Capitolul 4

1. MICROCONTROLLER

-Img1

- citește o instrucțiune pe 16 biți din memorie și o execută în 2 tacturi de ceas, în primul tact se decodifică instrucțiunea, iar în al doilea se calculează adresa următoarei instrucțiuni.

Componente interne:

-ALU
-OPERATIONAL_CONTROL
-PROGRAM_FLOW_CONTROL
-RAM_MEMORY
-FLAGS_REGISTER
-REGISTER_BOX

Semnale folosite:

FLAGS_IN, FLAGS_OUT – intrarea, respectiv iesirea registrului de FLAG-uri

INSTRUCTION - instrucțiunea pe 16 bitii care va fi executată

COMAND – comanda pentru comp. PROGRAM_FLOW_CONTROL, ce tip de salt sau simplă incrementare
CONDITION_OF_COMAND – conditia pentru COMAND
INSTRUCTION_ADDRESS – adresa de salt din instructiune

Sel_ALU -
Const_ALU -
Reg1 -
Reg2 -
R1 -
R2 -
Rez -
CR -
ZR -
INTRARE_REGISTRU -
OP_STATE – starea în care se afla componenta OPERATION_CONTROL, 0 decodifică, 1 asteaptă

DATA_IN – intrarea pentru memoria RAM, momentan nefolosită

NEXT_INSTRUCTION_ADDRESS - adresa la care se află următoarea instructiune
MEMORY_ADDRESS - adresa la care se află următoarea instrucțiuni transformată în integer

1.2 OPERATIONAL_CONTROL

-Img2

- această componentă se ocupă de decodificarea instrucțiunii curente pe frontul crescător al tactului de ceas dar doar când se află pe STATE-ul 0, STATE-ul este ieșirea de la un bistabil, numit STATE_SELECTOR care își schimba starea la terminarea unui tact de ceas. În consecință în primul tact decodifică iar în al doilea așteaptă.

Input/Output componentă:

Componente interne:

1.3 PROGRAM_FLOW_CONTROL

Funcționare:

-Img3

- această componentă se ocupă de calcularea adresei instrucțiunii următoare, funcționează doar când valoarea STATE-ului este pe 0. (STATE-ul ei fiind întotdeauna inversul STATE-ului din OPERATIONAL_CONTROL)

-Img4

- se verifică dacă se îndeplinește condiția necesară pentru salt, în funcție de valoarea de adevăr a condiției se stabilește CHIP_SELECT-urile pentru cele PROGRAM_COUNTER și PROGRAM_COUTER_STACK pe primul front descrescător al tactului de ceas.

- în cazul în care condiția este falsă atunci numarătorul va incrementa adresa cu 1, iar STACK-ul nu va face nimic.

-în cazul în care condiția este adevarată și instrucțiunea este de tip:
JUMP - numarătorul este încarcat paralel cu noua adresă din instrucțiune, iar STACK-ul nu va face nimic.
CALL -numarătorul este încarcat paralel cu noua adresă din instrucțiune, iar STACK-ul va salva adresa curentă incrementată cu 1.
RETURN - va fi incarcată paralel în numărător adresa de la ieșirea STACK-ului, iar STACK-ul va încarca pe ieșiri următoarea adresă de retur.
INC – numarătorul va incrementa adresa cu 1, iar STACK-UL nu va face nimic.

Input/Output componentă:

CLK : in – tactul de ceas
RESET : in – semnalul de reset
COMAND : in - tipul saltului sau simplă incrementare

CONDITION_OF_COMAND: in - conditia pentru COMAND
FLAGS_REG : in – valorile FLAG-urilor
INSTRUCTION_ADDRESS : in - adresa de salt din instructiune
NEXT_INSTRUCTION_ADDRESS : out - adresa următoarei instrucțiuni

Componente interne:

- STATE_SELECTOR
- process pentru verificarea conditiei și stabilirea chip select-urilor pentru COUNTER și STACK
- PROGRAM_COUNTER
- PROGRAM_COUNTER_STACK
- POARTA ȘI

Semnale interne:

STATE, NOT_STATE – starea în care se află, 0 execută, 1 așteaptă
SET – folosit pentru resetarea STATE_SELECTORULUI, îl resetează pe valoarea '1' (stand-by)
COUNTER_CLK - semnalul de tact trimis numarătorului
CURRENT_ADDRESS – adresa curentă a instrucțiunii, este ieșirea PROGRAM_COUNTER-ului, este folosită de către PROGRAM_COUNTER_STACK ca adresa actuală pe care sa o urce pe stivă, iar la final va devenii adresa următoarei instrucțiunii
JUMP_ADDRESS – adresa de salt, folosită la incărcarea paralelă a numarătorului
POP_ADDRESS – adresa de ieșire a STACK-ului, adresa de revenire în cazul unui return
STACK_CS – CHIP SELECT-ul pentru PROGRAM_COUNTER_STACK
COUNTER_CS - CHIP SELECT-ul pentru PROGRAM_COUNTER

1.3.1 PROGRAM_COUNTER

Funcționare:

-Img5

- realizează cascadarea a doua număratoare sincrone pe 4 biți pentru a stabili adresă urmatoarei instrucțiuni.

