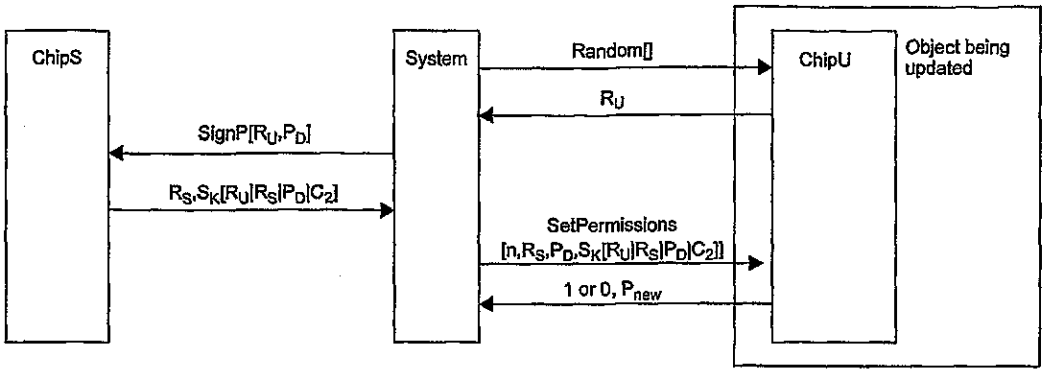


FIG. 336

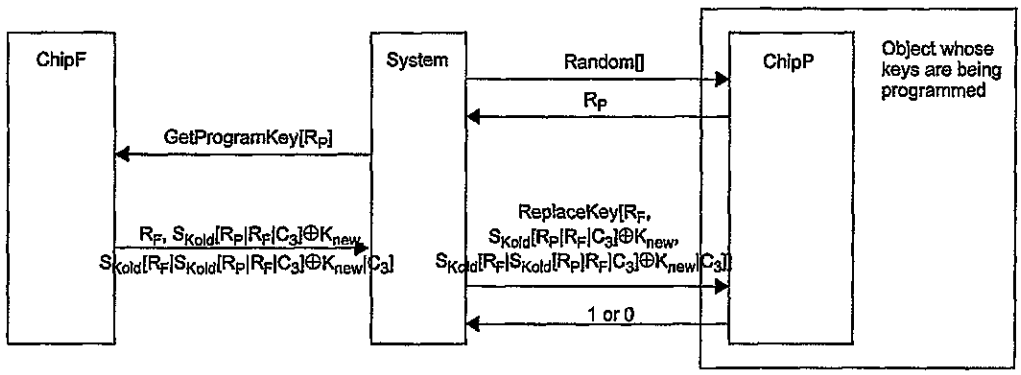


FIG. 337

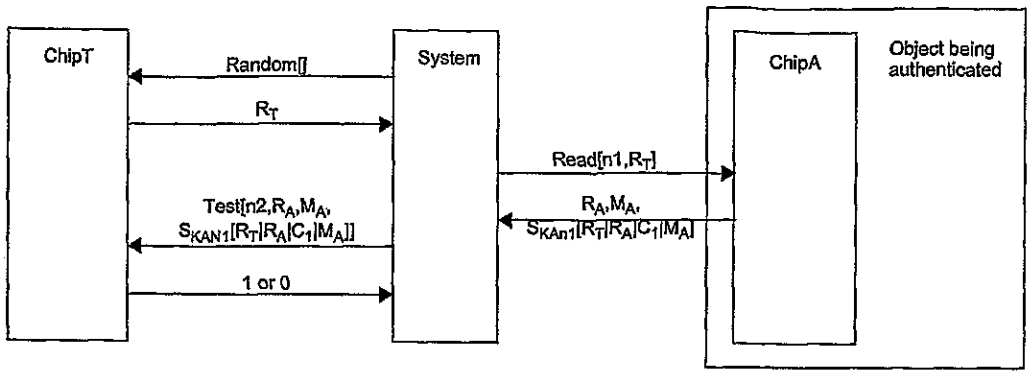


FIG. 338

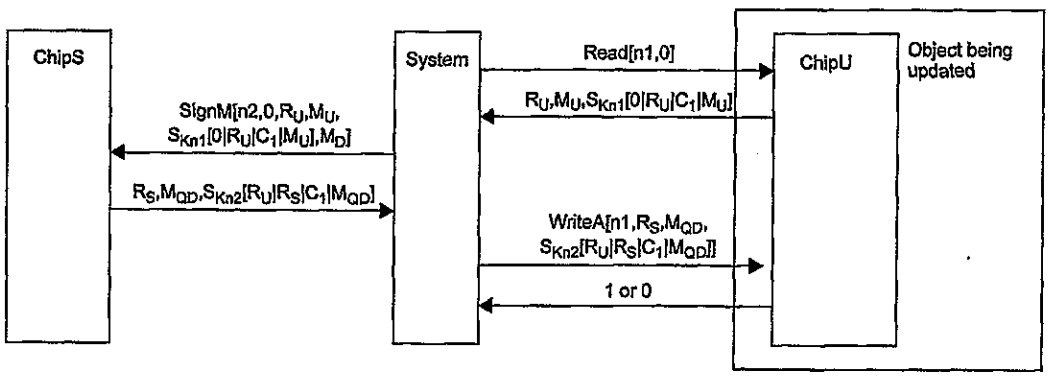


FIG. 339

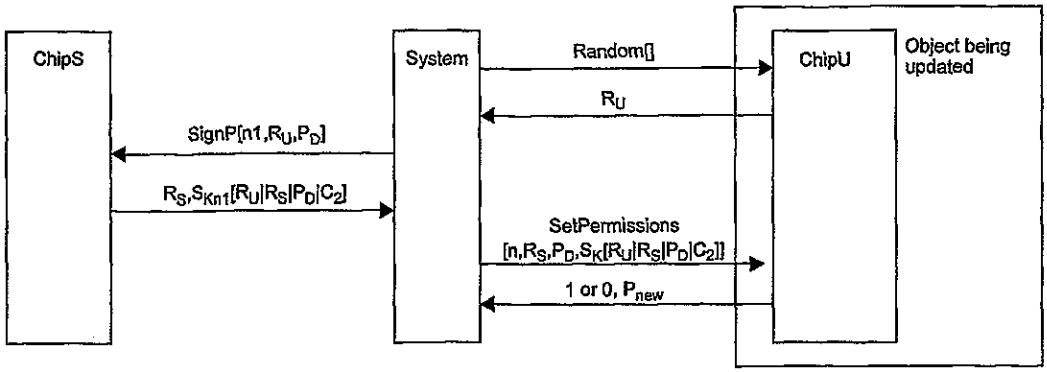


FIG. 340

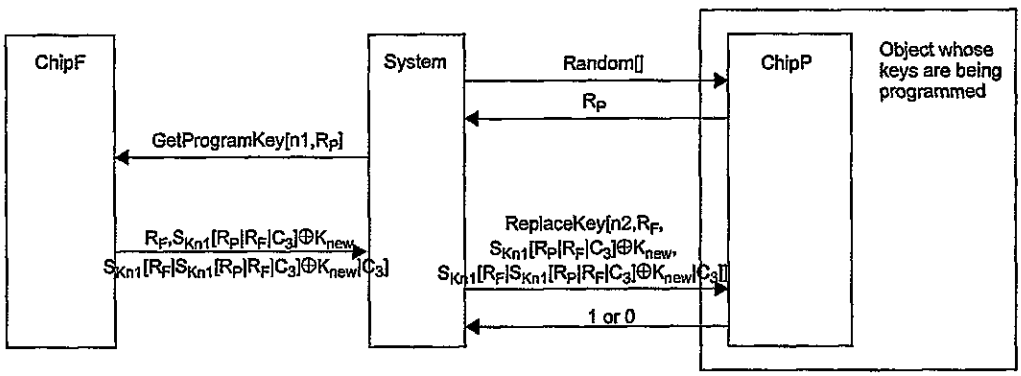


FIG. 341

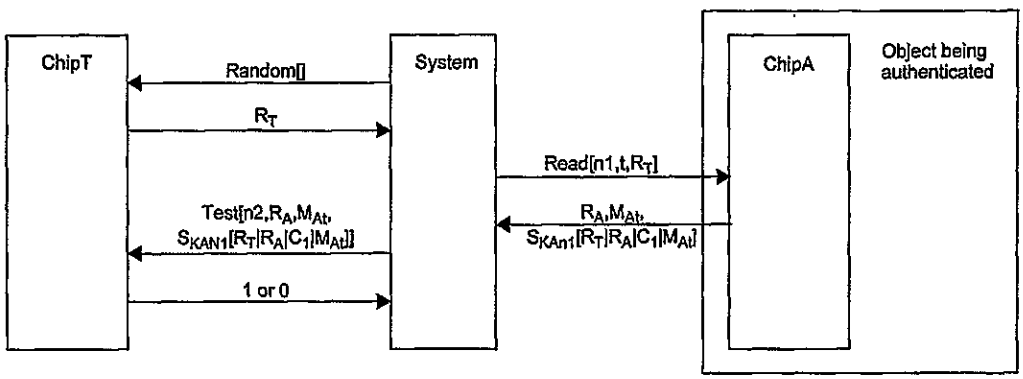


FIG. 342

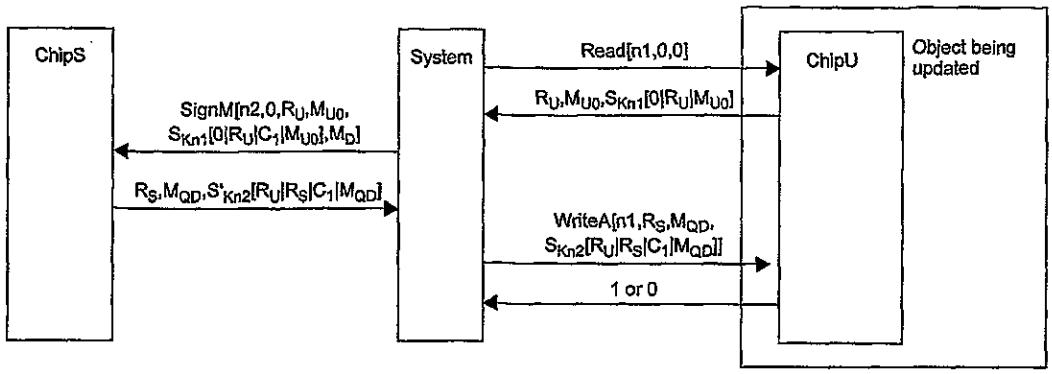


FIG. 343

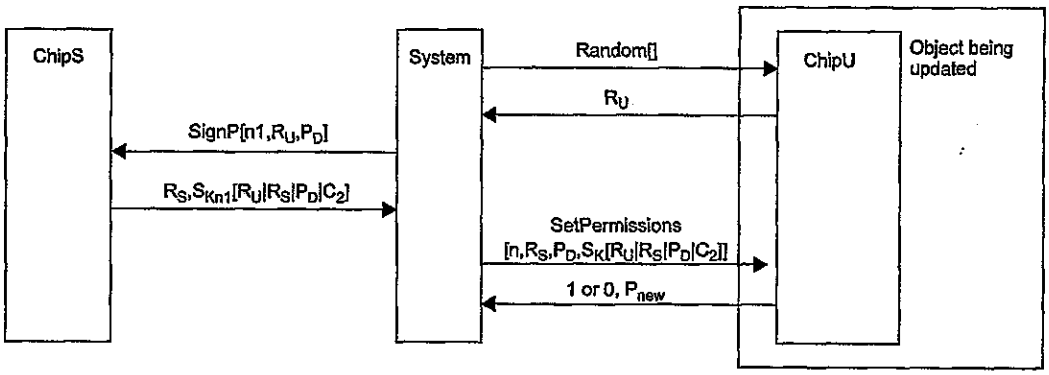


FIG. 344

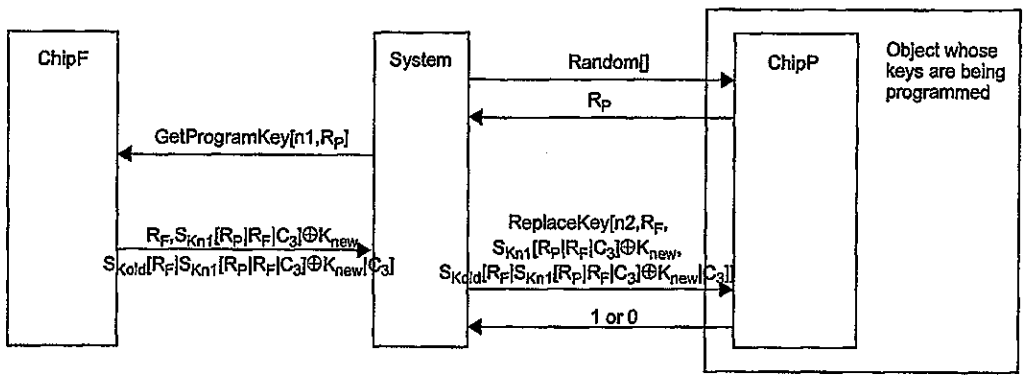


FIG. 345

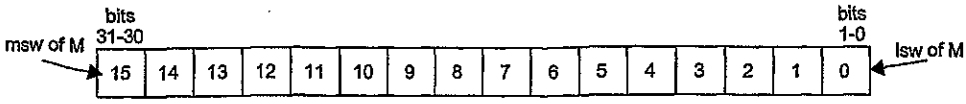


FIG. 346

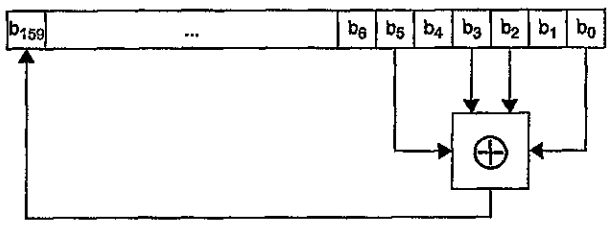


FIG. 347

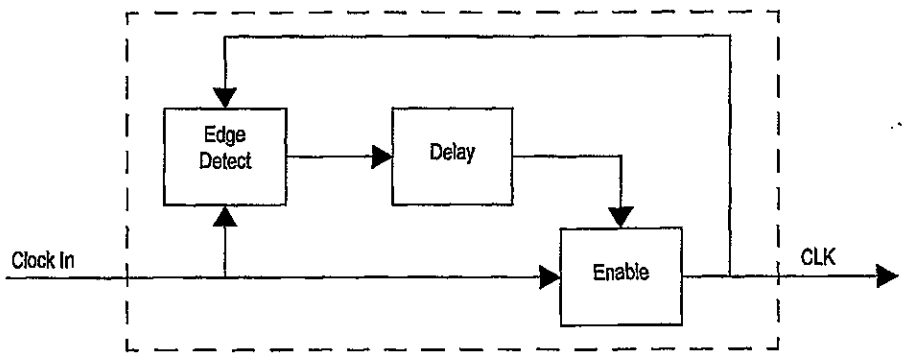


FIG. 348

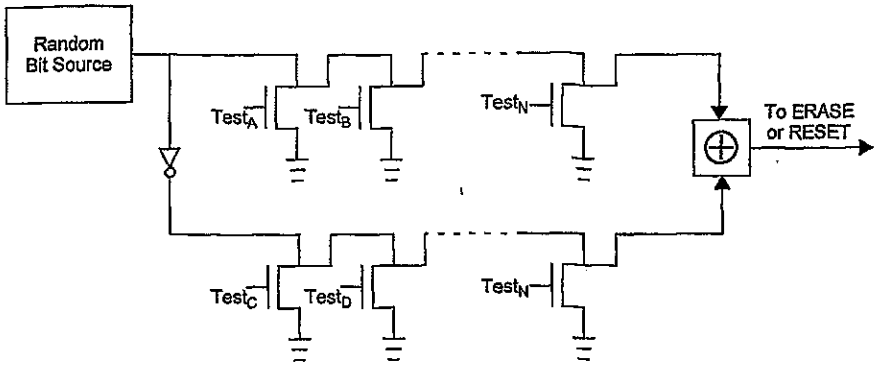


FIG. 349

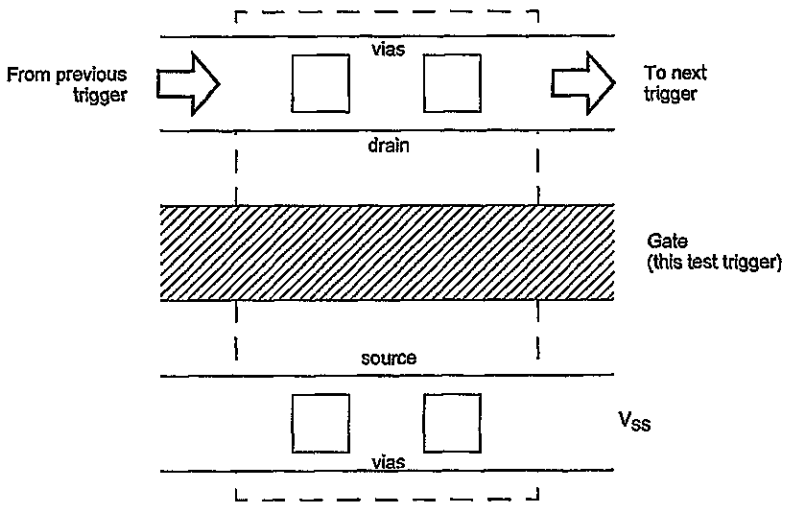


FIG. 350

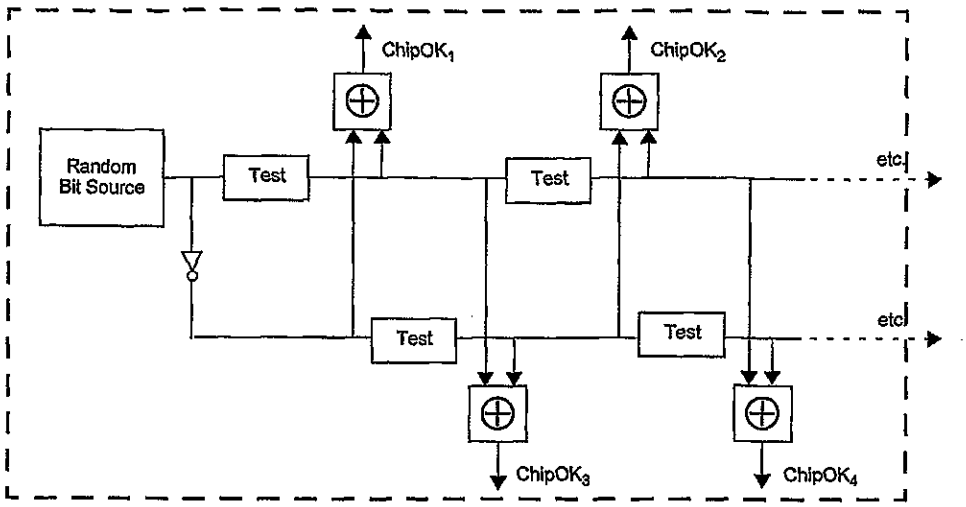


FIG. 351

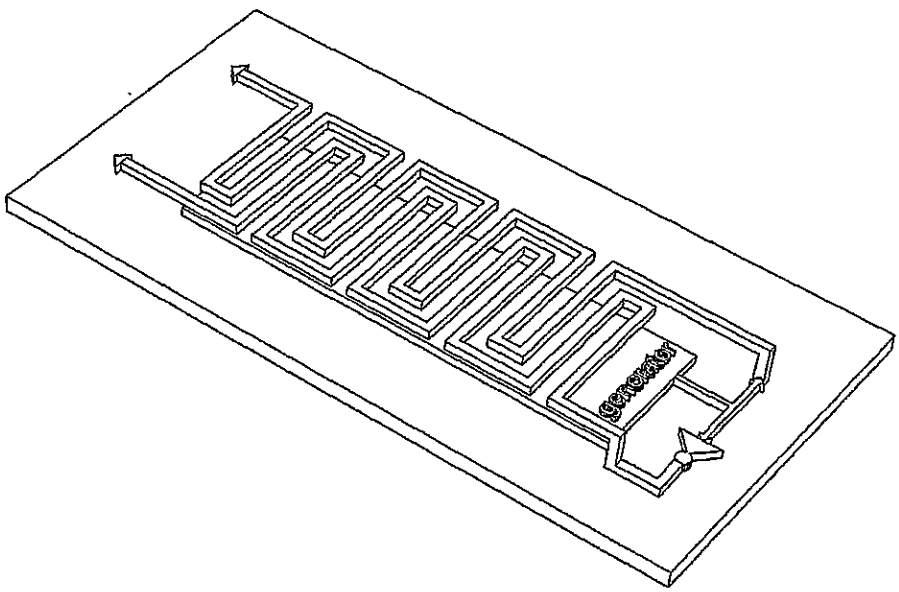


FIG. 352

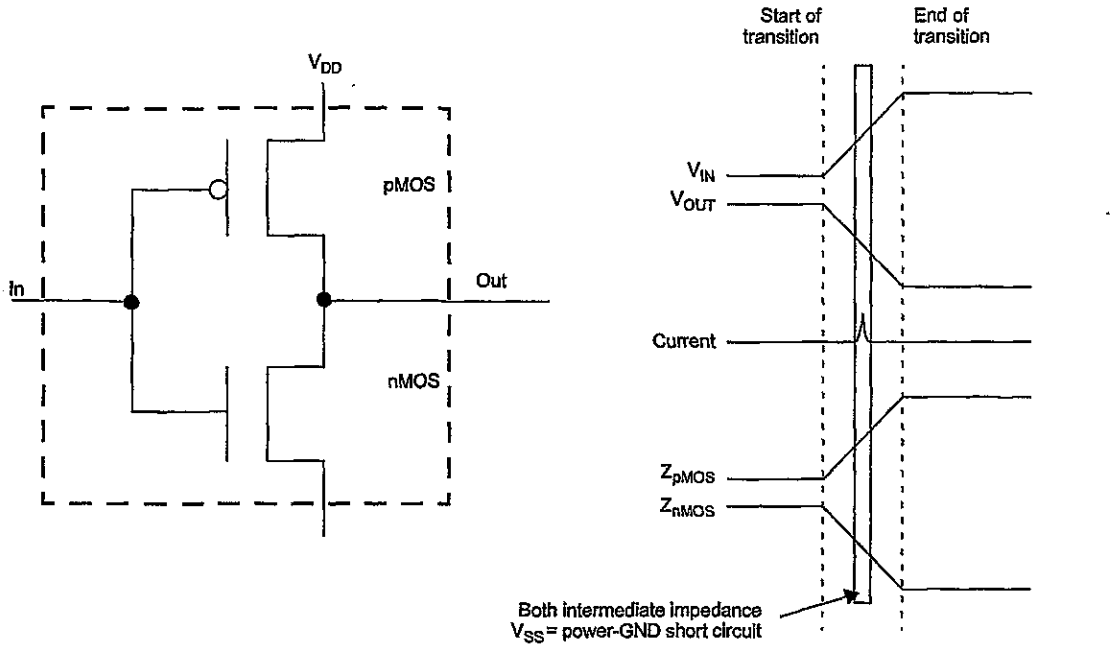


FIG. 353

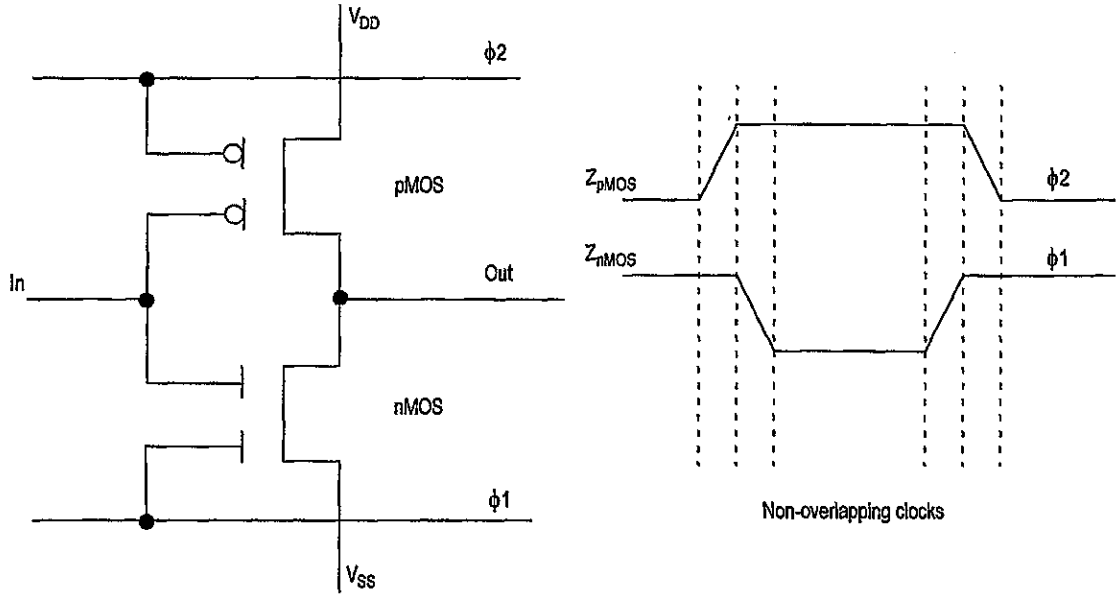


FIG. 354

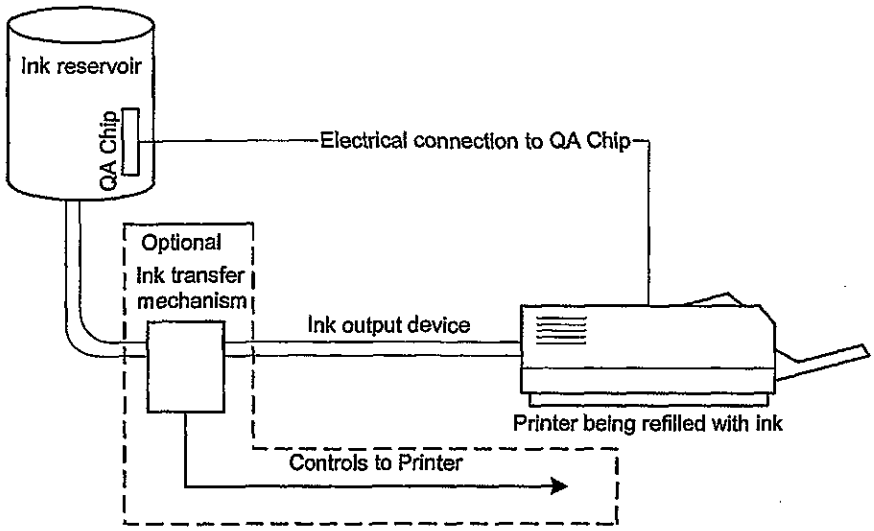


FIG. 355

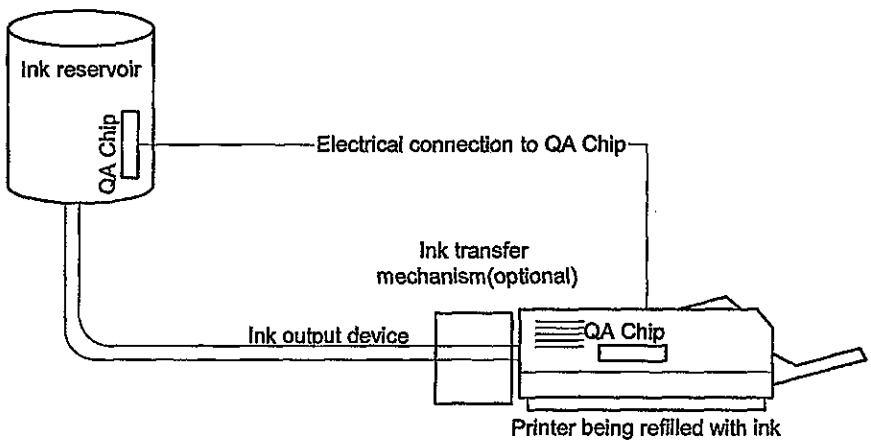


FIG. 356

