סימולציה 3 יבש:

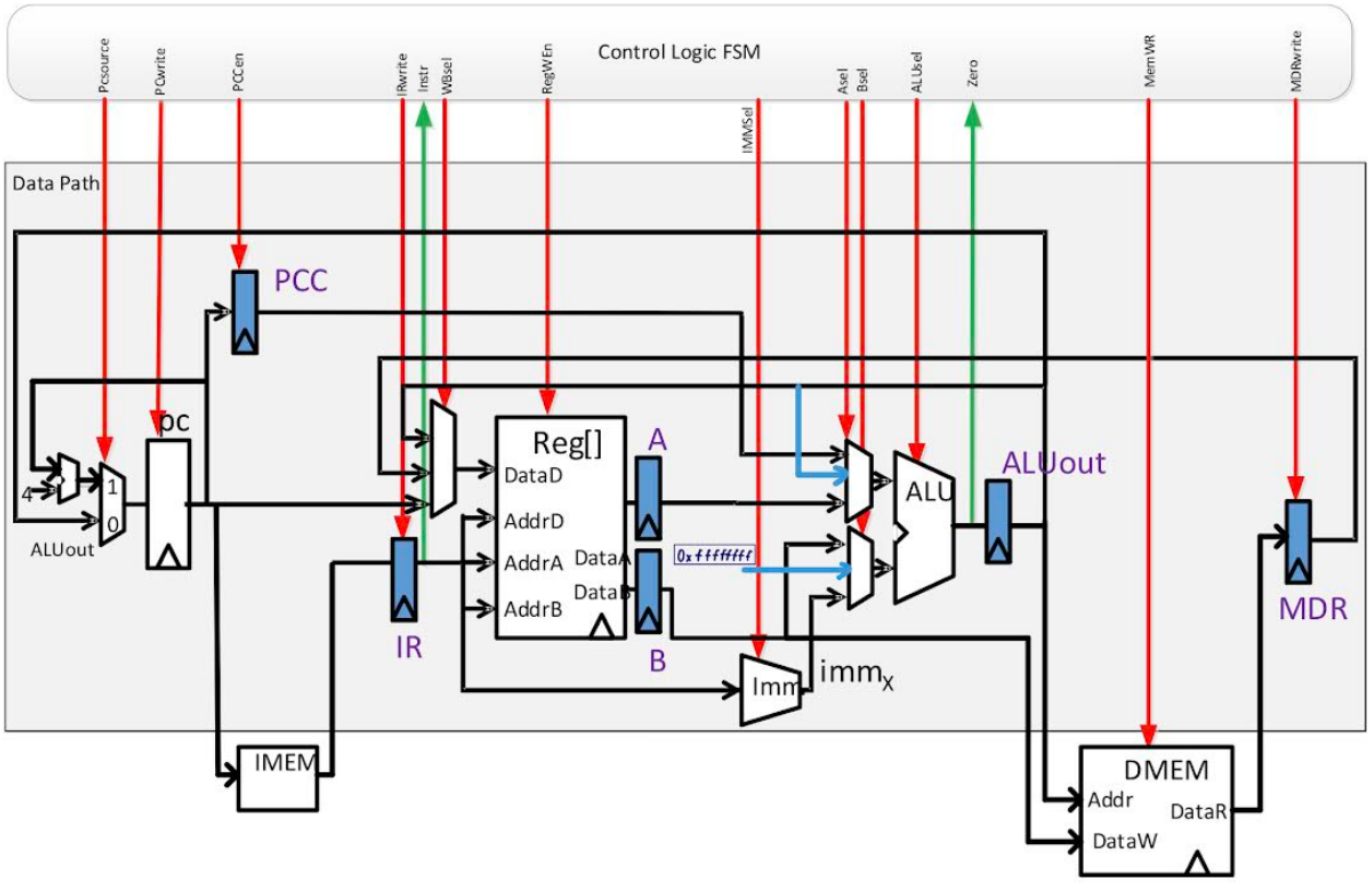
מגישים:

עידו טאוסי	214008997
נעם ביטון	213745953

סעיף 2.1:

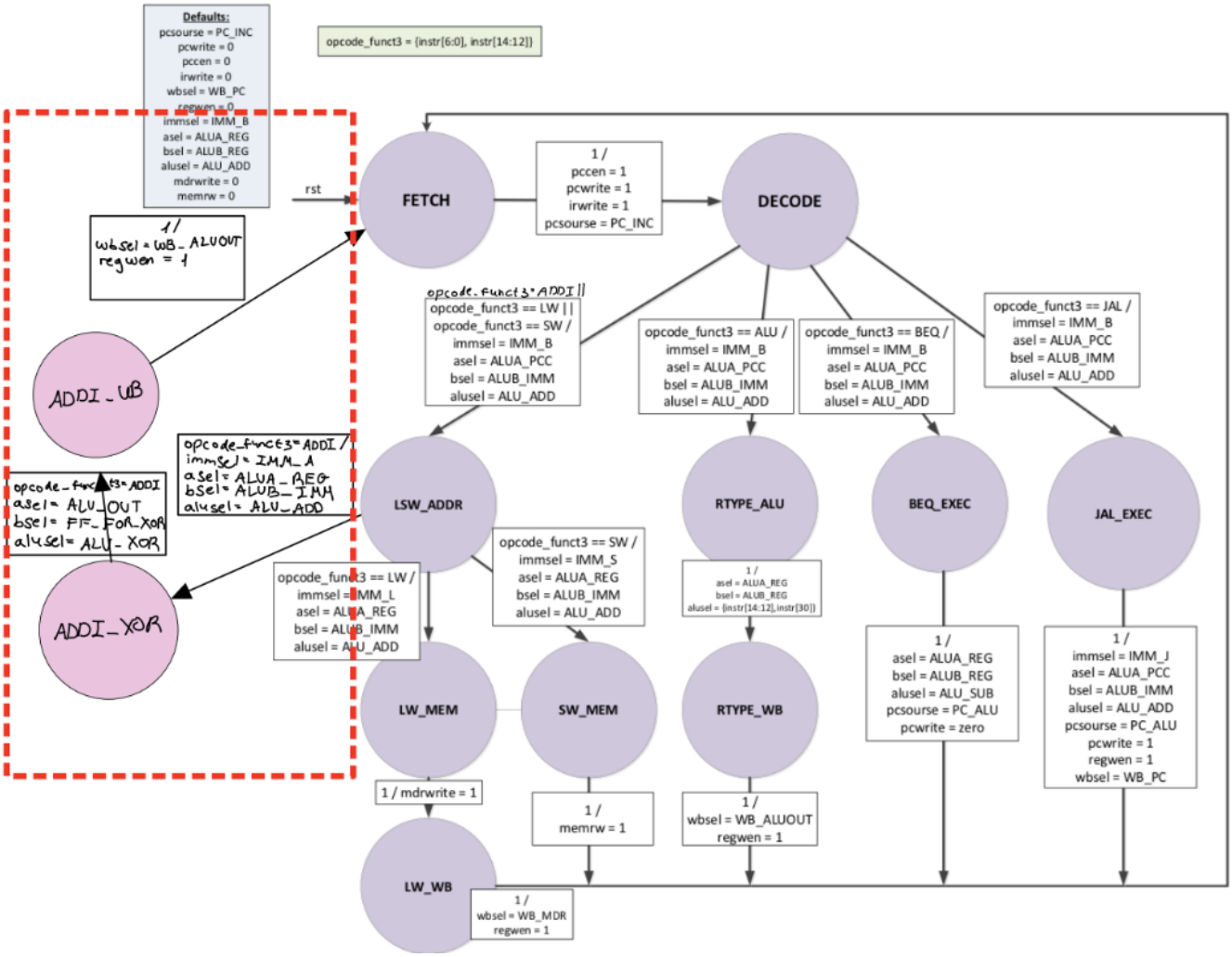
השינויים שהוספנו במעבד הם (מופיעים בכחול):

- הוספת חיבור $ALUout$ אל הבורר של A.
- הוספת ערך $0xffffffff$ אל הבורר של B.