-are 2 regimuri de funcționare, încărcare paralelă, pentru CHIP_SELECT = 0 și numărare crescătoare pentru CHIP_SELECT = 1

Input/Output componentă:

CLK : in – tactul de ceas
RESET: in – semnalul de resetare/inițializare
CHIP_SELECT : in – semnalul de control
JUMP_ADDRESS : in – adresa la care se realizează saltul
NEXT_INSTRUCTION_ADDRESS : out – adresa instrțiunii curente/următoarei instrucțiuni în funcție de momentul în care ne aflam

Componente interne:

- 2 numaratoare pe 4 biti (COUNTER)
- 1 multiplexor 2:1 pentru alegerea CHIP_SELECT-ului numărătorului superior
- 1 multiplexor 2:1 pentru alegerea valorii pentru incărcarea paralelă pentru număratorul superior

Semnale interne:

COUNTER0_CHIP_SELECT, COUNTER1_CHIP_SELECT
COUNTER1_PARALEL_LOAD
COUNTER0_CARRY, COUNTER1_CARRY
Q – Ieșirile concatenate ale celor 2 număratoare, care formează adresa pe 8 biți.

1.3.1.1 COUNTER

Funcționare:

- este un numărator pe 4 biți sincron, cu 2 regimuri de funcționare, pentru CHIP_SELECT = 0, încărcare paralelă, respectiv CHIP_SELECT = 1 incrementare cu 1 pe frontul crescător al tactului de ceas.

Input/Output componentă:

RESET : in – semnalul de reset
CLK : in
CHIP_SELECT : in
PARALEL_LOAD : in
Q : buffer – ieșirile bistabilelor concatenate într-o magistrală
CARRY : out – semnalul de carry out, când toate cele 4 bistabile sunt pe ‘1’ logic

Componente interne:

- 4 bistabile JK
-porți logice pentru realizarea logicii, inversoare, porți ȘI, porți SAU

Semnale interne:

J3, K3, J2, K2, J1, K1, J0, K0 – intrările de control al bistabilelor
notCS, notPL3, notPL2, notPL1, notPL0 - valorile negate ale intrarilor, CHIP_SELECT și PARALEL_LOAD
- restul semnalelor sunt pentru legarea porților logice, având nume simbolice pentru operația logica efectuată și operanzi

-Img6

-Img7

1.3.2 PROGRAM_COUNTER_STACK

Funcționare:

-Img5

- salvează pe stivă adresă curentă incrementată cu 1

- are 3 regimuri de funcționare în funcție de CHIP_SELECT:

= 00 echivalent cu funcția de RETURN, pune pe ieșire adresa care se află pe poziția STACK_POINTER-ului în memoria de tip stivă și incrementează STACK_POINTER-ul

= 11 echivalent cu funcția de CALL, încarcă paralel numaratorul cu adresa curentă, după care o incrementează cu 1 și decrementează STACK_POINTER-ul, după care scrie adresa incrementată la adresa STACK_POINTER-ului în memorie

= 10 sau 01 componenta nu face nimic

Input/Output componentă:

CLK : in – tactul de ceas
RESET : în - semnalul de reset
STATE : in – starea în care se află componenta
CHIP_SELECT : in
PUSH_ADDRESS: in – adresa care trebuie urcată pe stivă
POP_ADDRESS: out – adresa care este scoasă de pe stivă

Componente interne:

- PROGRAM_COUNTER -pentru incrementarea adresei curente
- process (POINTER) – care decrementeaza/incrementeaza STACK_POINTER-ul în funcție de momentul de timp și CHIP_SELECT
- STACK_MEMORY – memorie RAM folosită drept stivă
- Porți logice de tipul inversor și poarta ȘI

Semnale interne:

NOT_CLK, NOT_STATE – valorile negate ale CLK-ului și ale STATE-ului, folosite pentru logică
COUNTER_CS, WRITE_STROBE – CHIP_SELECT-ul pentru numărător și WRITE_STROBE-ul pentru memoria RAM
COUNTER_ADDRESS, MEMORY_DATA_OUT - ieșirea de la numărător (adresa incrementată)
STACK_POINTER – adresa la care se scrie / se citește următoarea adresă de instrucțiune