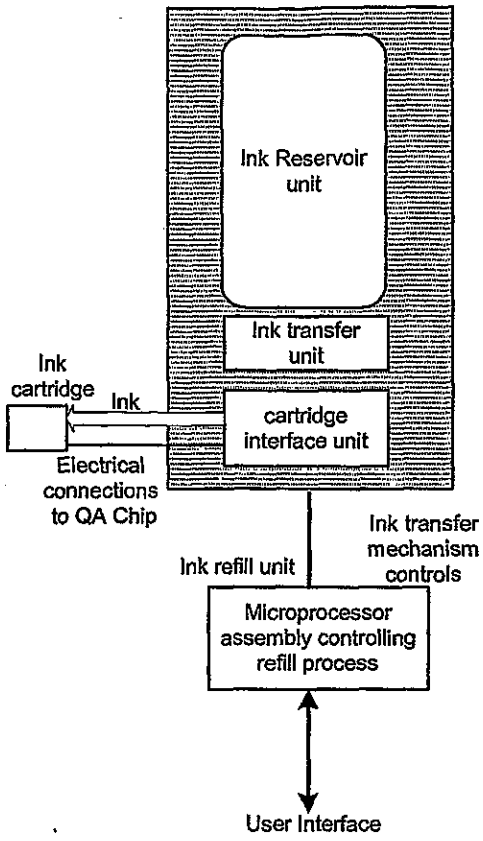


FIG. 357

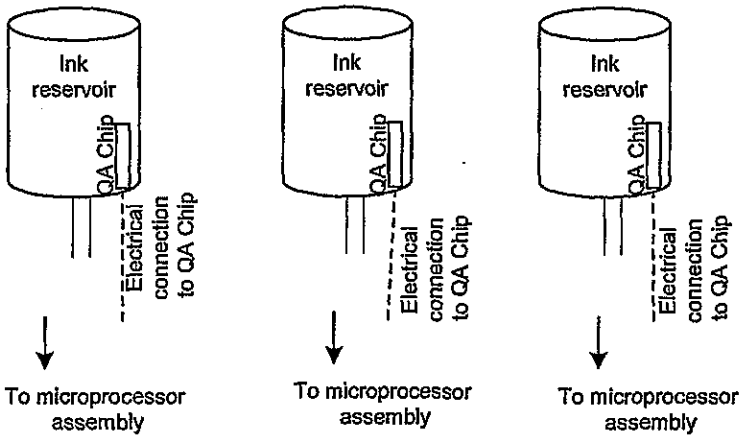


FIG. 358

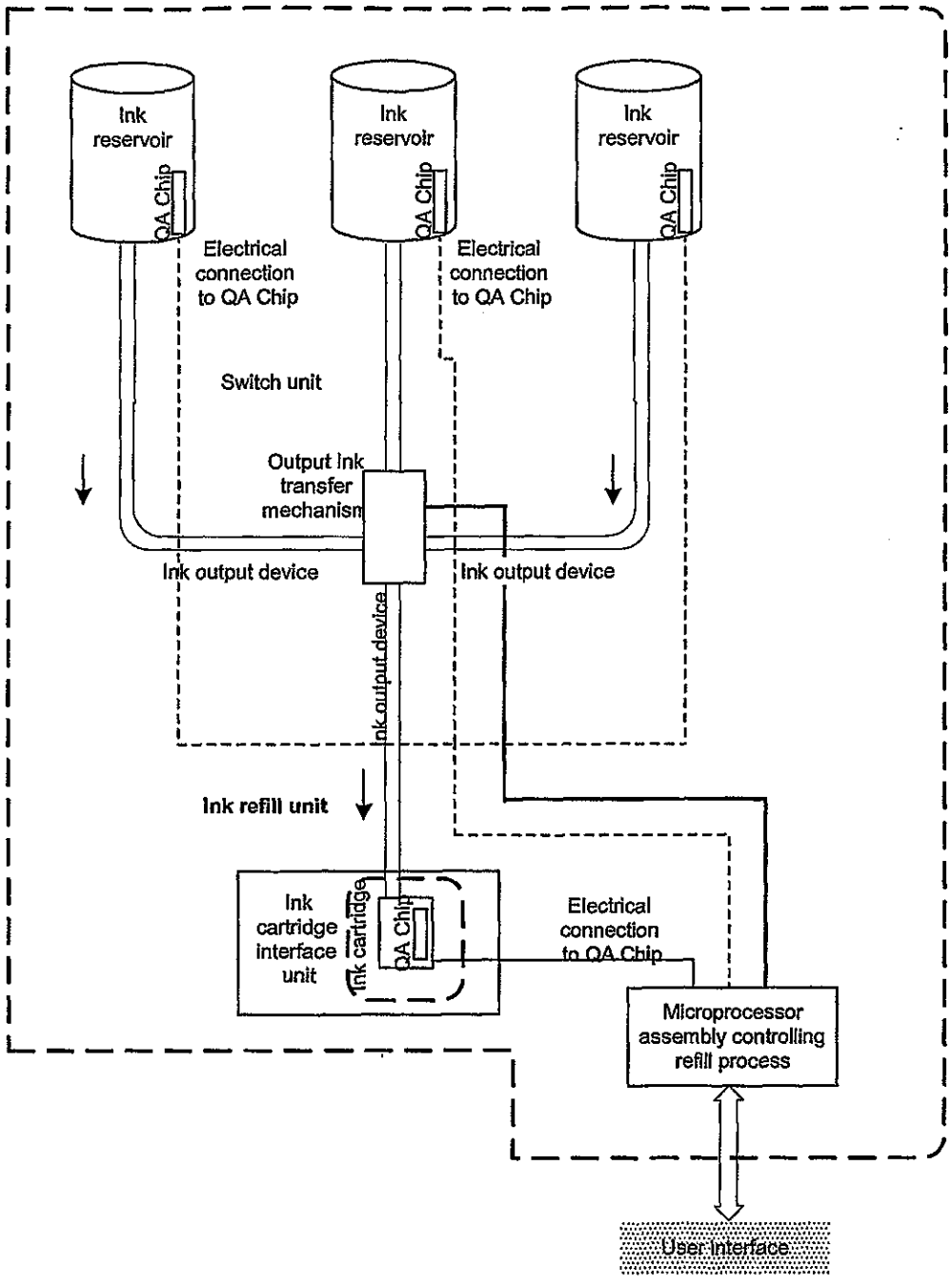

Ink reservoir
QA Chip
Ink reservoir
QA Chip
Ink reservoir
QA Chip
Electrical connection to QA Chip
Electrical connection to QA Chip
Switch unit
Output Ink transfer mechanism
Ink output device
Ink output device
Ink output device
Ink refill unit
Ink cartridge interface unit
Ink cartridge
QA Chip
Electrical connection to QA Chip
Microprocessor assembly controlling refill process
User Interface

FIG. 359

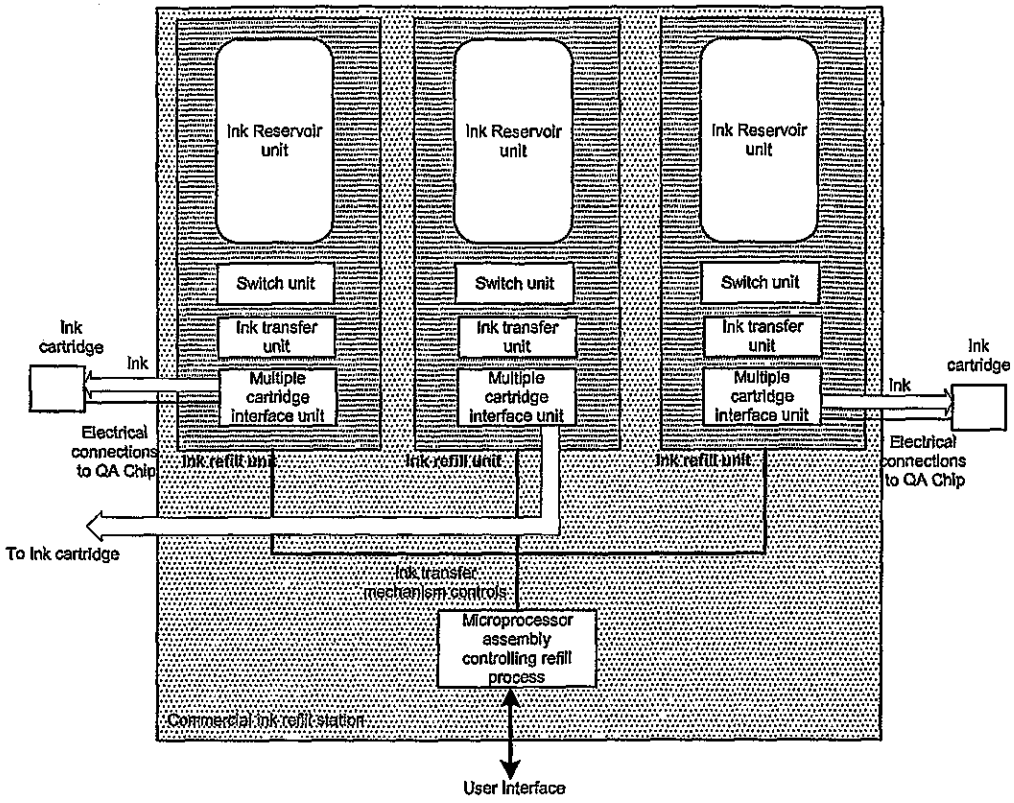


FIG. 360

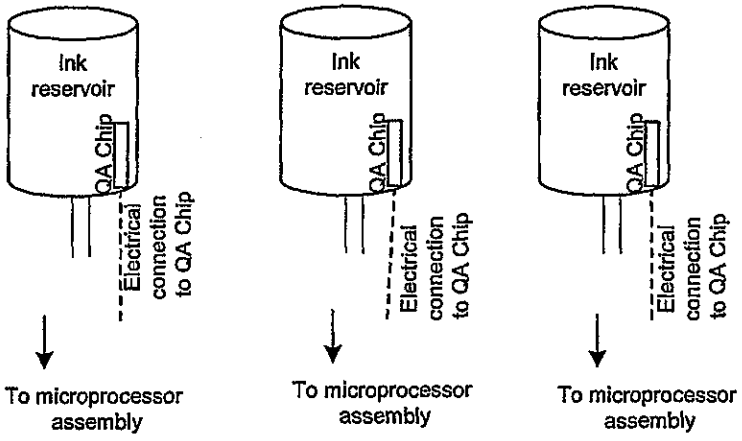


FIG. 361

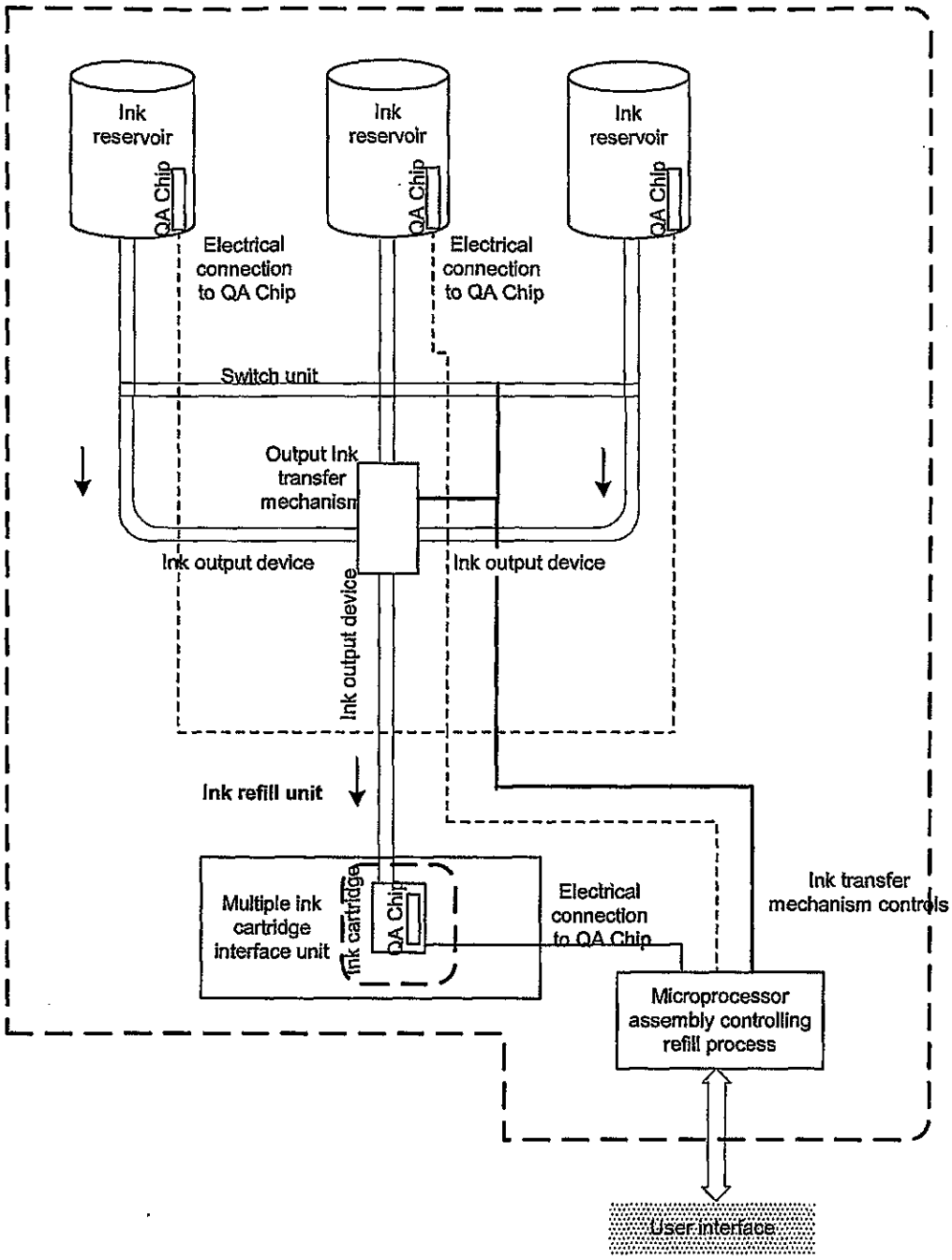

Ink reservoir
QA Chip
Ink reservoir
QA Chip
Ink reservoir
QA Chip
Electrical connection to QA Chip
Electrical connection to QA Chip
Switch unit
Output Ink transfer mechanism
Ink output device
Ink output device
Ink output device
Ink refill unit
Multiple ink cartridge interface unit
Ink cartridge
QA Chip
Electrical connection to QA Chip
Ink transfer mechanism controls
Microprocessor assembly controlling refill process
User interface