נשים לב שהפקודה שלנו, וגם הפקודות lw, sw, מקבלות 2 רגיסטרים, וערך immediate, לכן אם נרצה לבצע $addi$, נתחיל בללכת ממצב Fetch, למצב Decode, למצב LSW_ADDER, ומשם לשני מצבים חדשים:
(ADDI_XOR, ADDI_WB), והמעברים החדשים שהגדרנו, יגדירו את החיבור, ה-XOR, והכתיבה חזרה לרגיסטר התוצאה rd וחזרה למצב 0.

במעבר למצב $ADDI\_XOR$ נבצע את החישוב $R[rs1] + imm$ ובמעבר למצב $ADDI\_WB$ נבצע את החישוב $(R[rs1] + imm)$^$0xffffffff$ ולבסוף במעבר חזרה ל-$FETCH$ נבצע את הכתיבה של התוצאה לרגיסטר.


Defaults:
pcsourse = PC_INC
pcwrite = 0
pccen = 0
irwrite = 0
wbsel = WB_PC
regwen = 0
immsel = IMM_B
asel = ALUA_REG
bsel = ALUB_REG
alusel = ALU_ADD
mdrwrite = 0
memrw = 0
opcode_funct3 = {instr[6:0], instr[14:12]}
rst
FETCH
1 /
pccen = 1
pcwrite = 1
irwrite = 1
pcsourse = PC_INC
DECODE
opcode_funct3 == LW || opcode_funct3 == SW /
immsel = IMM_B
asel = ALUA_PCC
bsel = ALUB_IMM
alusel = ALU_ADD
opcode_funct3 == ALU /
immsel = IMM_B
asel = ALUA_PCC
bsel = ALUB_IMM
alusel = ALU_ADD
opcode_funct3 == BEQ /
immsel = IMM_B
asel = ALUA_PCC
bsel = ALUB_IMM
alusel = ALU_ADD
opcode_funct3 == JAL /
immsel = IMM_B
asel = ALUA_PCC
bsel = ALUB_IMM
alusel = ALU_ADD
LSW_ADDR
RTYPE_ALU
BEQ_EXEC
JAL_EXEC
opcode_funct3 == LW /
immsel = IMM_L
asel = ALUA_REG
bsel = ALUB_IMM
alusel = ALU_ADD
opcode_funct3 == SW /
immsel = IMM_S
asel = ALUA_REG
bsel = ALUB_IMM
alusel = ALU_ADD
1 /
asel = ALUA_REG
bsel = ALUB_REG
alusel = {instr[14:12],instr[30]}
1 /
asel = ALUA_REG
bsel = ALUB_REG
alusel = ALU_SUB
pcsourse = PC_ALU
pcwrite = zero
1 /
immsel = IMM_J
asel = ALUA_PCC
bsel = ALUB_IMM
alusel = ALU_ADD
pcsourse = PC_ALU
pcwrite = 1
regwen = 1
wbsel = WB_PC
LW_MEM
SW_MEM
RTYPE_WB
1 / mdrwrite = 1
1 /
memrw = 1
1 /
wbsel = WB_ALUOUT
regwen = 1
LW_WB
1 /
wbsel = WB_MDR
regwen = 1
1 /
wbsel = WB_ALUOUT
regwen = 1
ADDI_WB
opcode_funct3 == ADDI /
immsel = IMM_A
asel = ALUA_REG
bsel = ALUB_IMM
alusel = ALU_ADD
opcode_funct3 == ADDI
asel = ALU_OUT
bsel = FF_FOR_XOR
alusel = ALU_XOR
ADDI_XOR

סעיף 2.3:

ניתן לראות כי לאחר ההרצה, נוצר קובץ dmem_out.hex שמכיל בשורה החמישית החל מסיום ההערות בכותרת, את הערך $fffff406$.