FIG. 362

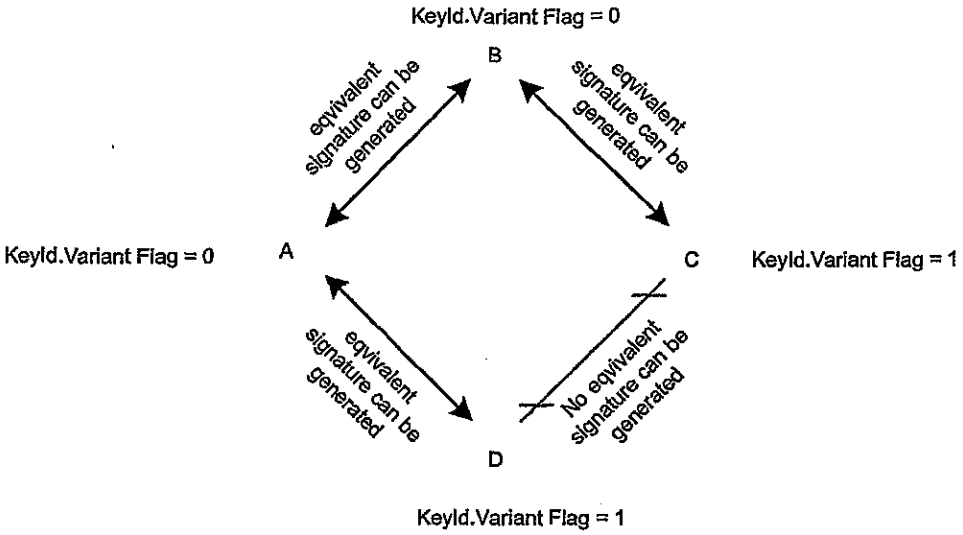


FIG. 363

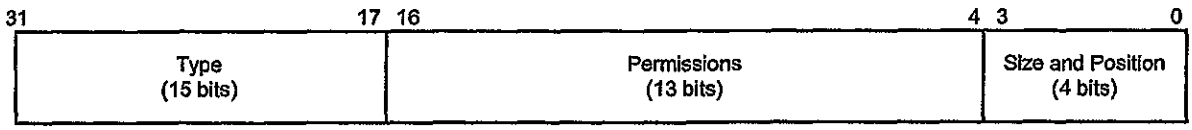


FIG. 364

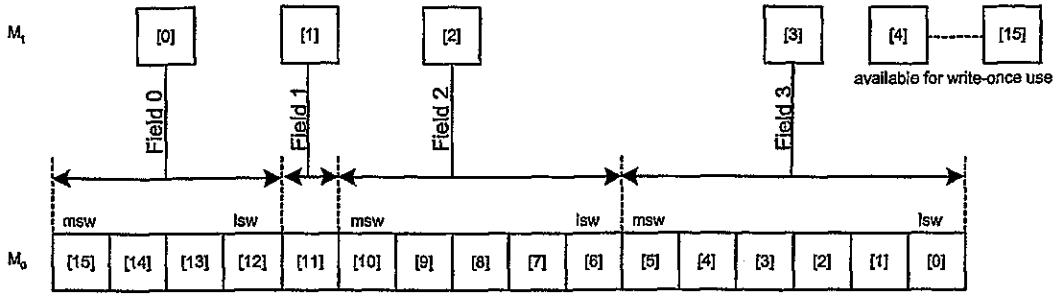


FIG. 365

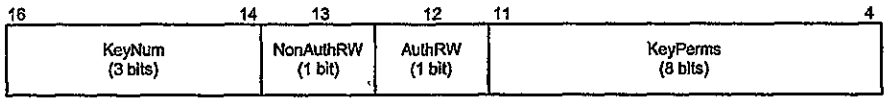


FIG. 366

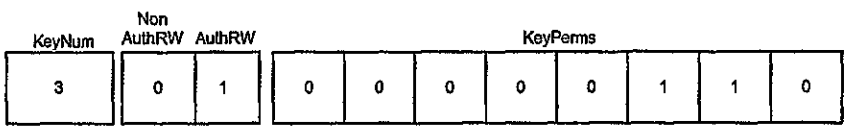


FIG. 367

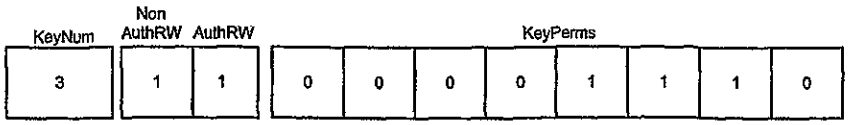


FIG. 368

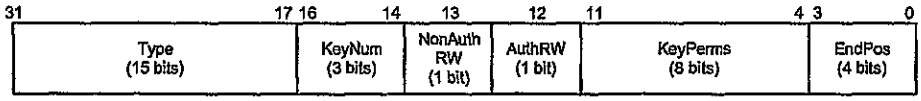


FIG. 369

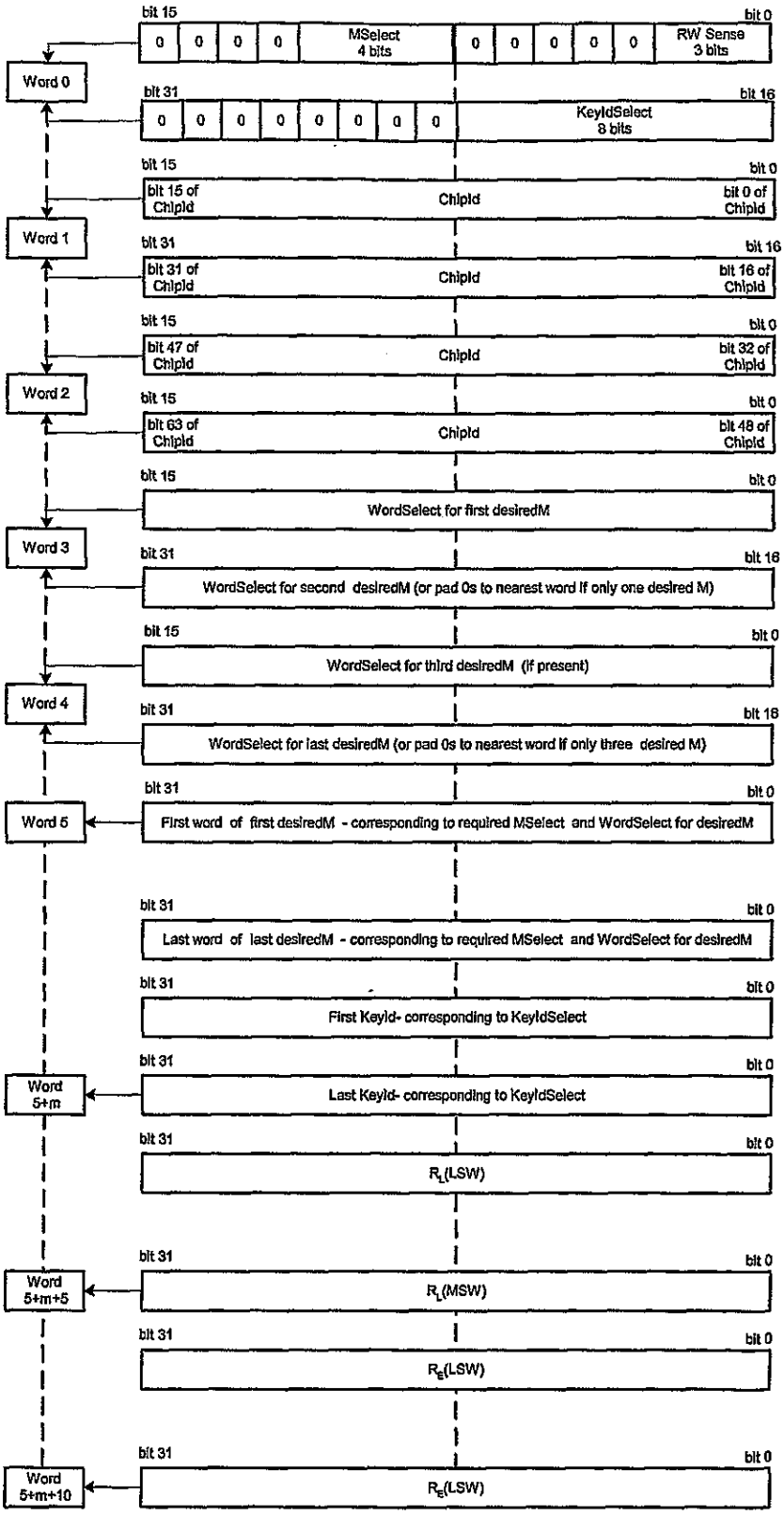

bit 15
bit 0
0 0 0 0 MSelect 4 bits 0 0 0 0 0 RW Sense 3 bits
Word 0
bit 31
bit 16
0 0 0 0 0 0 0 0 KeyIdSelect 8 bits
bit 15
bit 0
bit 15 of ChipId
ChipId
bit 0 of ChipId
Word 1
bit 31
bit 16
bit 31 of ChipId
ChipId
bit 16 of ChipId
bit 15
bit 0
bit 47 of ChipId
ChipId
bit 32 of ChipId
Word 2
bit 15
bit 0
bit 63 of ChipId
ChipId
bit 48 of ChipId
bit 15
bit 0
WordSelect for first desiredM
Word 3
bit 31
bit 16
WordSelect for second desiredM (or pad 0s to nearest word if only one desired M)
bit 15
bit 0
WordSelect for third desiredM (if present)
Word 4
bit 31
bit 16
WordSelect for last desiredM (or pad 0s to nearest word if only three desired M)
bit 31
bit 0
Word 5
First word of first desiredM - corresponding to required MSelect and WordSelect for desiredM
bit 31
bit 0
Last word of last desiredM - corresponding to required MSelect and WordSelect for desiredM
bit 31
bit 0
First KeyId- corresponding to KeyIdSelect
bit 31
bit 0
Word 5+m
Last KeyId- corresponding to KeyIdSelect
bit 31
bit 0
R_L(LSW)
bit 31
bit 0
Word 5+m+5
R_L(MSW)
bit 31
bit 0
R_E(LSW)
bit 31
bit 0
Word 5+m+10
R_E(LSW)

FIG. 370

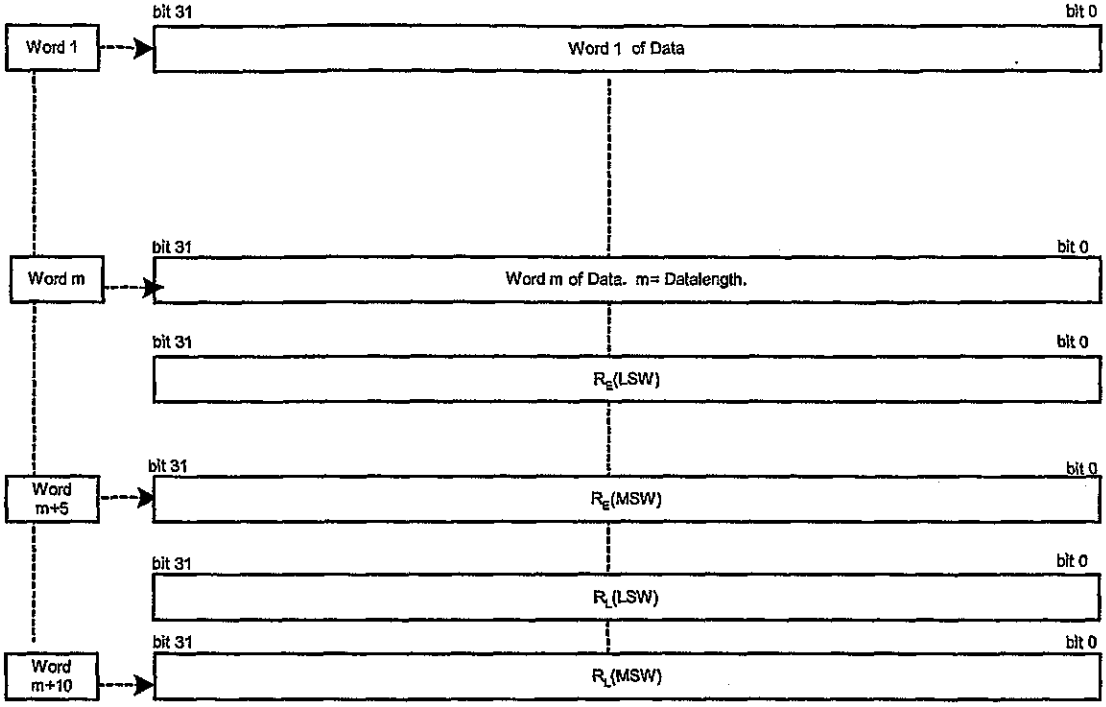

bit 31
bit 0
Word 1
Word 1 of Data
Word m
Word m of Data. m= Datalength.
R_E(LSW)
Word m+5
R_E(MSW)
R_L(LSW)
Word m+10
R_L(MSW)

FIG. 371

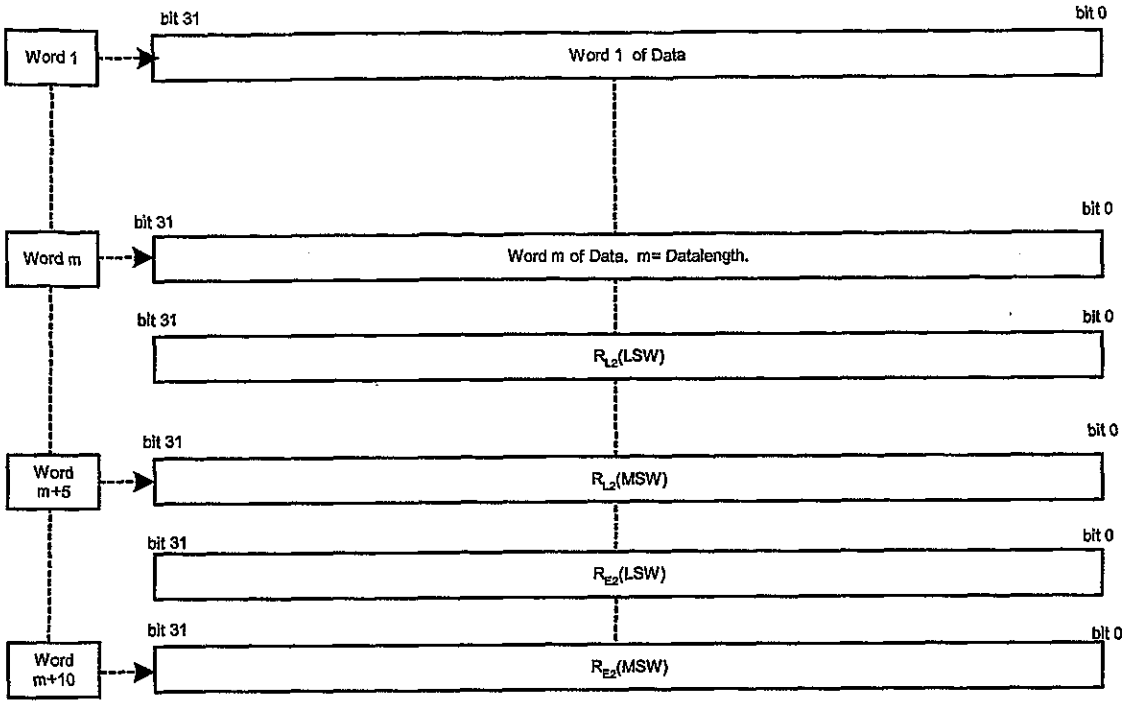

bit 31
bit 0
Word 1
Word 1 of Data
bit 31
bit 0
Word m
Word m of Data. m= Datalength.
bit 31
bit 0
R_{L2}(LSW)
bit 31
bit 0
Word m+5
R_{L2}(MSW)
bit 31
bit 0
R_{E2}(LSW)
bit 31
bit 0
Word m+10
R_{E2}(MSW)

FIG. 372

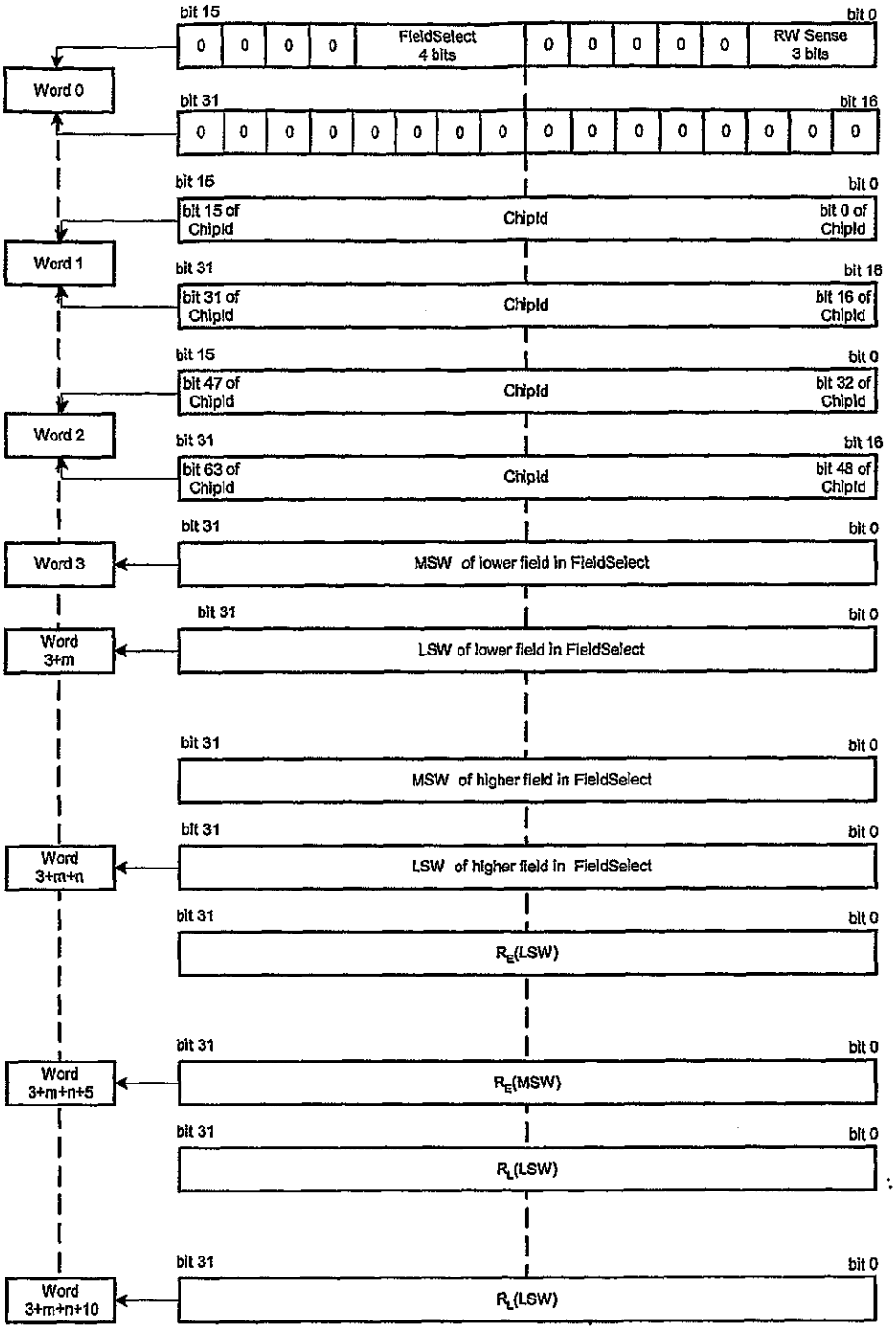

bit 15
bit 0
0 0 0 0 FieldSelect 4 bits 0 0 0 0 0 RW Sense 3 bits
Word 0
bit 31
bit 16
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
bit 15
bit 0
bit 15 of ChipId
ChipId
bit 0 of ChipId
Word 1
bit 31
bit 16
bit 31 of ChipId
ChipId
bit 16 of ChipId
bit 15
bit 0
bit 47 of ChipId
ChipId
bit 32 of ChipId
Word 2
bit 31
bit 16
bit 63 of ChipId
ChipId
bit 48 of ChipId
bit 31
bit 0
Word 3
MSW of lower field in FieldSelect
bit 31
bit 0
Word 3+m
LSW of lower field in FieldSelect
bit 31
bit 0
MSW of higher field in FieldSelect
bit 31
bit 0
Word 3+m+n
LSW of higher field in FieldSelect
bit 31
bit 0
R_E(LSW)
bit 31
bit 0
Word 3+m+n+5
R_E(MSW)
bit 31
bit 0
R_L(LSW)
bit 31
bit 0
Word 3+m+n+10
R_L(LSW)