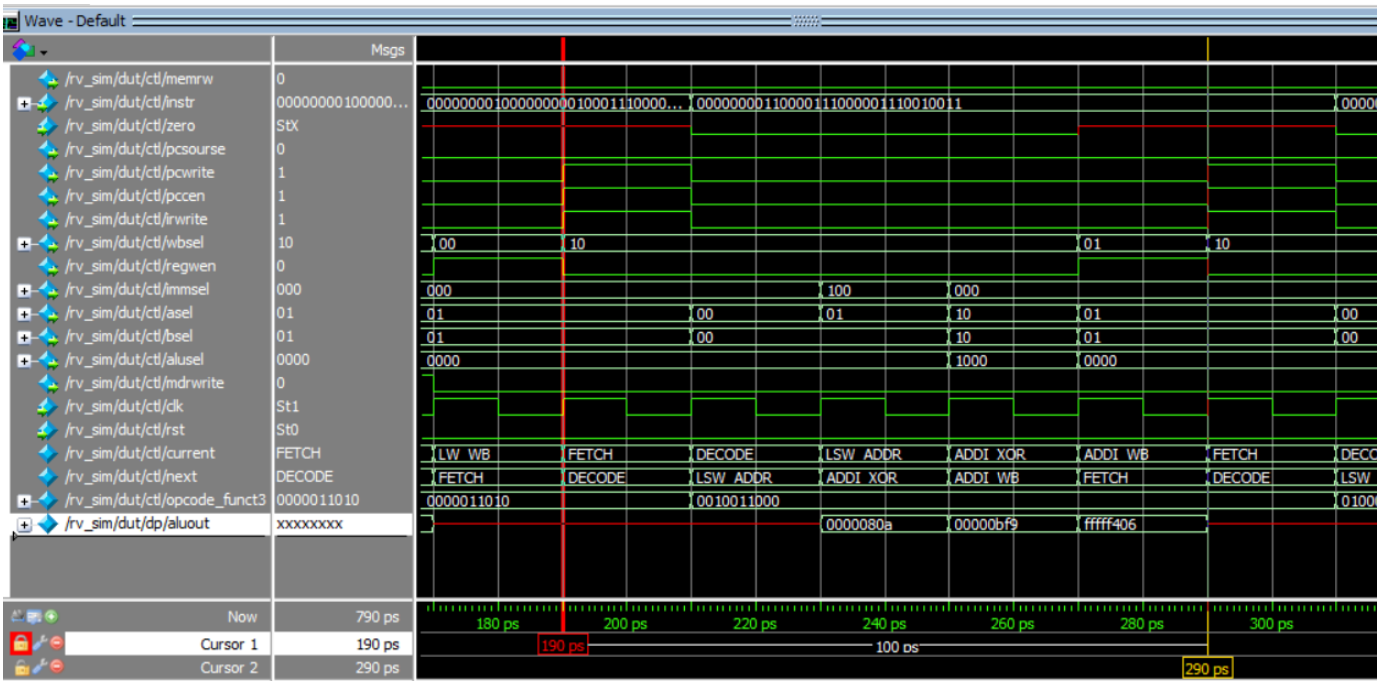
```
dmem_out.hex
// memory data file (do not edit the following line - required for mem load use)
// instance=/rv_sim/dmem
// format=hex addressradix=h dataradix=h version=1.0 wordsperline=1 noaddress
0000ff00
0000dead
00000bed
0000feed
fffff406
xxxxxxxx
xxxxxxxx
xxxxxxxx
xxxxxxxx
```

ניתן לראות כי ערך ה-pc מתחיל ב-0 וגדל ב-4 כל פעם שהוא מסיים פקודה ועובר לפקודה הבאה (השורה הבאה בקוד) כמצופה.

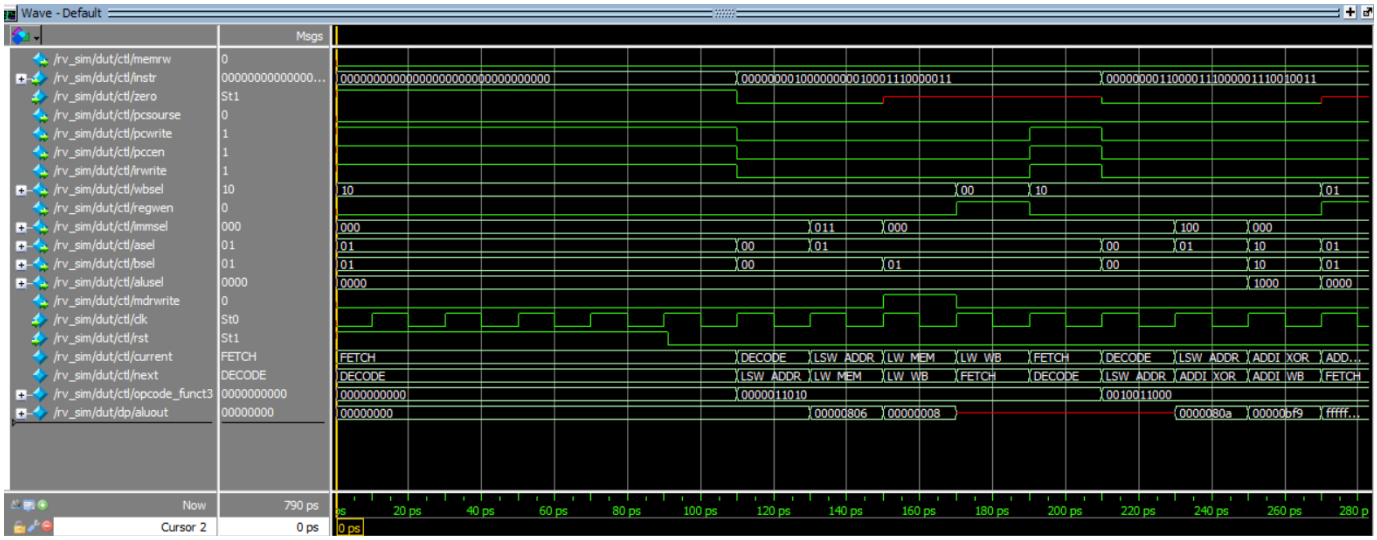
בנוסף, ערך ה-$aluot$ מקבל את תוצאת פעולת החיבור ולאחר מכן מקבל את תוצאת פעולת ה-xor.

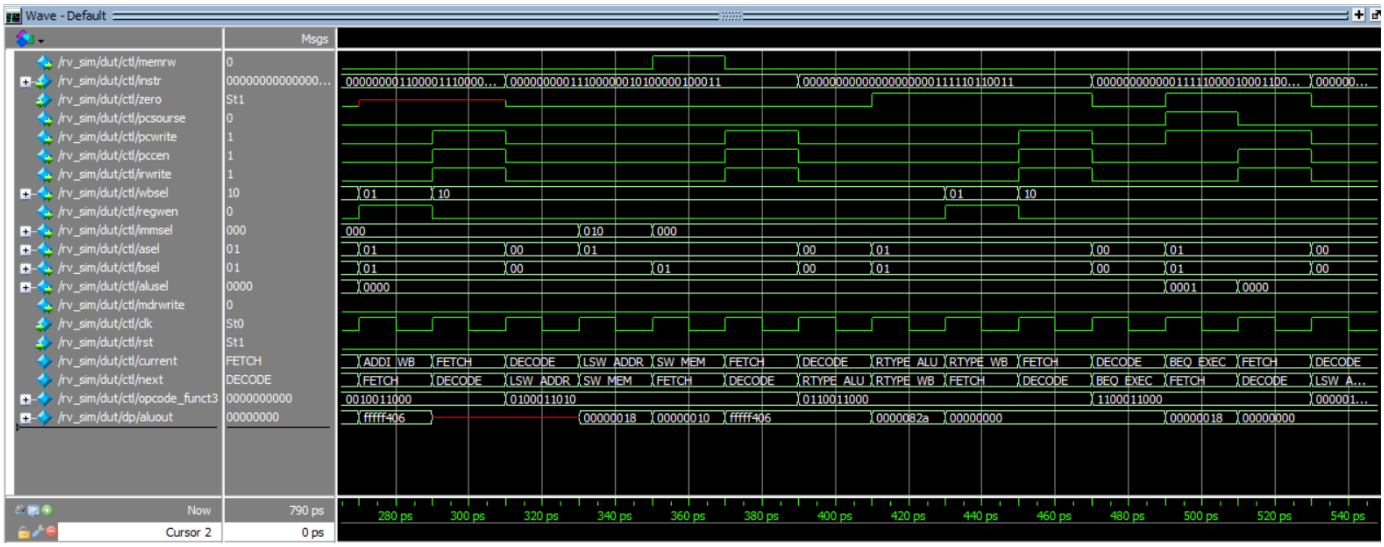
ניתן לראות את המעבר בין המצבים כפי שתואר בסעיף 2.1 במכונת המצבים שהצענו. שינוי זה גורר גם שינוי באותות הבקרה כפי שניתן לראות בדיאגרמת הגלים.
בפרט, ניתן לראות כי לאחר $190ps$, כאשר מתקבלת פקודת $addi$ המעבר בין המצבים מתחיל, באופן הבא:
$FETCH \rightarrow DECODE \rightarrow LSW\_ADDR \rightarrow ADDI\_XOR \rightarrow ADDI\_WB$ ולאחר מכן חוזר למצב $FETCH$ עד לקבלת פקודת ה-$ADDI$ הבאה.
בנוסף, ידוע לנו כי פקודת $ADDI$ תעבור ב-5 מצבים עד שתסיים את הפעולה, ולכן ניתן לראות שאכן עוברים 5 מחזורי שעון עד שהפעולה מסתיימת.