FIG. 373

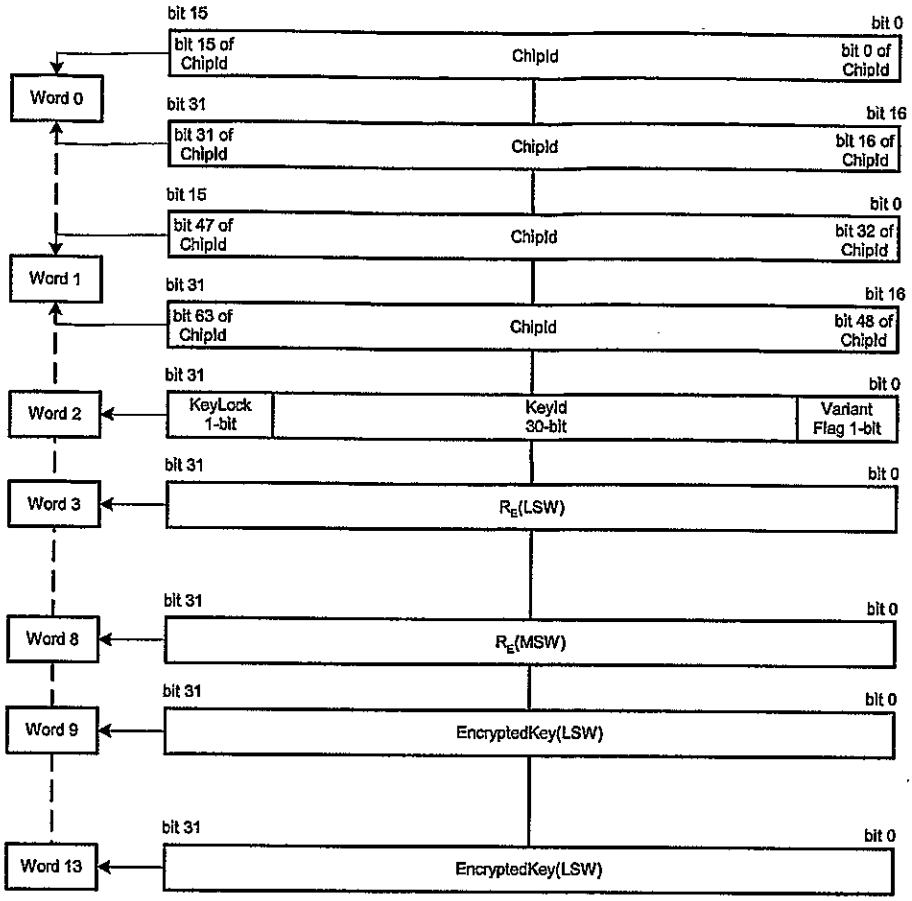

bit 15
bit 0
bit 15 of ChipId
ChipId
bit 0 of ChipId
Word 0
bit 31
bit 16
bit 31 of ChipId
ChipId
bit 16 of ChipId
bit 15
bit 0
bit 47 of ChipId
ChipId
bit 32 of ChipId
Word 1
bit 31
bit 16
bit 63 of ChipId
ChipId
bit 48 of ChipId
bit 31
bit 0
Word 2
KeyLock 1-bit
KeyId 30-bit
Variant Flag 1-bit
bit 31
bit 0
Word 3
R_E(LSW)
bit 31
bit 0
Word 8
R_E(MSW)
bit 31
bit 0
Word 9
EncryptedKey(LSW)
bit 31
bit 0
Word 13
EncryptedKey(LSW)

FIG. 374

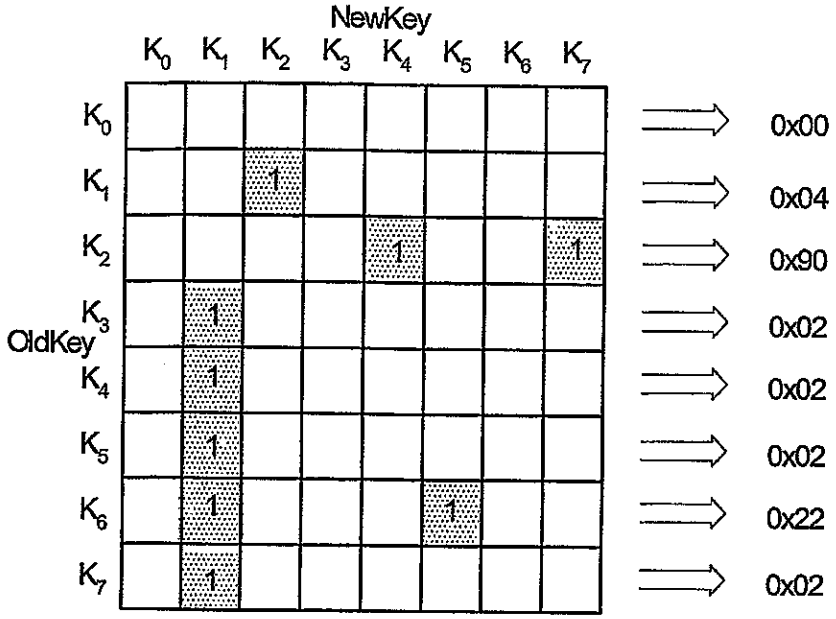


FIG. 375

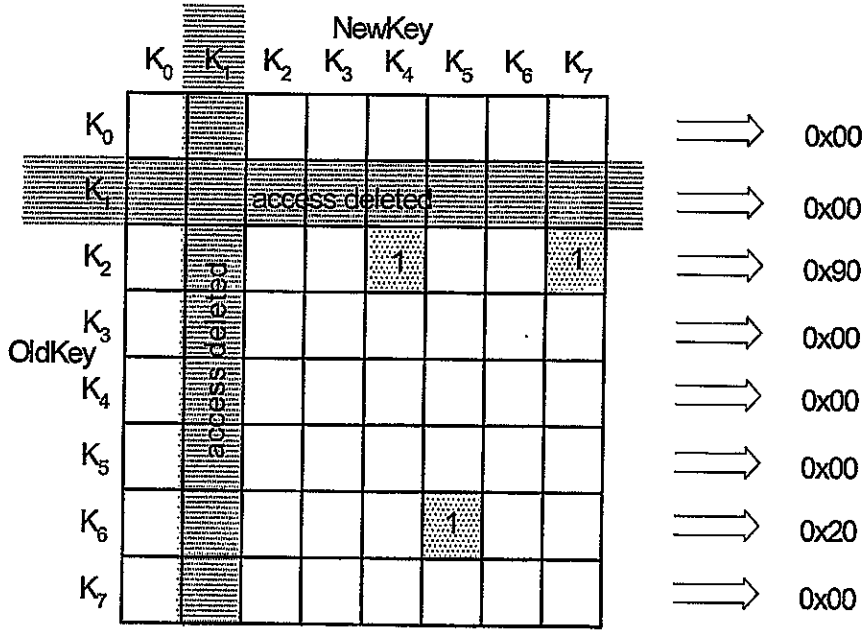


FIG. 376

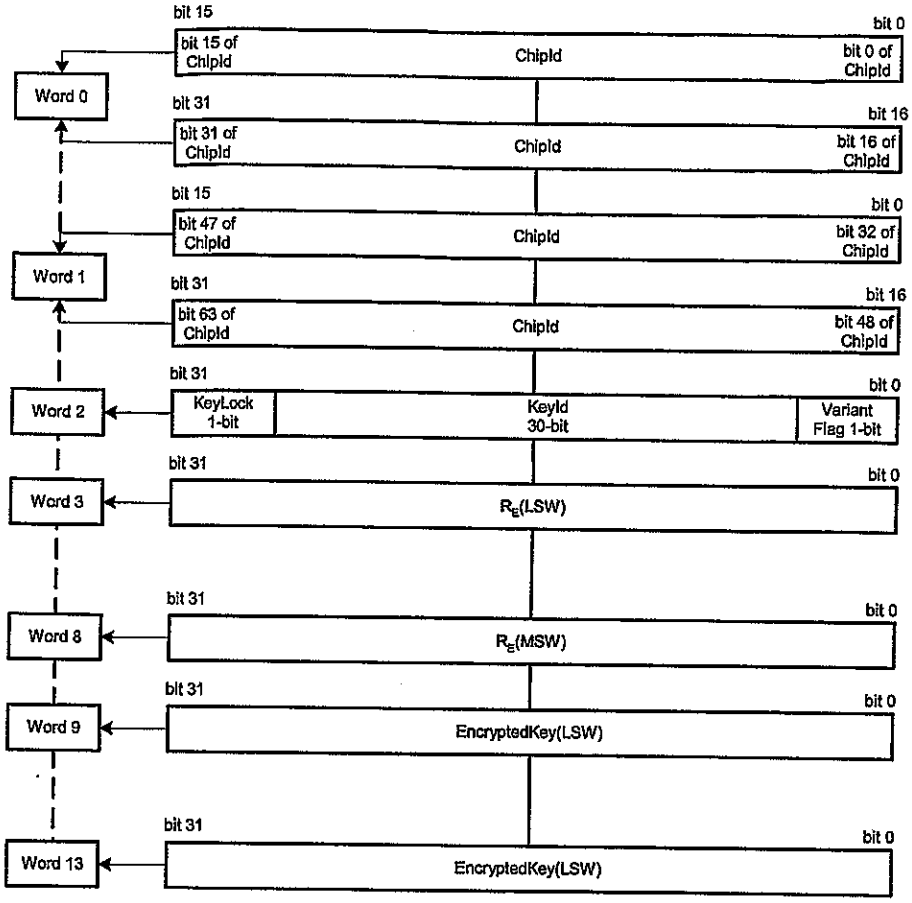

bit 15
bit 0
bit 15 of ChipId
ChipId
bit 0 of ChipId
Word 0
bit 31
bit 16
bit 31 of ChipId
ChipId
bit 16 of ChipId
bit 15
bit 0
bit 47 of ChipId
ChipId
bit 32 of ChipId
Word 1
bit 31
bit 16
bit 63 of ChipId
ChipId
bit 48 of ChipId
bit 31
bit 0
Word 2
KeyLock 1-bit
KeyId 30-bit
Variant Flag 1-bit
bit 31
bit 0
Word 3
R_E(LSW)
bit 31
bit 0
Word 8
R_E(MSW)
bit 31
bit 0
Word 9
EncryptedKey(LSW)
bit 31
bit 0
Word 13
EncryptedKey(LSW)

FIG. 377

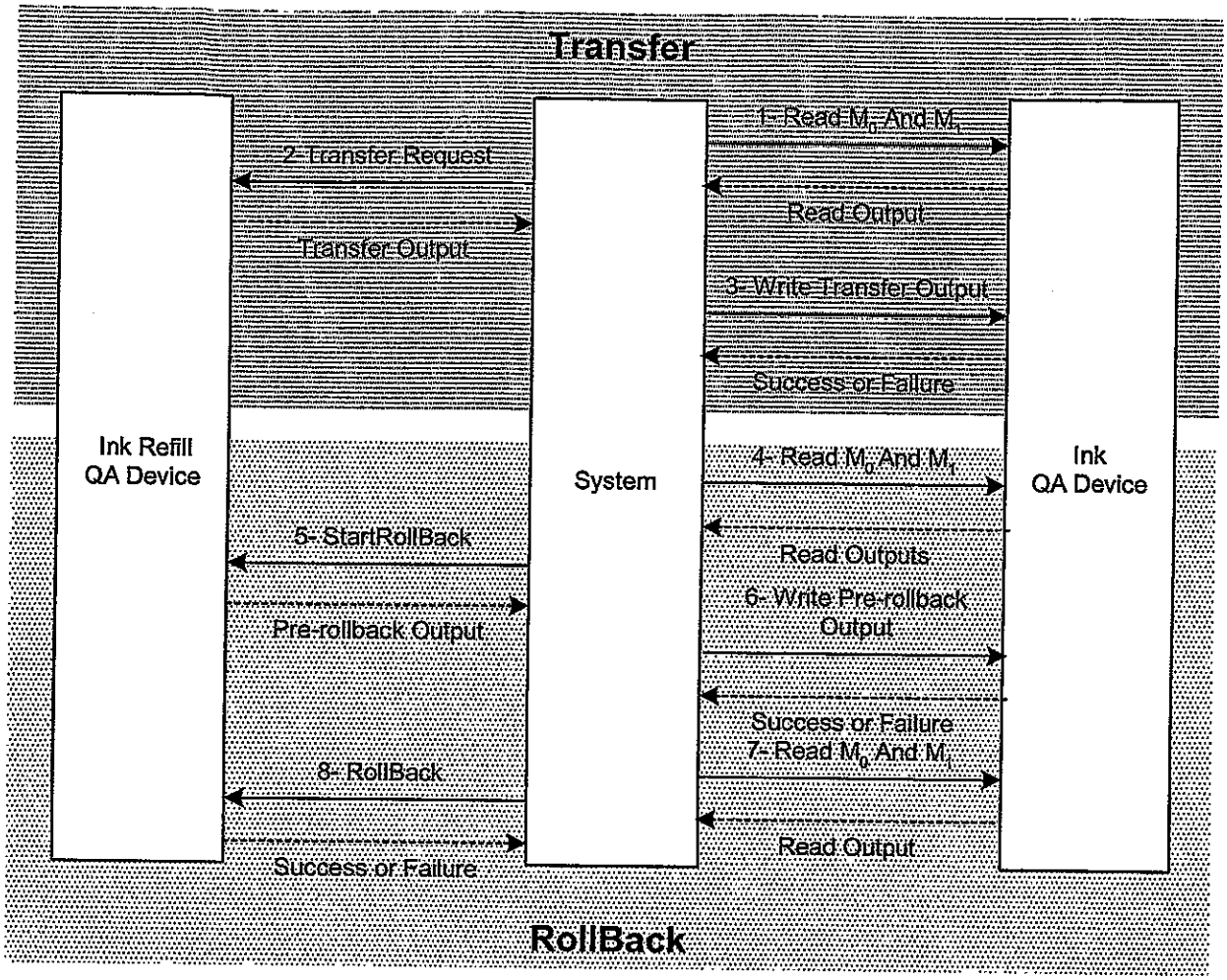

Transfer
Ink Refill QA Device
System
Ink QA Device
1- Read M_0 And M_t
Read Output
2- Transfer Request
Transfer Output
3- Write Transfer Output
Success or Failure
4- Read M_0 And M_t
Read Outputs
5- StartRollBack
Pre-rollback Output
6- Write Pre-rollback Output
Success or Failure
7- Read M_0 And M_t
Read Output
8- RollBack
Success or Failure
RollBack

FIG. 378

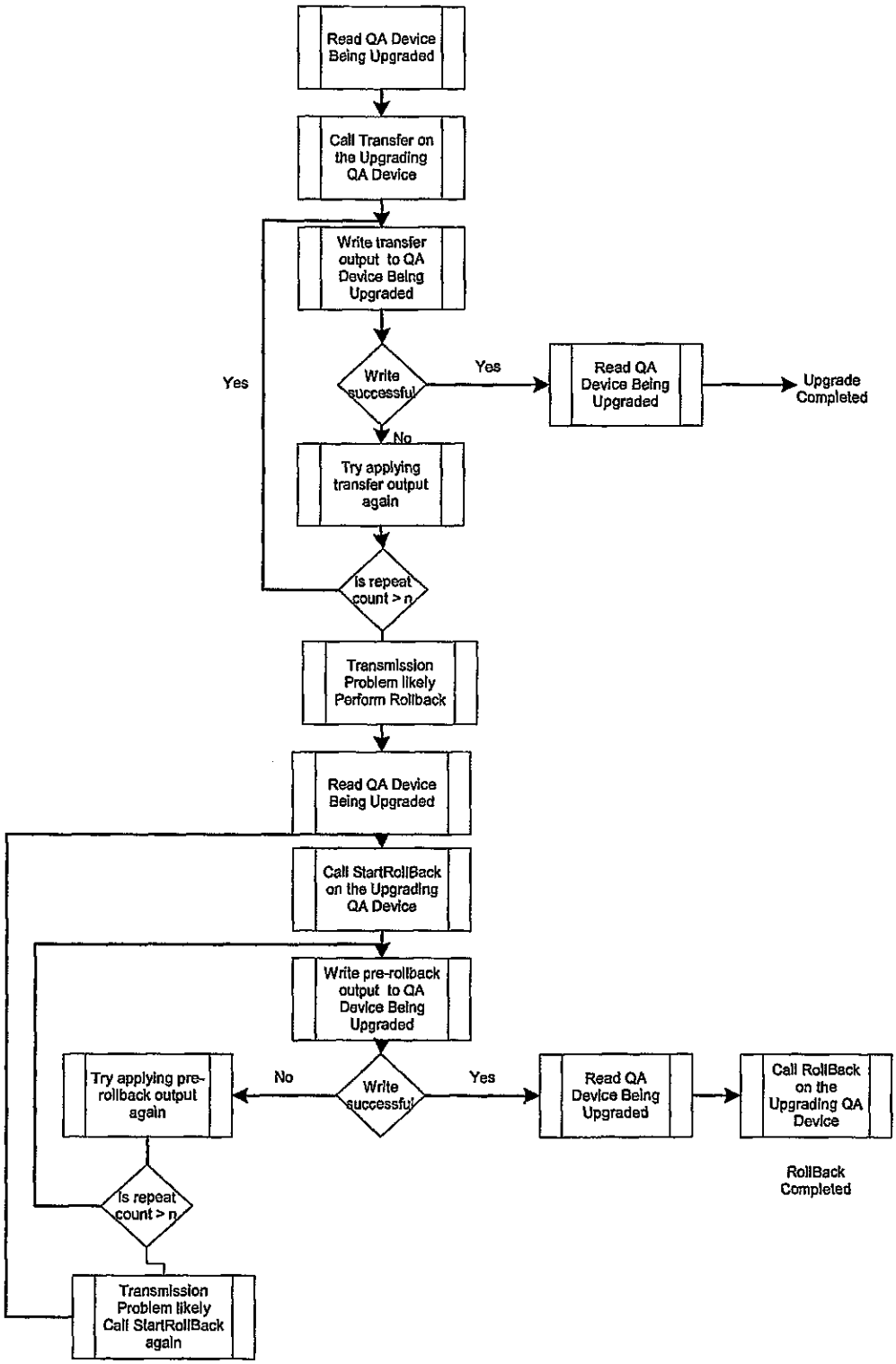

Read QA Device Being Upgraded
Call Transfer on the Upgrading QA Device
Write transfer output to QA Device Being Upgraded
Write successful
Yes
Yes
Read QA Device Being Upgraded
Upgrade Completed
No
Try applying transfer output again
Is repeat count > n
Yes
Transmission Problem likely Perform Rollback
Read QA Device Being Upgraded
Call StartRollBack on the Upgrading QA Device
Write pre-rollback output to QA Device Being Upgraded
Write successful
No
Yes
Try applying pre-rollback output again
Read QA Device Being Upgraded
Call RollBack on the Upgrading QA Device
RollBack Completed
Is repeat count > n
Transmission Problem likely Call StartRollBack again

FIG. 379

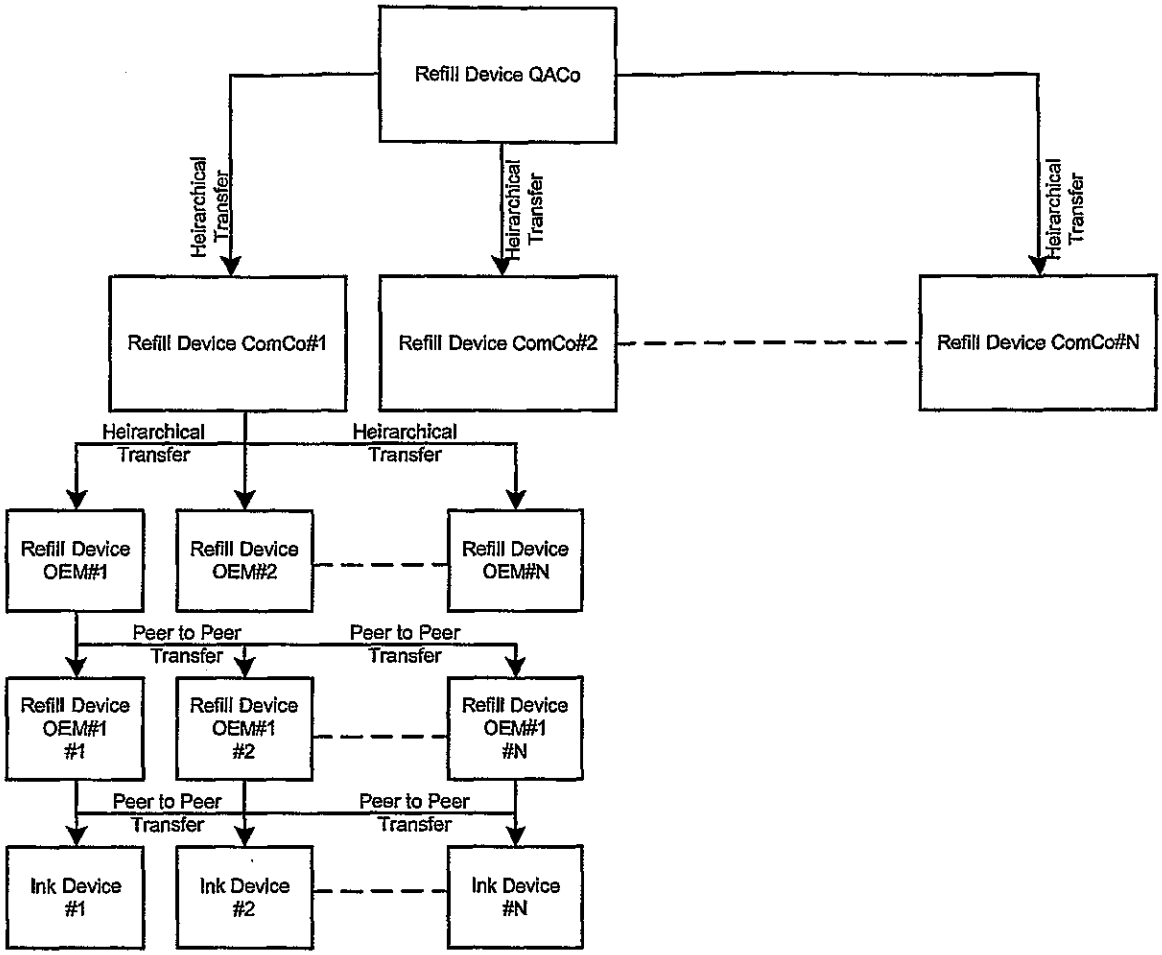

Refill Device QACo
Heirarchical Transfer
Heirarchical Transfer
Heirarchical Transfer
Refill Device ComCo#1
Refill Device ComCo#2
Refill Device ComCo#N
Heirarchical Transfer
Heirarchical Transfer
Refill Device OEM#1
Refill Device OEM#2
Refill Device OEM#N
Peer to Peer Transfer
Peer to Peer Transfer
Refill Device OEM#1 #1
Refill Device OEM#1 #2
Refill Device OEM#1 #N
Peer to Peer Transfer
Peer to Peer Transfer
Ink Device #1
Ink Device #2
Ink Device #N

FIG. 380

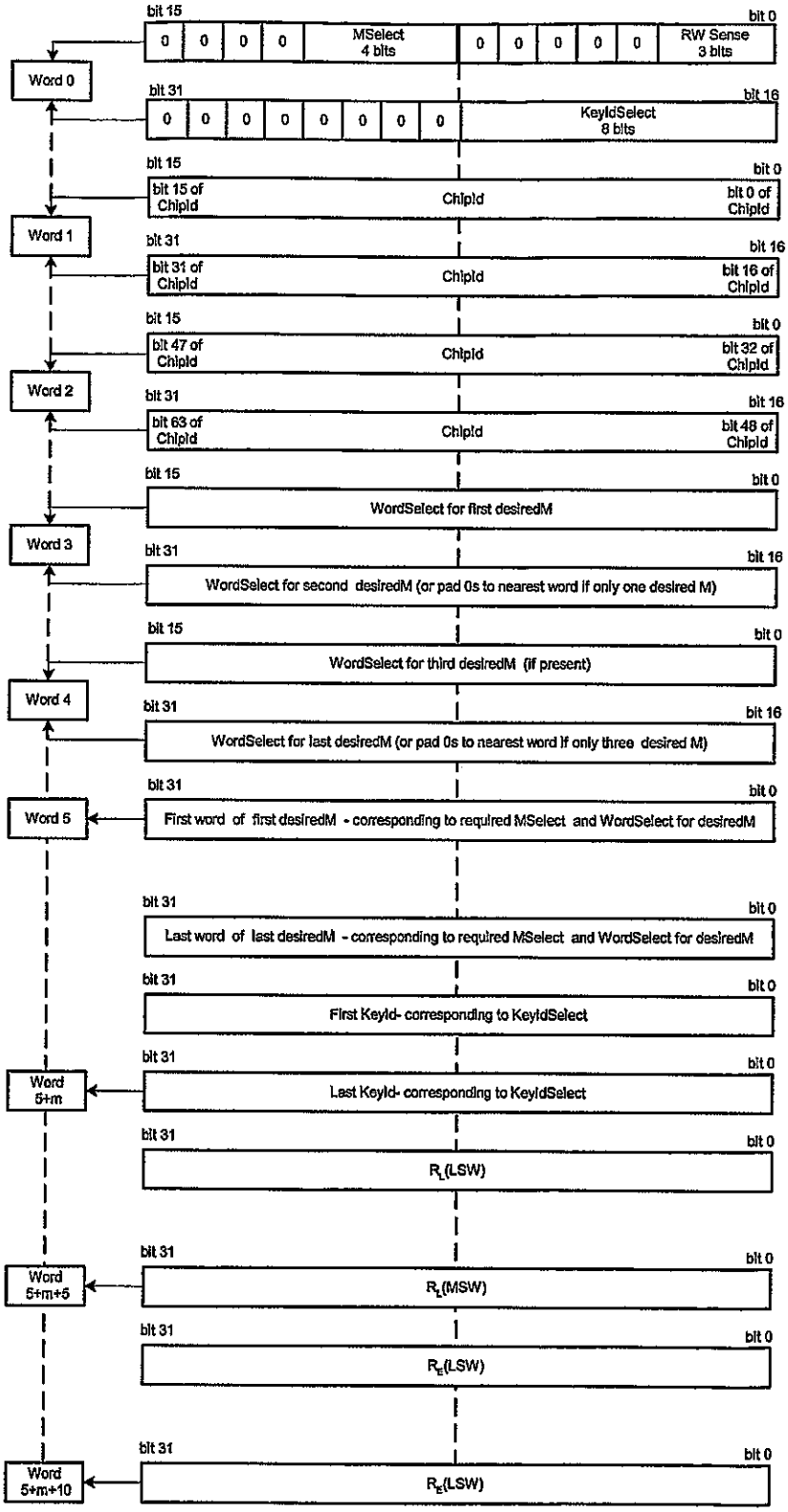

bit 15
bit 0
0 0 0 0 MSelect 4 bits 0 0 0 0 0 RW Sense 3 bits
Word 0
bit 31
bit 16
0 0 0 0 0 0 0 0 KeyIdSelect 8 bits
bit 15
bit 0
bit 15 of ChipId
ChipId
bit 0 of ChipId
Word 1
bit 31
bit 16
bit 31 of ChipId
ChipId
bit 16 of ChipId
bit 15
bit 0
bit 47 of ChipId
ChipId
bit 32 of ChipId
Word 2
bit 31
bit 16
bit 63 of ChipId
ChipId
bit 48 of ChipId
bit 15
bit 0
WordSelect for first desiredM
Word 3
bit 31
bit 16
WordSelect for second desiredM (or pad 0s to nearest word if only one desired M)
bit 15
bit 0
WordSelect for third desiredM (if present)
Word 4
bit 31
bit 16
WordSelect for last desiredM (or pad 0s to nearest word if only three desired M)
bit 31
bit 0
Word 5
First word of first desiredM - corresponding to required MSelect and WordSelect for desiredM
bit 31
bit 0
Last word of last desiredM - corresponding to required MSelect and WordSelect for desiredM
bit 31
bit 0
First KeyId- corresponding to KeyIdSelect
bit 31
bit 0
Word 5+m
Last KeyId- corresponding to KeyIdSelect
bit 31
bit 0
R_L(LSW)
bit 31
bit 0
Word 5+m+5
R_L(MSW)
bit 31
bit 0
R_E(LSW)
bit 31
bit 0
Word 5+m+10
R_E(LSW)

FIG. 381

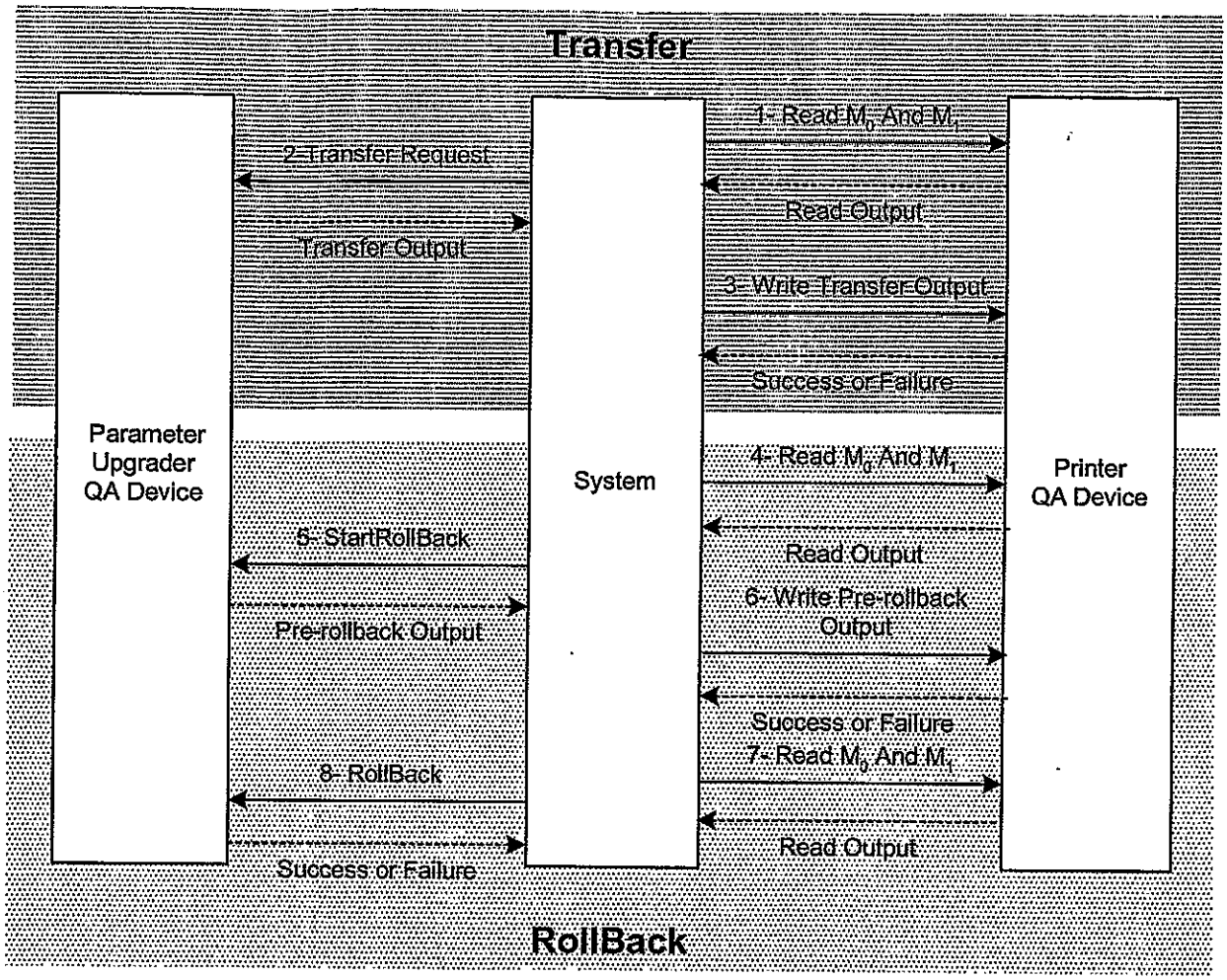

Transfer
Parameter Upgrader QA Device
System
Printer QA Device
1- Read M_0 And M_1
Read Output
2- Transfer Request
Transfer Output
3- Write Transfer Output
Success or Failure
4- Read M_0 And M_1
Read Output
5- StartRollBack
Pre-rollback Output
6- Write Pre-rollback Output
Success or Failure
7- Read M_0 And M_1
Read Output
8- RollBack
Success or Failure
RollBack

FIG. 382

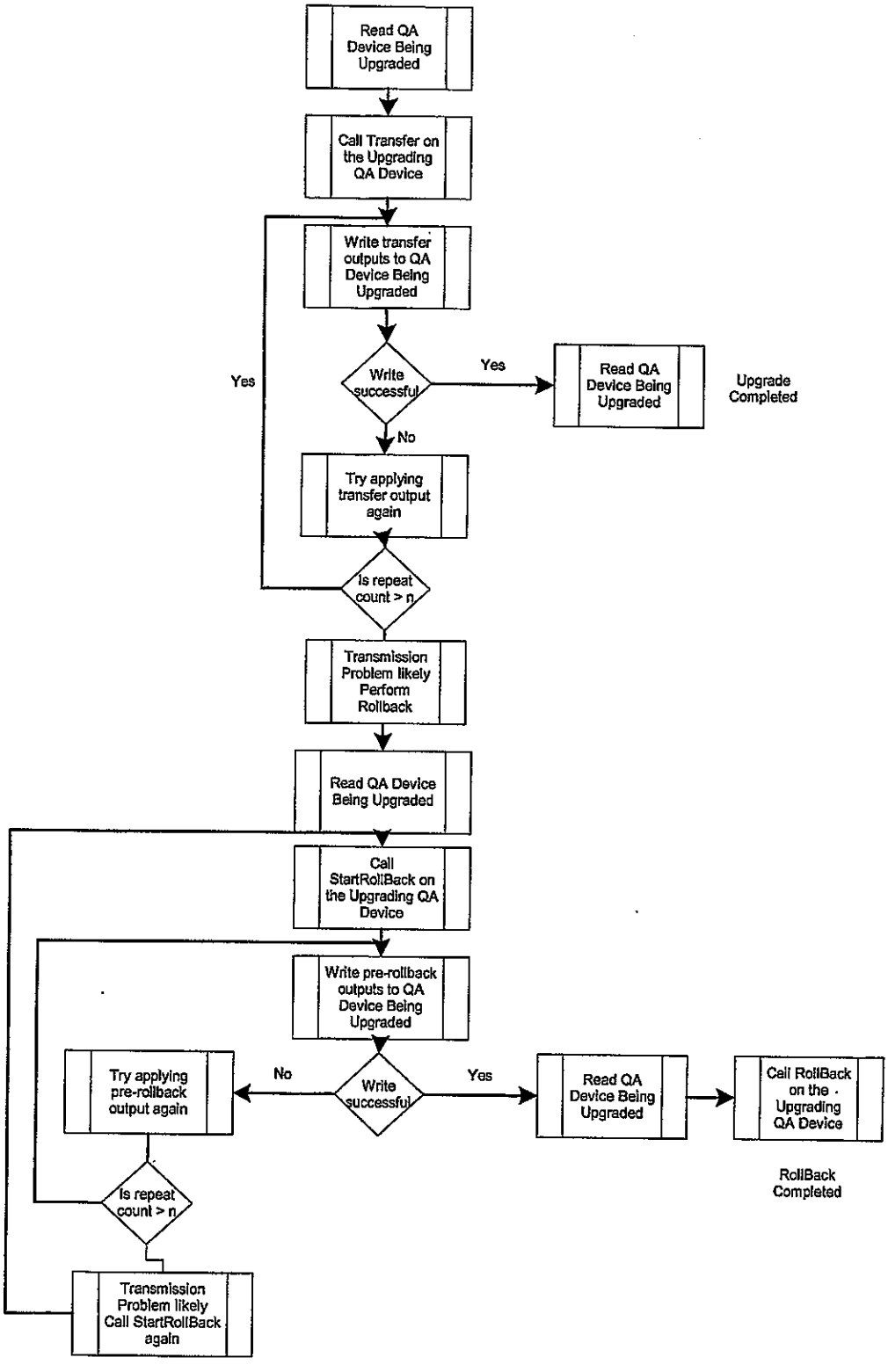

Read QA Device Being Upgraded
Call Transfer on the Upgrading QA Device
Write transfer outputs to QA Device Being Upgraded
Write successful
Yes
Read QA Device Being Upgraded
Upgrade Completed
No
Yes
Try applying transfer output again
Is repeat count > n
Transmission Problem likely Perform Rollback
Read QA Device Being Upgraded
Call StartRollBack on the Upgrading QA Device
Write pre-rollback outputs to QA Device Being Upgraded
Try applying pre-rollback output again
No
Write successful
Yes
Read QA Device Being Upgraded
Call RollBack on the Upgrading QA Device
RollBack Completed
Is repeat count > n
Transmission Problem likely Call StartRollBack again

FIG. 383

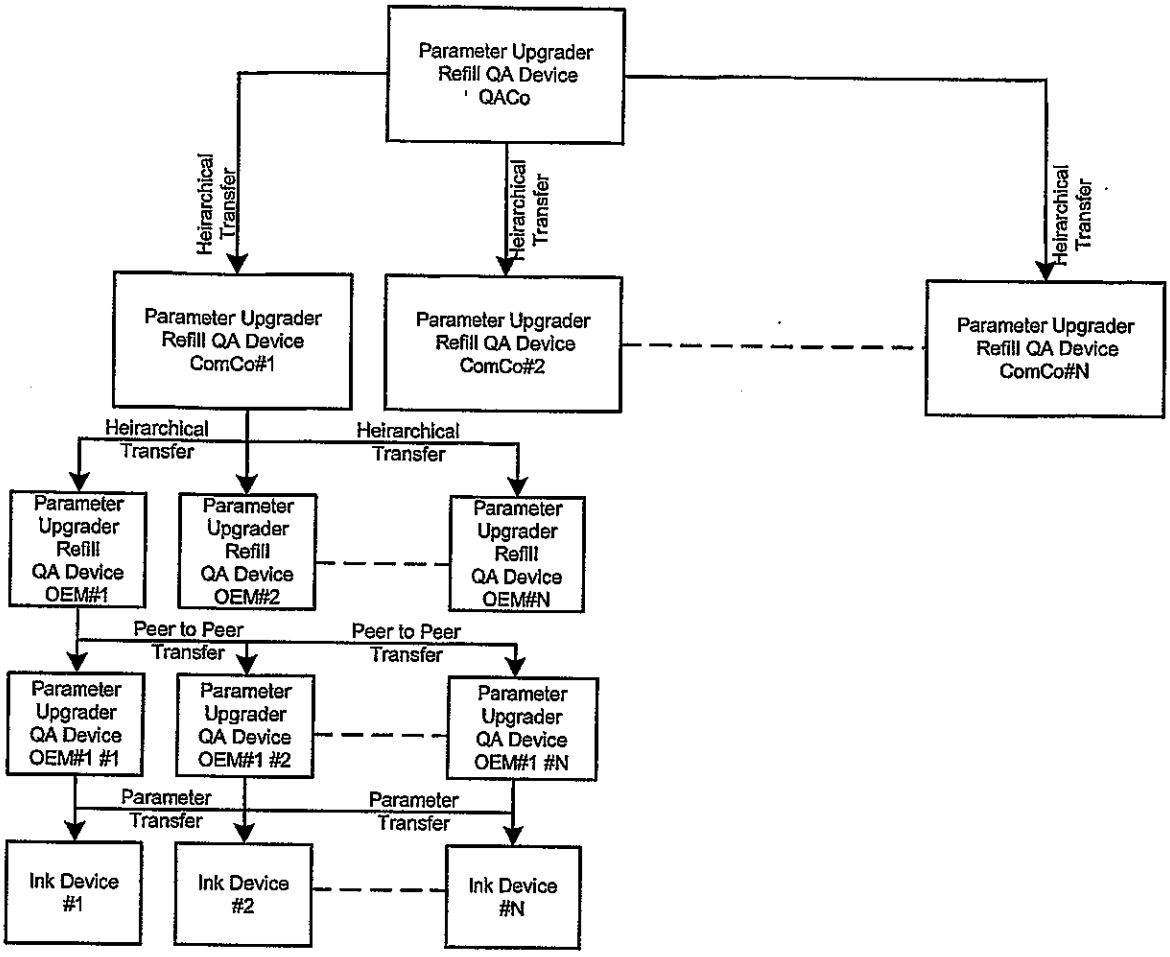


FIG. 384

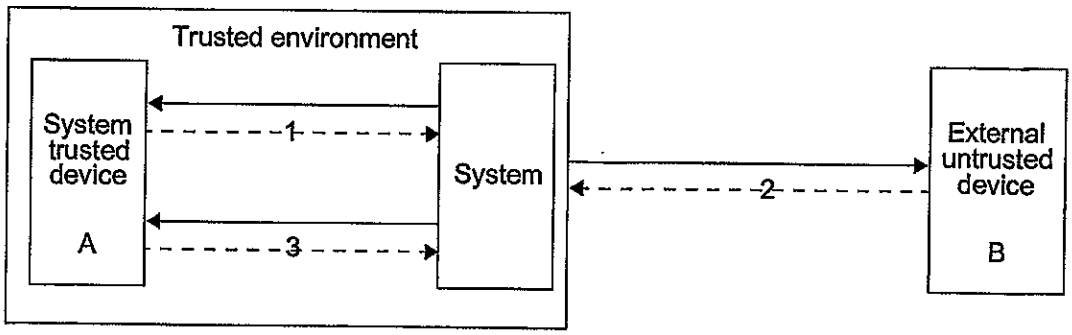


FIG. 385

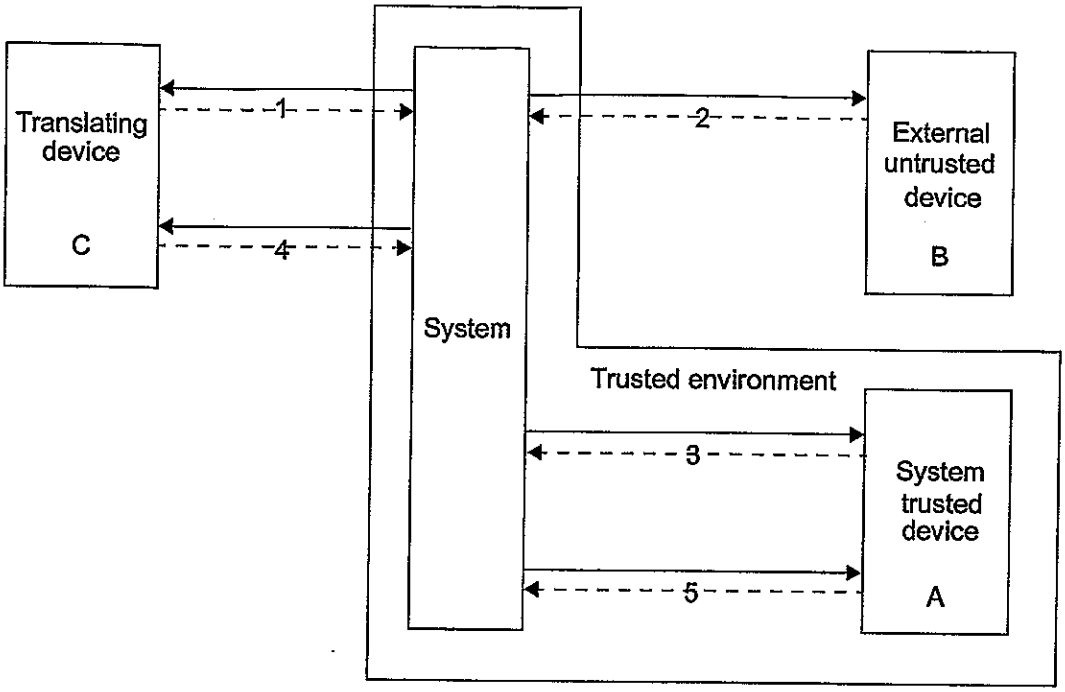

Translating device
C
1
4
System
2
External untrusted device
B
Trusted environment
3
System trusted device
A
5

FIG. 386

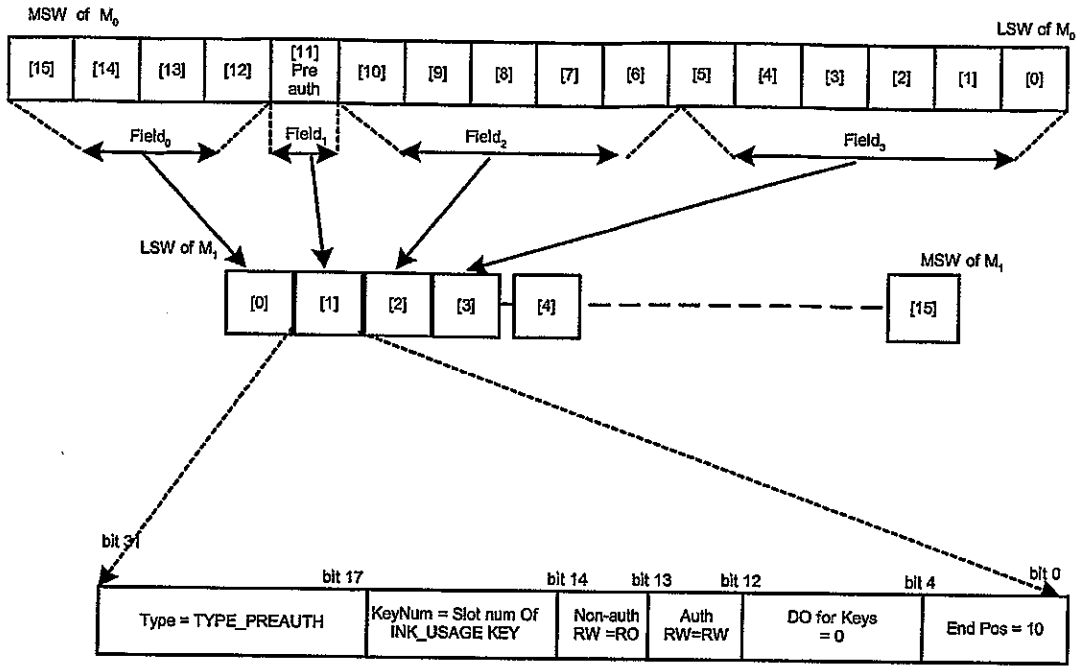


FIG. 387

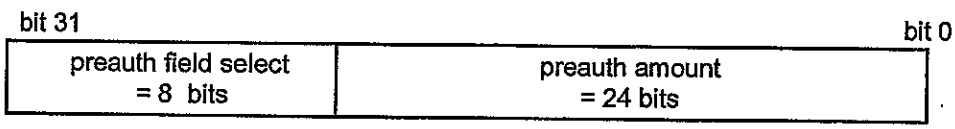


FIG. 388

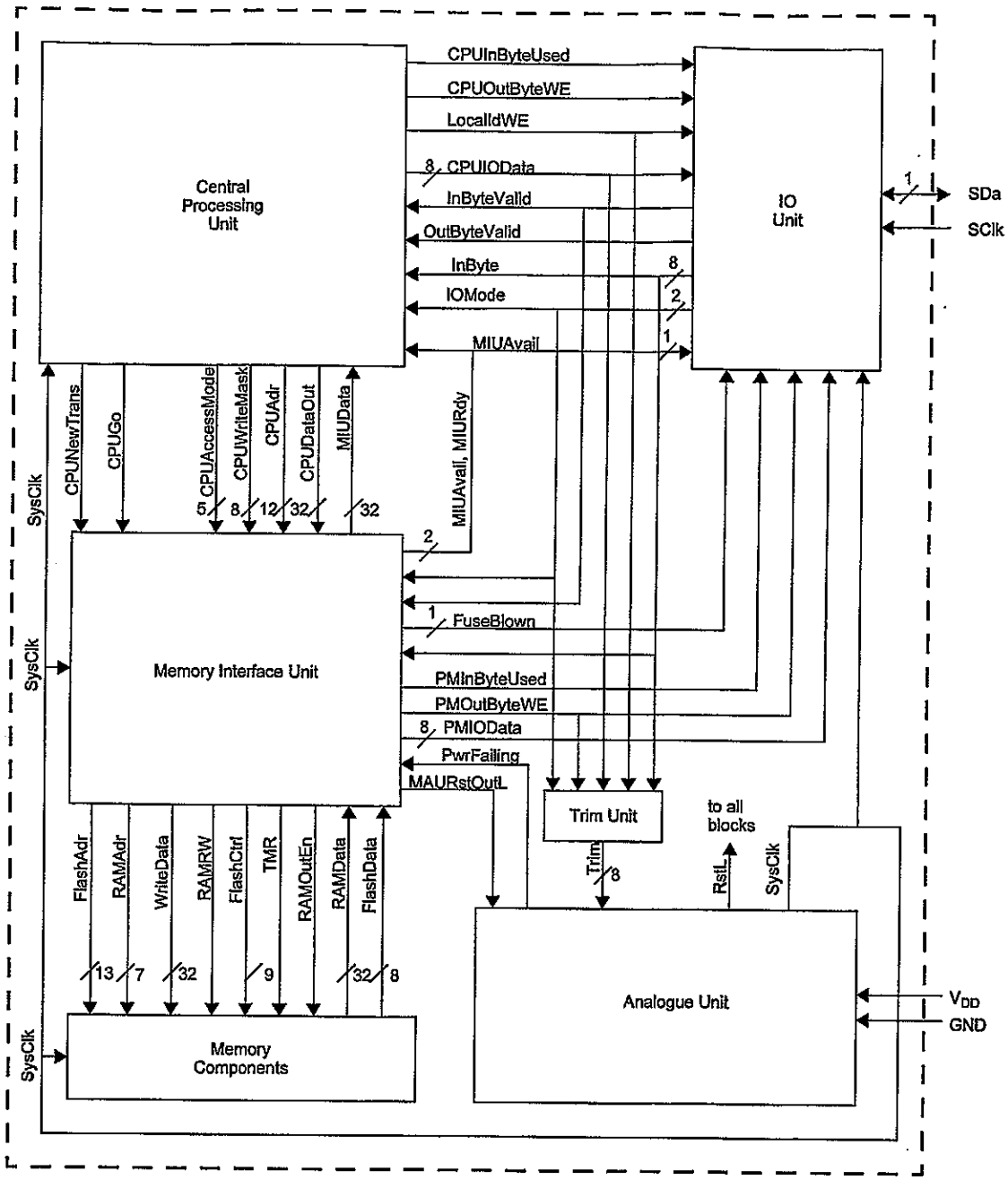

Central Processing Unit
IO Unit
Memory Interface Unit
Memory Components
Analogue Unit
Trim Unit
CPUInByteUsed
CPUOutByteWE
LocalIdWE
8 CPUIOData
InByteValid
OutByteValid
InByte
IOMode
MIUAvail
SDa
SClk
CPUNewTrans
CPUGo
CPUAccessMode
CPUWriteMask
CPUAdr
CPUDataOut
MIUData
MIUAvail, MIURdy
SysClk
FuseBlown
PMInByteUsed
PMOutByteWE
8 PMIOData
PwrFailing
MAURstOutL
FlashAdr
RAMAdr
WriteData
RAMRW
FlashCtrl
TMR
RAMOutEn
RAMData
FlashData
Trim
to all blocks
RstL
SysClk
VDD
GND

FIG. 389

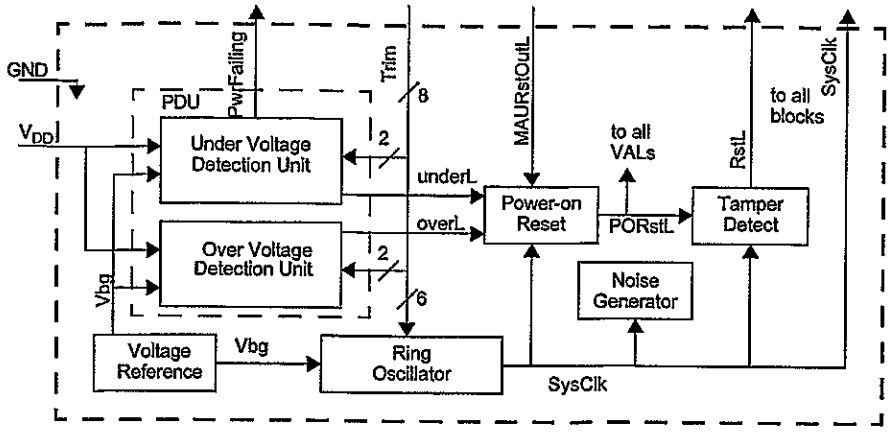


FIG. 390

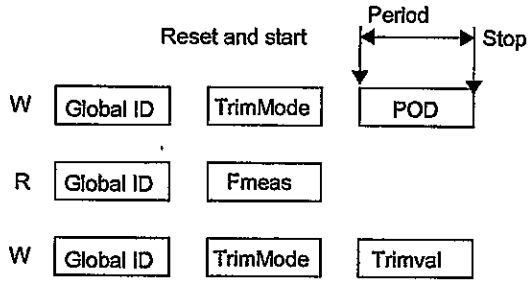


FIG. 391

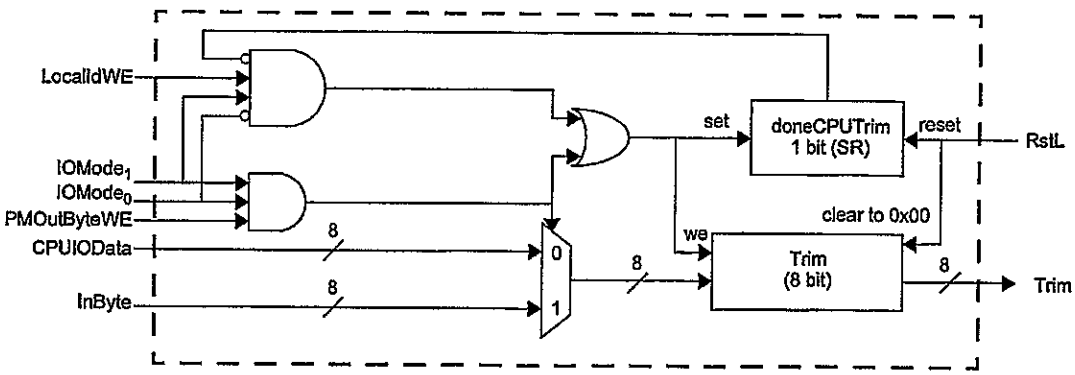


FIG. 392

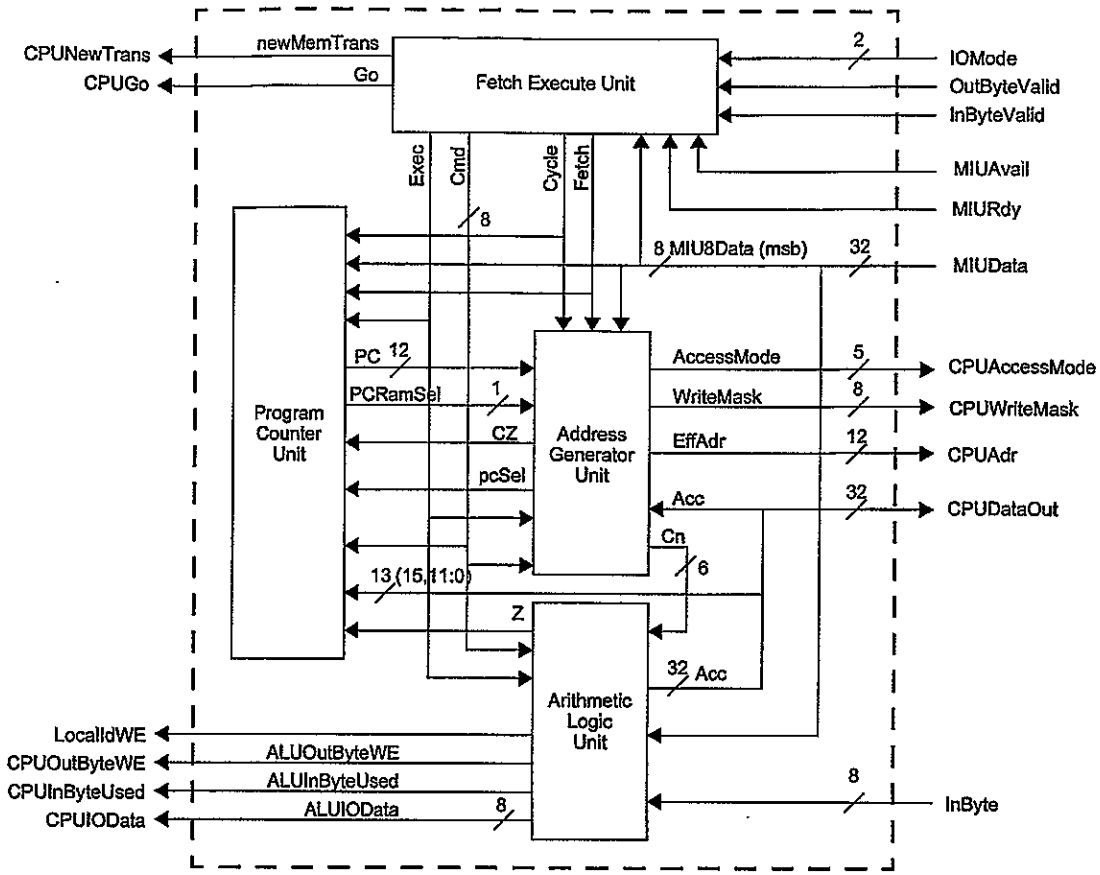

CPUNewTrans
CPUGo
newMemTrans
Go
Fetch Execute Unit
2
IOMode
OutByteValid
InByteValid
MIUAvail
MIURdy
Exec
Cmd
Cycle
Fetch
8
8 MIU8Data (msb)
32
MIUData
Program Counter Unit
PC 12
PCRamSel 1
CZ
pcSel
13 (15,11:0)
Address Generator Unit
AccessMode
5
CPUAccessMode
WriteMask
8
CPUWriteMask
EffAdr
12
CPUAdr
Acc
32
CPUDataOut
Cn
6
Z
32 Acc
Arithmetic Logic Unit
LocalIdWE
CPUOutByteWE
CPUInByteUsed
CPUIOData
ALUOutByteWE
ALUInByteUsed
ALUIOData
8
8
InByte

FIG. 393

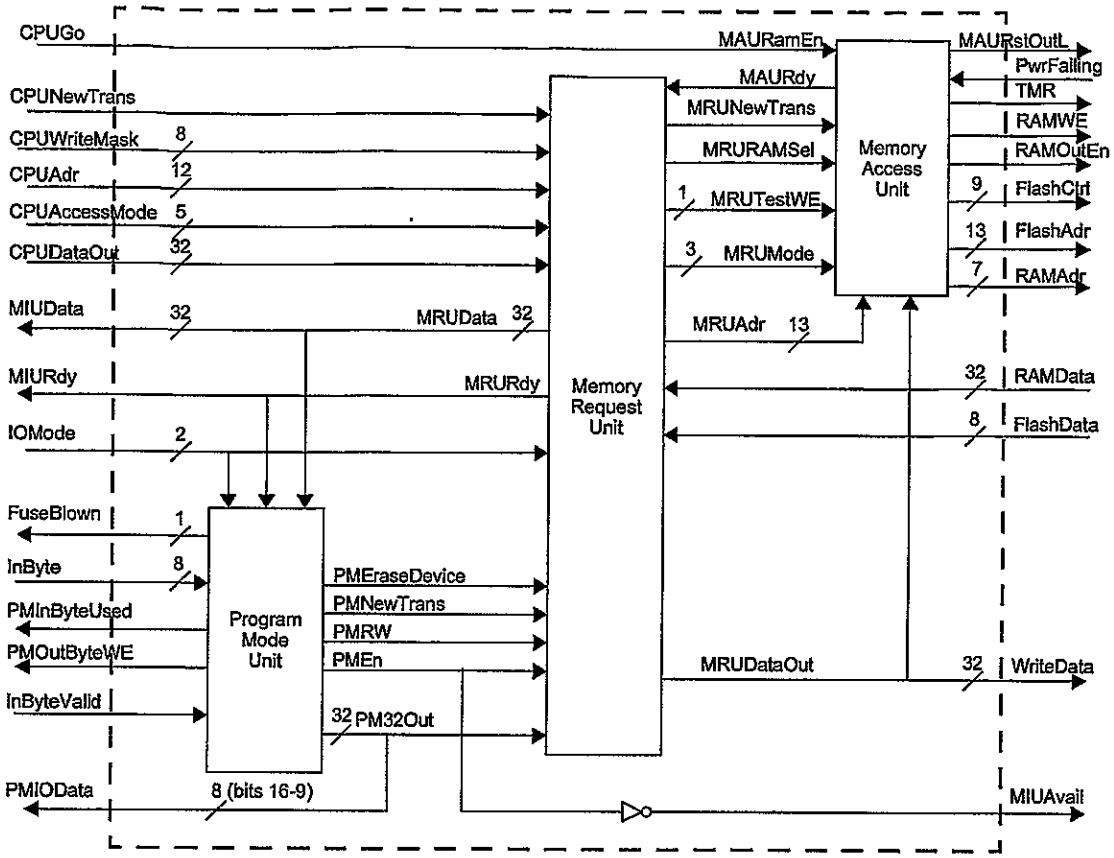


FIG. 394

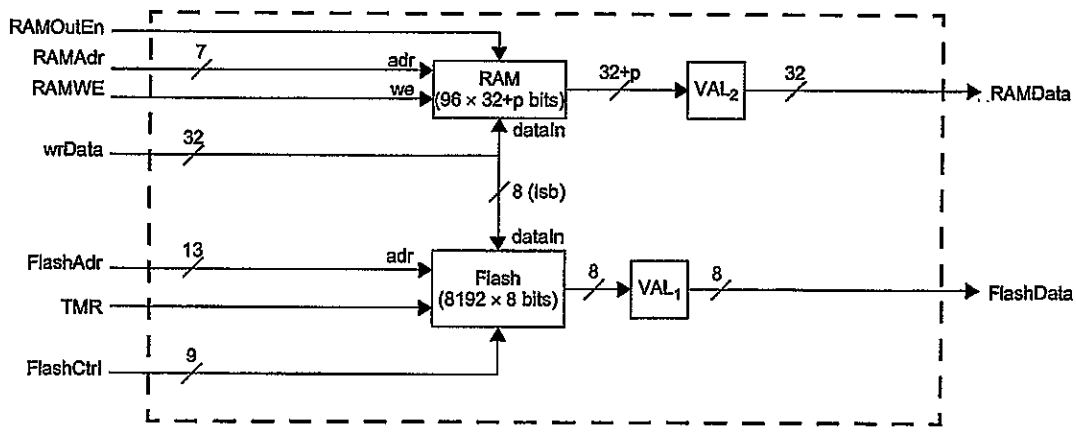


FIG. 395

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PrilD6	PrilD5	PrilD4	PrilD3	PrilD2	PrilD1	PrilD0	R/*W 0 = write 1 = read

FIG. 396

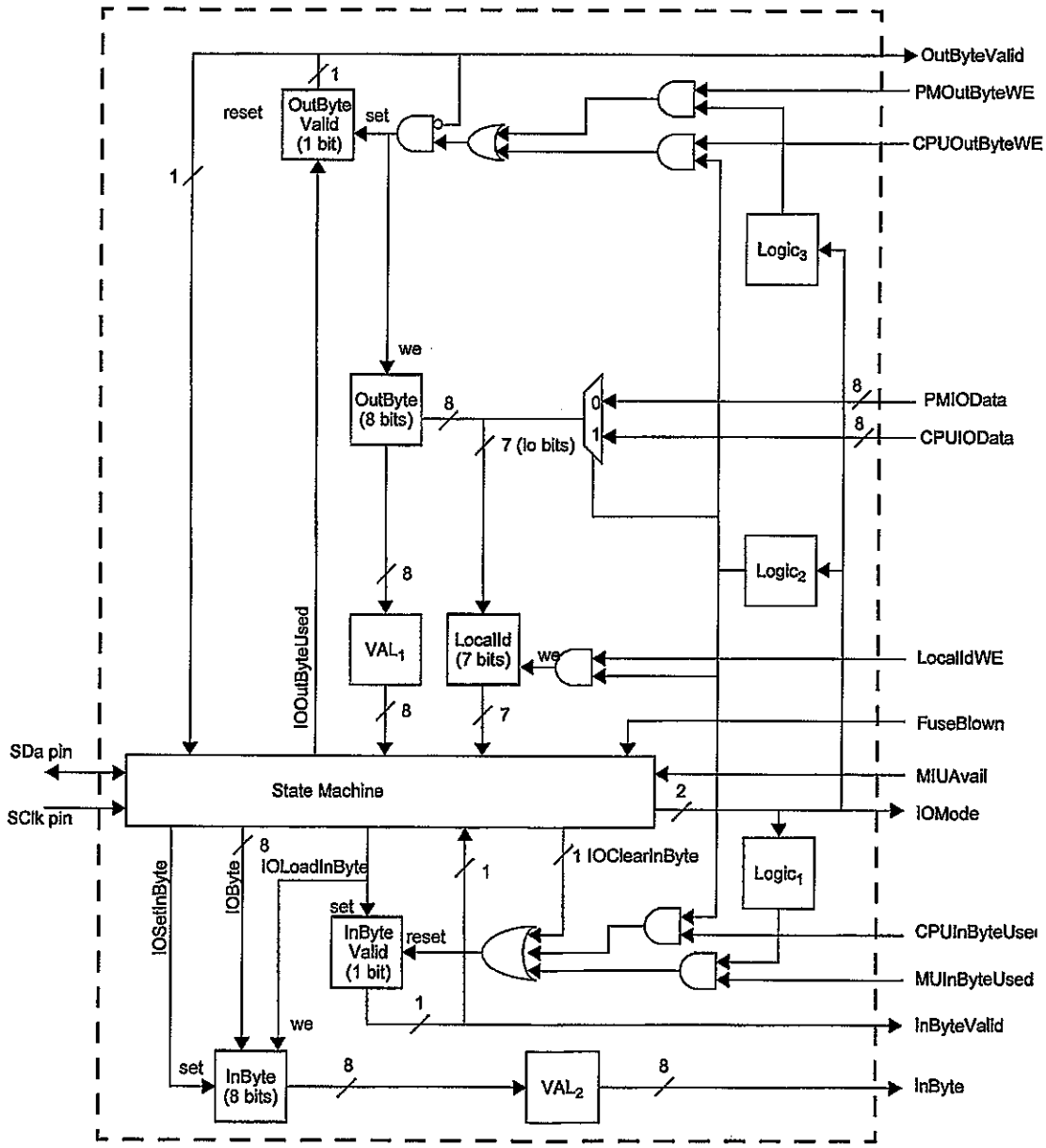


FIG. 397

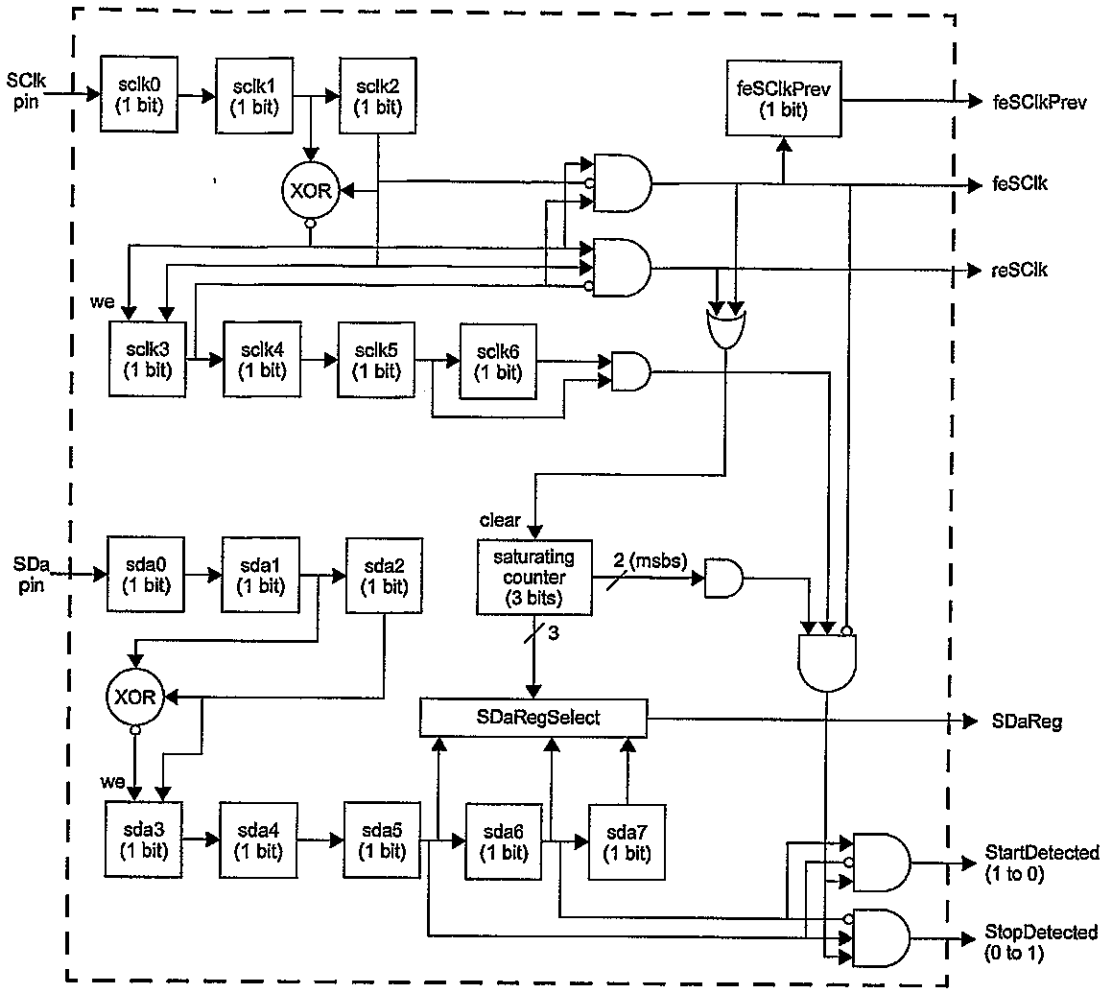

SClk pin
sclk0 (1 bit)
sclk1 (1 bit)
sclk2 (1 bit)
feSClkPrev (1 bit)
feSClkPrev
XOR
feSClk
reSClk
we
sclk3 (1 bit)
sclk4 (1 bit)
sclk5 (1 bit)
sclk6 (1 bit)
clear
SDa pin
sda0 (1 bit)
sda1 (1 bit)
sda2 (1 bit)
saturating counter (3 bits)
2 (msbs)
3
XOR
SDaRegSelect
SDaReg
we
sda3 (1 bit)
sda4 (1 bit)
sda5 (1 bit)
sda6 (1 bit)
sda7 (1 bit)
StartDetected (1 to 0)
StopDetected (0 to 1)

FIG. 398

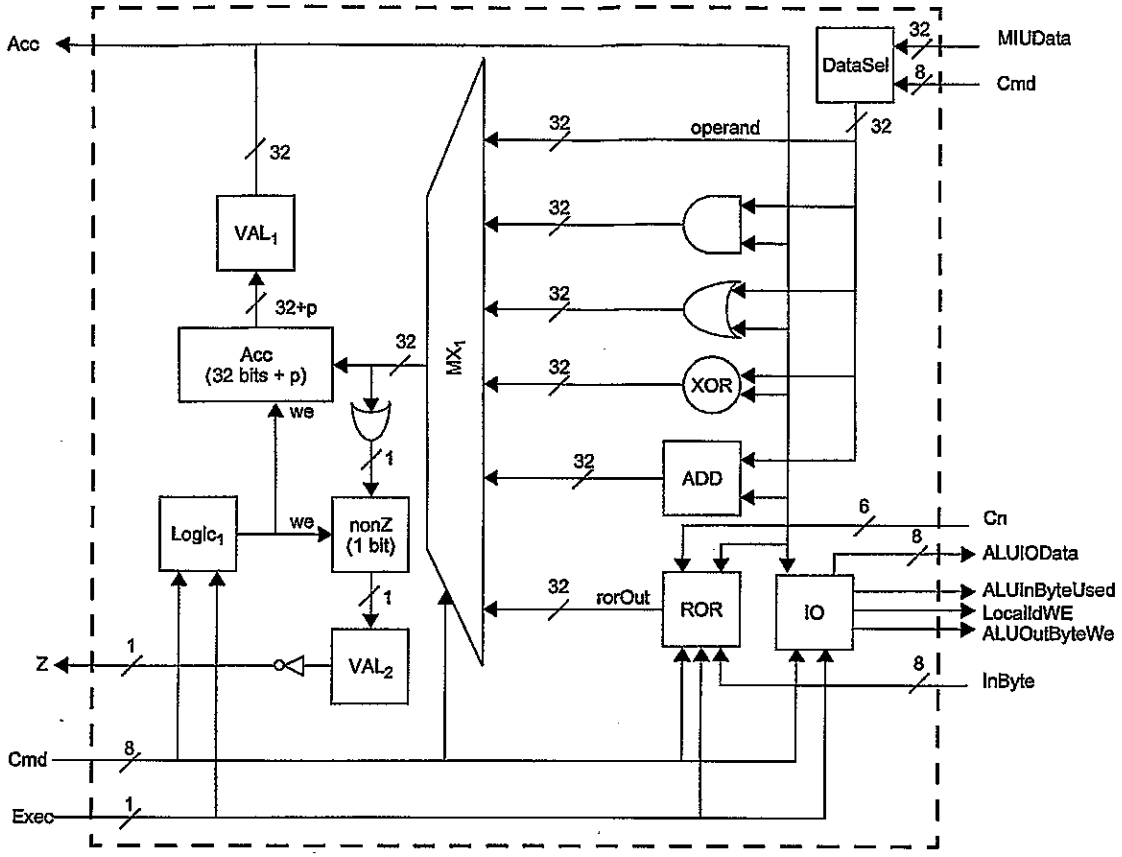


FIG. 399

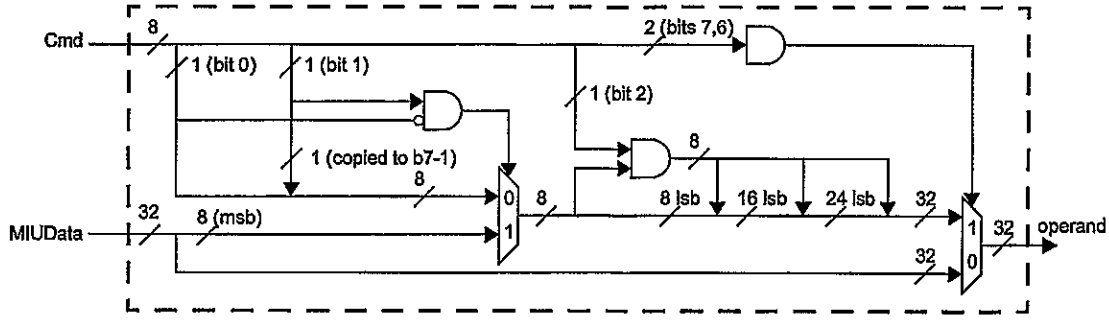


FIG. 400

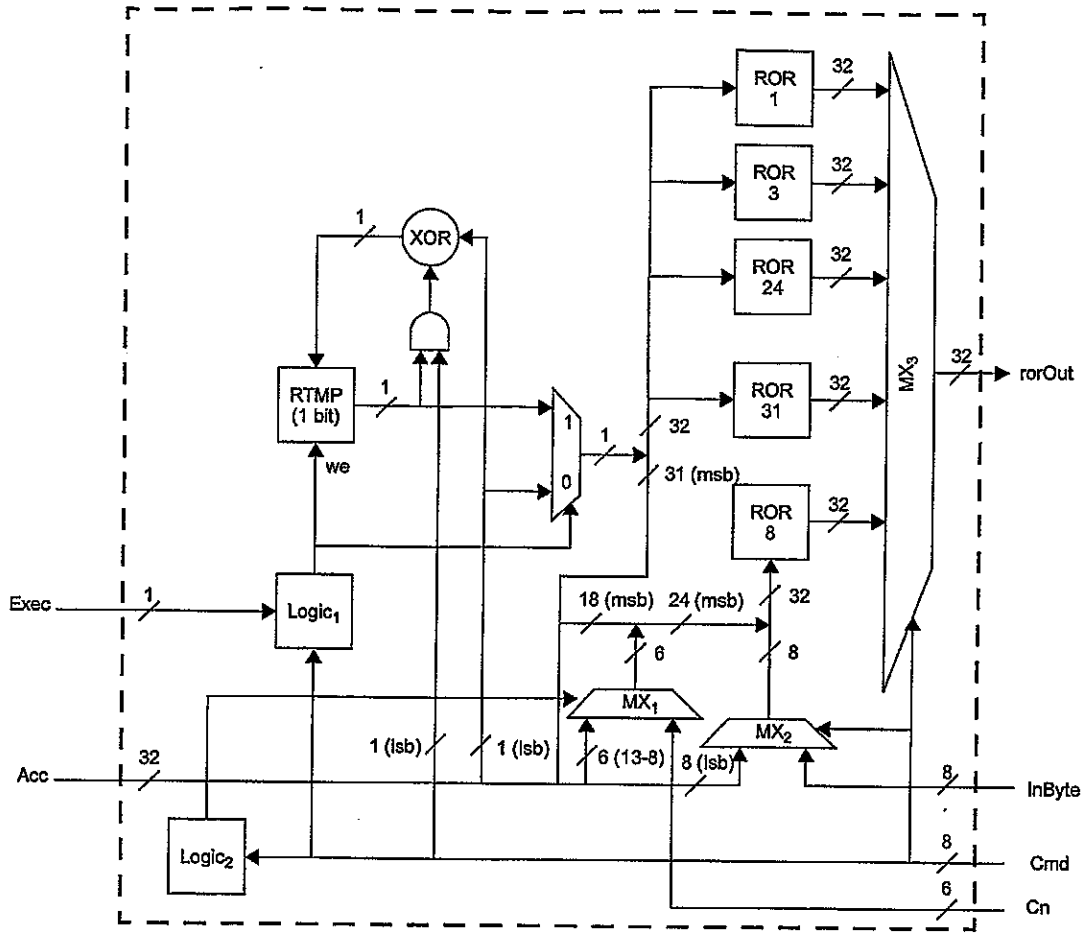

ROR 1
ROR 3
ROR 24
ROR 31
ROR 8
32
XOR
1
RTMP (1 bit)
we
1
0
31 (msb)
MX3
rorOut
Exec
Logic1
18 (msb)
24 (msb)
6
8
MX1
MX2
Acc
1 (lsb)
6 (13-8)
8 (lsb)
InByte
Logic2
Cmd
Cn

FIG. 401

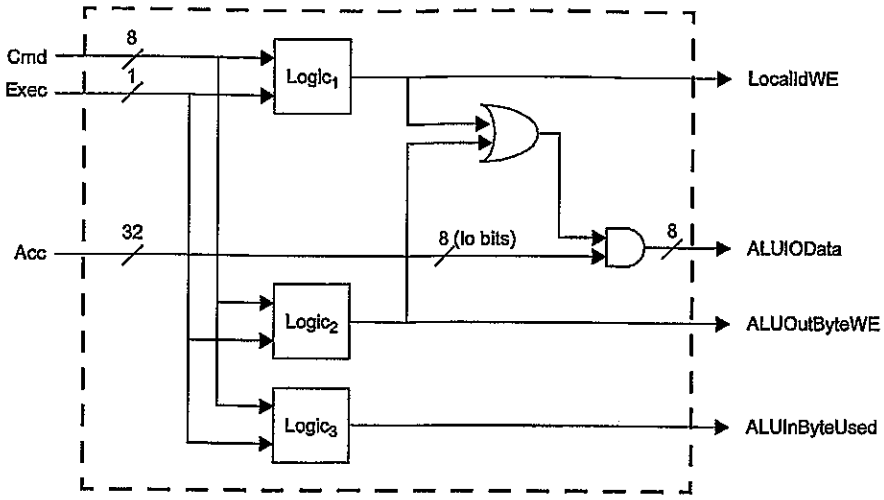

Cmd
Exec
8
1
Logic1
LocalIdWE
Acc
32
8 (lo bits)
8
ALUIOData
Logic2
ALUOutByteWE
Logic3
ALUInByteUsed

FIG. 402

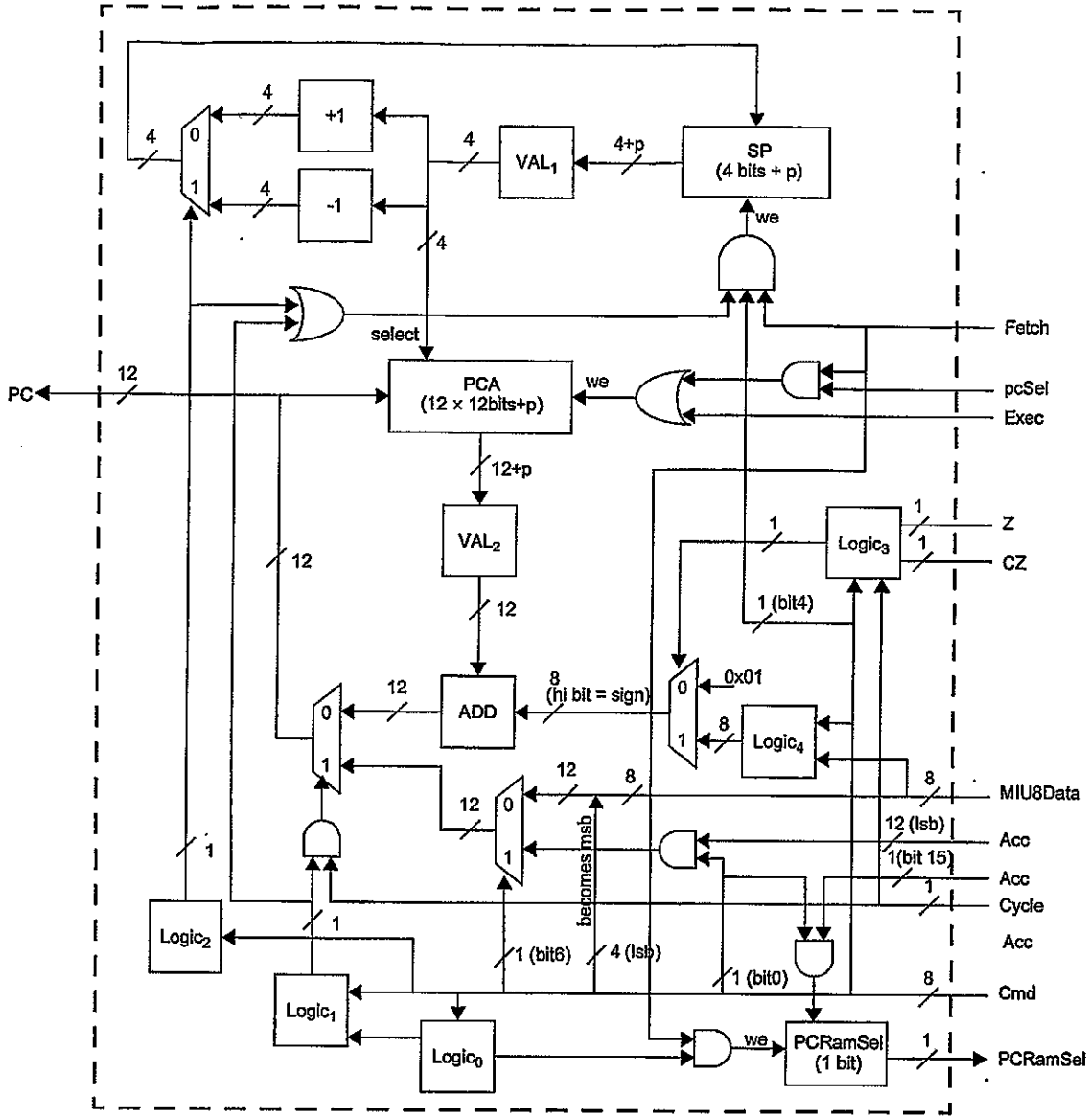

+1
-1
VAL1
4+p
SP
(4 bits + p)
we
select
Fetch
PC
12
PCA
(12 x 12bits+p)
we
pcSel
Exec
12+p
VAL2
Logic3
Z
CZ
1 (bit4)
ADD
8
(hi bit = sign)
0x01
Logic4
MIU8Data
12 (lsb)
Acc
1(bit 15)
Acc
Cycle
becomes msb
Logic2
1 (bit6)
4 (lsb)
Acc
1 (bit0)
Cmd
Logic1
Logic0
we
PCRamSel
(1 bit)
PCRamSel

FIG. 403

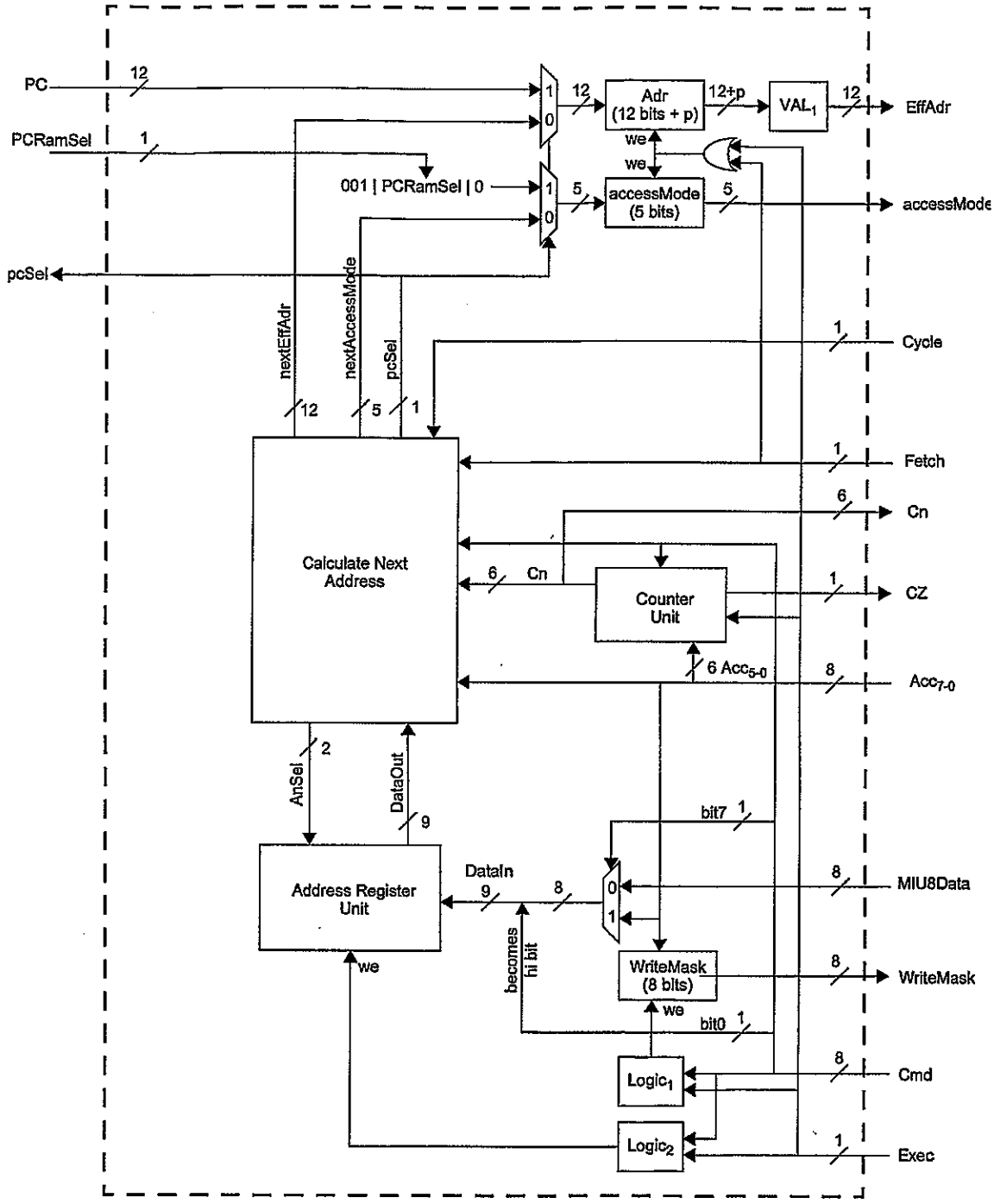

PC
12
PCRamSel
1
001 | PCRamSel | 0
pcSel
nextEffAdr
nextAccessMode
pcSel
12
5
1
Adr
(12 bits + p)
12+p
VAL1
12
EffAdr
we
we
accessMode
(5 bits)
5
5
accessMode
Cycle
Fetch
Cn
6
Calculate Next
Address
6 Cn
Counter
Unit
CZ
6 Acc5-0
Acc7-0
8
AnSel
2
DataOut
9
bit7
Address Register
Unit
DataIn
9
8
MIU8Data
becomes
hi bit
WriteMask
(8 bits)
WriteMask
we
bit0
Logic1
Cmd
Logic2
Exec

FIG. 404

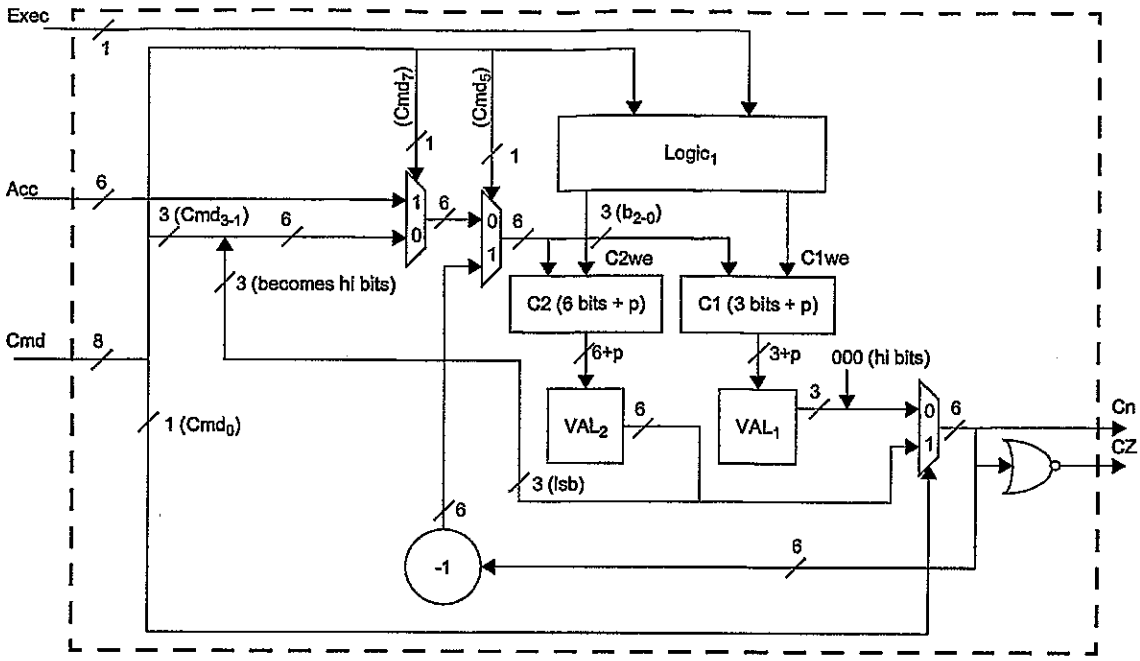


FIG. 405

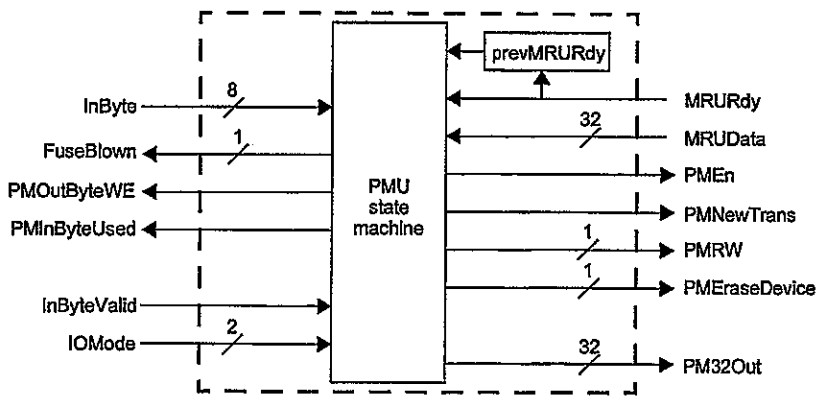


FIG. 406

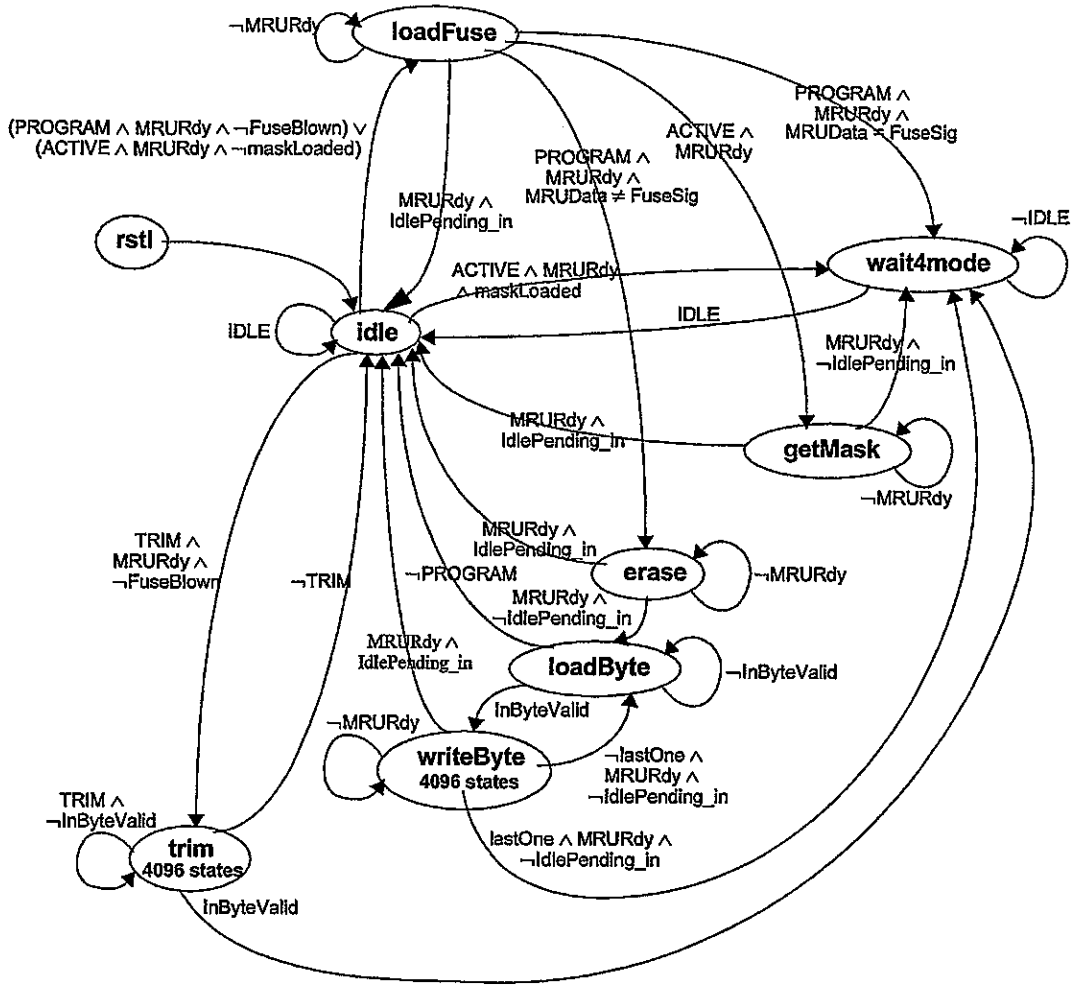

¬MRURdy
loadFuse
(PROGRAM ∧ MRURdy ∧ ¬FuseBlown) ∨
(ACTIVE ∧ MRURdy ∧ ¬maskLoaded)
PROGRAM ∧
MRURdy ∧
MRUData ≠ FuseSig
ACTIVE ∧
MRURdy
PROGRAM ∧
MRURdy ∧
MRUData ≠ FuseSig
MRURdy ∧
IdlePending_in
¬IDLE
rstl
wait4mode
ACTIVE ∧ MRURdy
∧ maskLoaded
IDLE
IDLE
idle
MRURdy ∧
¬IdlePending_in
MRURdy ∧
IdlePending_in
getMask
¬MRURdy
TRIM ∧
MRURdy ∧
¬FuseBlown
¬TRIM
MRURdy ∧
IdlePending_in
¬PROGRAM
erase
¬MRURdy
MRURdy ∧
¬IdlePending_in
MRURdy ∧
IdlePending_in
loadByte
¬InByteValid
InByteValid
¬MRURdy
writeByte
4096 states
¬lastOne ∧
MRURdy ∧
¬IdlePending_in
TRIM ∧
¬InByteValid
trim
4096 states
lastOne ∧ MRURdy ∧
¬IdlePending_in
InByteValid

FIG. 407

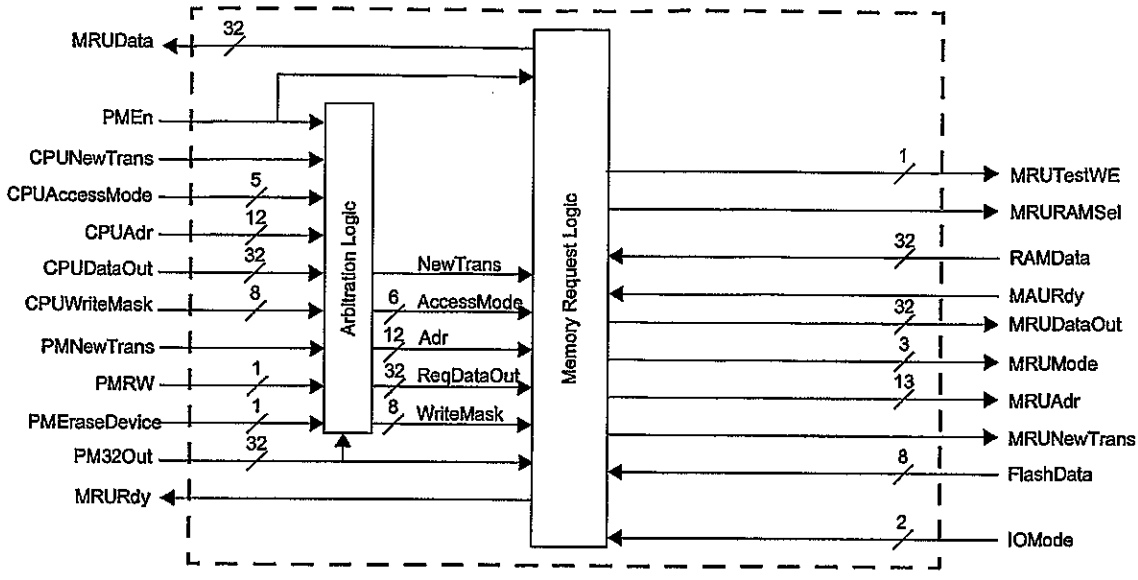

MRUData
32
PMEn
CPUNewTrans
CPUAccessMode
5
CPUAdr
12
CPUDataOut
32
CPUWriteMask
8
PMNewTrans
PMRW
1
PMEraseDevice
1
PM32Out
32
MRURdy
Arbitration Logic
NewTrans
6
AccessMode
12
Adr
32
RegDataOut
8
WriteMask
Memory Request Logic
1
MRUTestWE
MRURAMSel
32
RAMData
MAURdy
32
MRUDataOut
3
MRUMode
13
MRUAdr
MRUNewTrans
8
FlashData
2
IOMode

FIG. 408

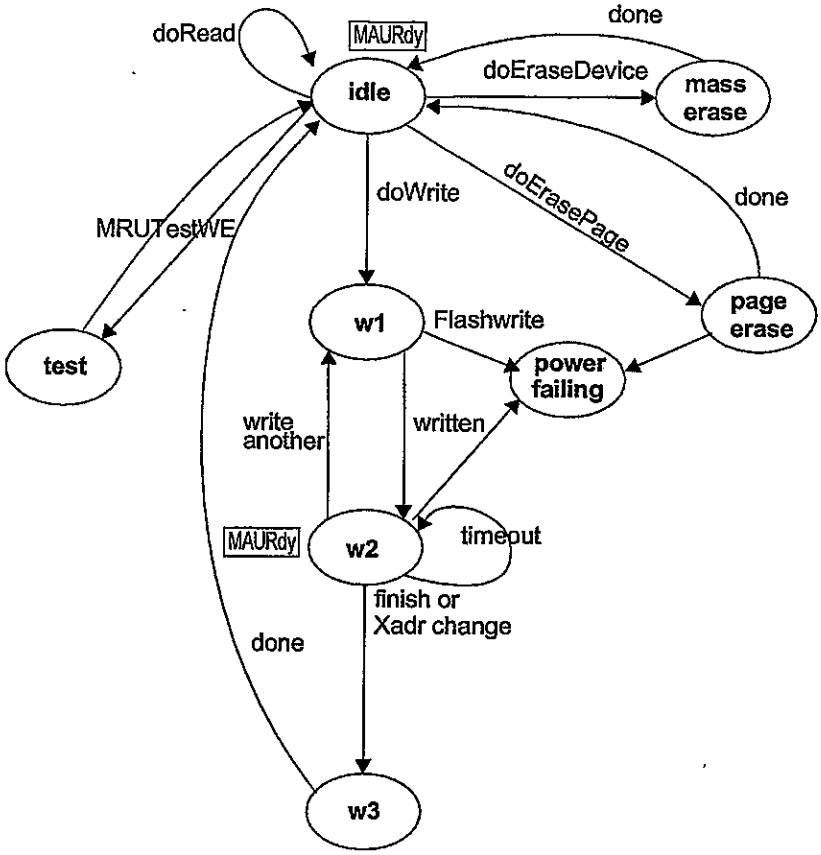


FIG. 409

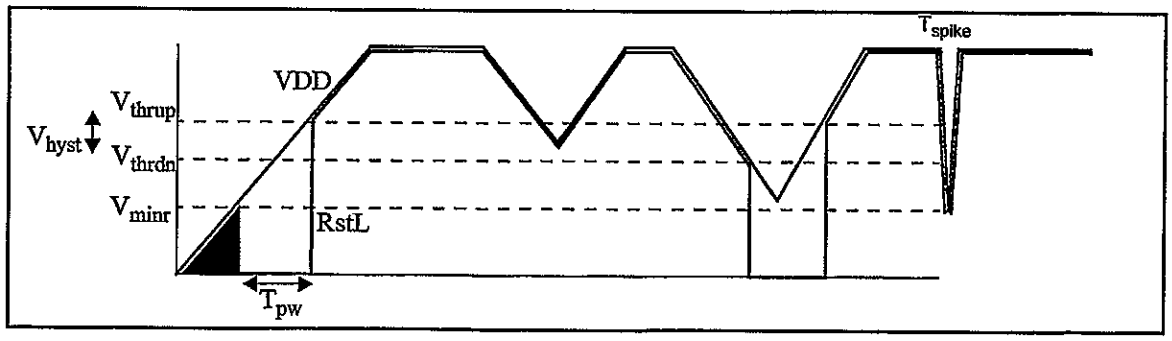


FIG. 410

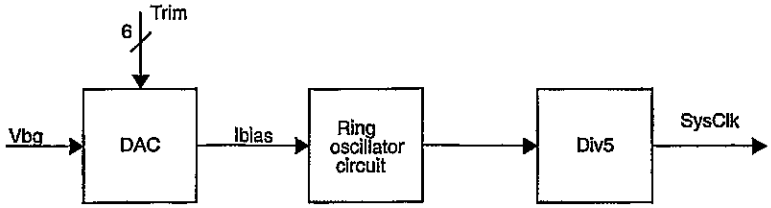


FIG. 411

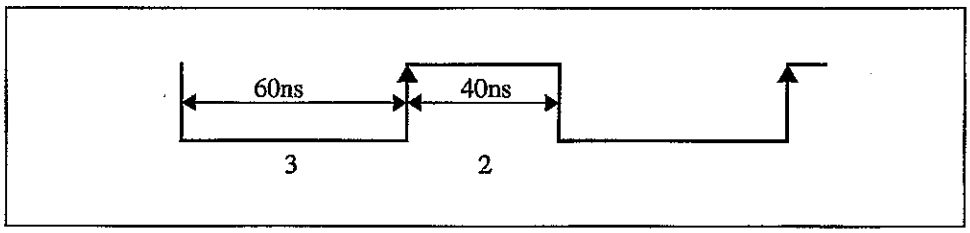


FIG. 412