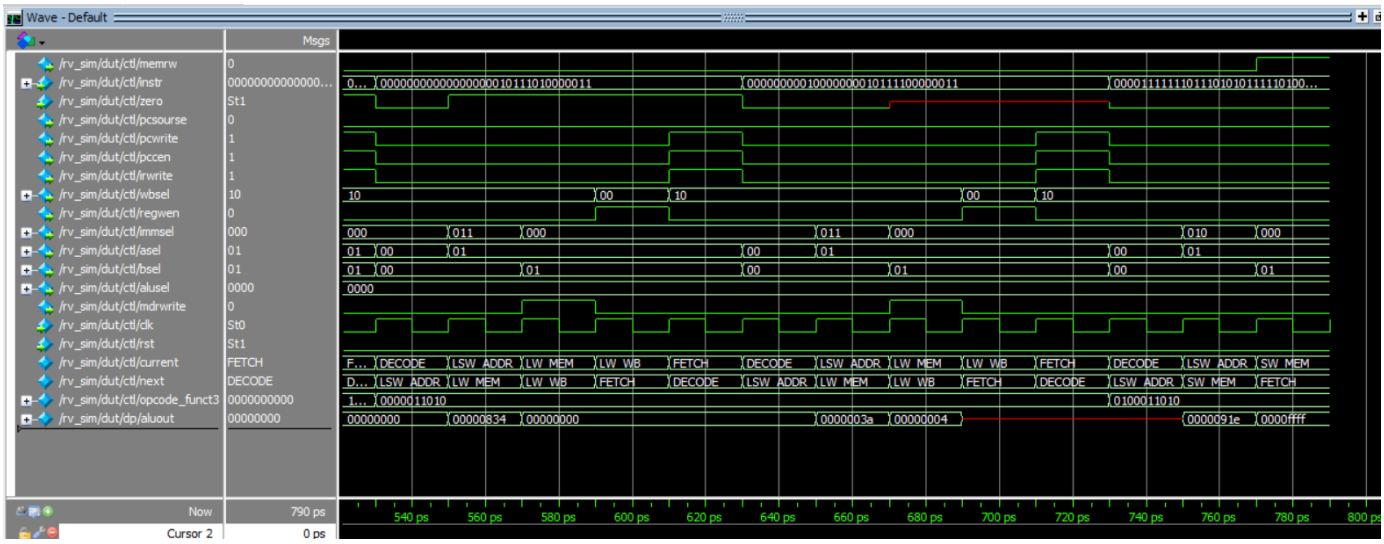
צילום מסך של דיאגרמת הגלים בזמן פקודת $ADDI$:



<u>צילום מסך מחולק לחלקים שבהם ניתן לראות את המעבר של המצבים של כל הפקודות בתוכנית:</u>







צילום מסך מלא: