Rekonfigurovateľné elektronické systémy

32-bit CPU

Vypracoval: Matej Novák
Dátum: 7.12.2020

Zadanie úlohy:
Design and describe (e.g. in Verilog language) a processor that has the following attributes and performs the following functionality. Data are 32-bit, addresses of data and addresses of instructions are 32-bit, the instructions are 32-bit as well, and the amount of registers is 64. Use Harvard architecture, i.e. the processor communicates with 2 memories, data memory and instruction memory. The size of memories is the maximum possible according to the address size for the given memory.
Top module of the processor has to be named as CPU.
Inputs of CPU are:
- 1-bit CLK - clock
- 1-bit RST - reset
- 32-bit DATA_IN - data coming from data memory
- 32-bit INSTR_IN - instructions coming from instruction memory

Outputs of CPU are:
- 32-bit DATA_OUT - data for data memory
- 32-bit DADDR - address for data memory
- 1-bit DWR - write/read for data memory (1 is write, 0 is read)
- 1-bit DEN - enable signal for data memory (1 is enabled, 0 is disabled)
- 32-bit IADDR - address for instruction memory
- 1-bit IEN - enable signal for instruction memory (1 is enabled, 0 is disabled)

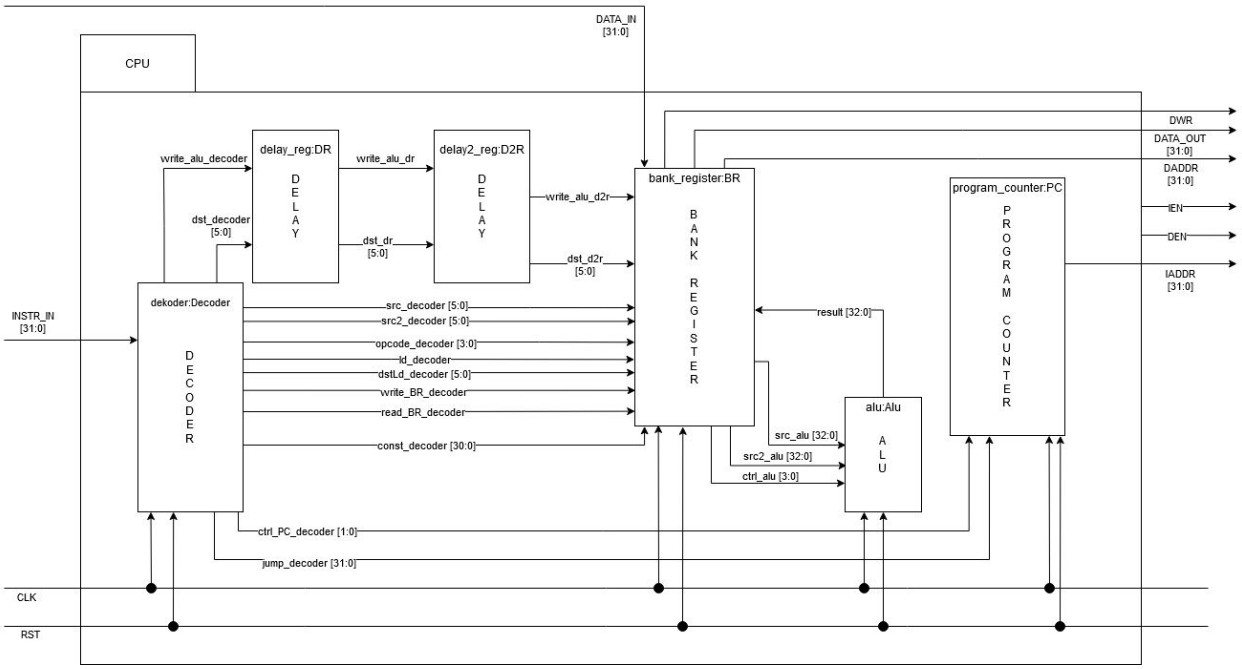
Projekt má pracovať s **tabuľkou inštrukcii** zobrazenou nižšie:

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NOP	0	0	0	0	0	0	0	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
ADD	0	0	0	0	0	0	1	x	x	x	x	x	x	x	dst						src2						src					
SUB	0	0	0	0	0	1	0	x	x	x	x	x	x	x	dst						src2						src					
AND	0	0	0	0	0	1	1	x	x	x	x	x	x	x	dst						src2						src					
OR	0	0	0	0	1	0	0	x	x	x	x	x	x	x	dst						src2						src					
XOR	0	0	0	0	1	0	1	x	x	x	x	x	x	x	dst						src2						src					
SHROL	0	0	0	0	1	1	0	x	x	x	x	x	x	x	dst						src2						rot	const				
SHROR	0	0	0	0	1	1	1	x	x	x	x	x	x	x	dst						src2						rot	const				
WR	0	0	0	1	0	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	addr						src					
RD	0	0	0	1	1	x	x	x	x	x	x	x	x	x	dst						addr						x	x	x	x	x	x
LD	0	1	const[23:12]												dst						const[11:0]											
JMP	1	const																														
BEQ	0	0	1	0	const																addr						src					
BNE	0	0	1	1	const																addr						src					

1. Hlavný modul

Hlavný modul projekto je nazvaný CPU.v. Tento modul obsahuje 4 vstupy: clk,RST,DATA_IN a INSTR_IN. V úvode sa modul inicializuje a nastaví inštrukciu na INSTR_IN na hodnotu 32'b0. To spôsobí načítanie prvej inštrukcii na danej adrese do submodulu Decoder, ktorý danú inštrukciu vyhodnotí. Rozdelujeme 2 typy inštrukcii - dlhé a krátke. Krátke sa vykonávajú ihneď v druhom cykle, dlhé sú aritmeticko-logické operácie, ktore prechádzajú cez ALU jednotku. Vzhľadom k faktu, že aritmeticko-logické operácie rozdeľujeme v tomto projekte na viac cyklov (čítanie z banky registrov, odoslanie hodnotôt do submodulu ALU, vykonanie operácie a odoslanie výsledku späť na ALU), implementovali sme si dva submoduly, ktorých cieľom je spomaliť príchod registra dst_decoder a write_alu_decoder o dva cykly tak, aby sa nachádzali v banke registrov presne v momente, keď zo submodulu ALU príde výsledok operácie. Výstupom hlavného modulu sú DATA_OUT, ktoré nesú informácie o výstupných dátach, DADDR, ktoré obsahujú adresu externej pamäte, DWR, ktorá hovorí o povolení čítania na externej pamäti a DEN a IEN, ktoré mojím návrhom len prechádzajú, nakoľko register IEN je v rámci návrhu vždy rovný 1.

Diagram celého modulu CPU môžeme vidieť na obrázku nižšie.



2. Submoduly

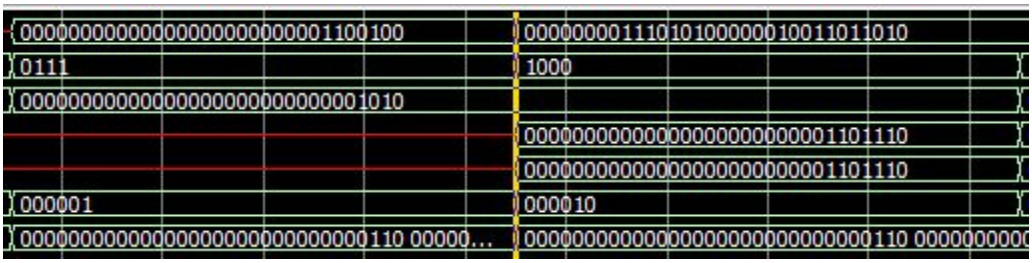
V projekte sme použili nasledovné submoduly:

Opis submodulu	**Označenie v diagrame a programe**
Aritmeticko-logická jednotka	alu
Banka registrov	bank_register
Dekóder	decoder
Oneskorenie signálu o jeden cyklus clk	delay_reg
Oneskorenie signálu o dva cykly clk	delay2_reg
Program counter	program_counter

● Aritmeticko-logická jednotka (alu)

Názov	Input/Output	Veľkosť [bit]
clk	Input	1
RST	Input	1
src	Input	32
src2	Input	32
opcode	Input	4
result	Output	32

Má za úlohu vykonávať aritmetické operácie, ktoré prichádzajú skrz inštrukciu do dekódera, kde sme si extrahovali časť bitov, ktoré obsahujú informáciu o tom, aká operácia sa s danými hodnotami (získanými taktiež z inštrukcii ako dve šesť bitové čísla, ktoré odkazujú na adresu registra) má vykonať. Tento register je následne poslaný na banku registra spolu s registrami src a src2, ktoré následne v banke registra cez registre pamäte odkážu na hodnotu adresy, ktorá sa má poslať do alu submodulu. V ďalšom cykle sa tieto hodnoty za pomoci opcodu vykonajú (ADD,SUB,AND,OR,XOR) a výsledok sa uloží na register result, ktorý je následne poslaný do banky registra.



Na obrázku môžeme vidieť príklad operácie - 2 takty oddelené žltou vertikálnou čiarou. V ľavej časti na 1. Riadku môžeme vidieť priebeh prvej hodnoty (src_alu), na treťom riadku prieheb druhej hodnoty (src2_alu), medzi nimi na druhom riadku vidíme opcode (ctrl_alu), ktorá predstavuje operáciu ADD, teda sčítanie. Na prevej strane potom vidíme v 4. Riadku výsledok (result) odosielaný do bank registra, ktorý bude uložený na adresu, ktorú vidíme v 5. Riadku, teda 000010.

- Bank register

Názov	Input/Output	Veľkosť [bit]
clk	Input	1
RST	Input	1
DATA_IN	Input	32
src_in	Input	6
src2_in	Input	6
dst_in	Input	6
dstLd_in	Input	6
const_in	Input	31
opcode	Input	4
ld	Input	1
write	Input	1
read	Input	1
write_ALU	Input	1
src_alu	Output	32
src2_alu	Output	32
ctrl_alu	Output	4
DATA_OUT	Output	32

DADDR	Output	32
DWR	Output	1

Submodul príjma registre z dekódera, ktoré následne spracúva podľa kontrolných bitov, ktoré do modulu prichádzajú.
Ak je na bank register privedená inštrukcia LOAD (ld), hodnota inštrukcie sa zapíše na register s adresou neoskorenej destinácie (dst_decoder) prichádzajúcej z dekódera spolu s konštantou, ktorá má byť na register zapísaná.
Ak je aktívny kontrolný bit ALU (write_ALU), do registra sa zapisuje result aritmetickej operácie z modulu alu, ktorá sa ukladá do oneskorenej destinačnej adresy (dst_d2r).
Zápis do registra (WR), v projekte privedený ako register write_BR_decoder, prebieha formou zápisu hodnoty registra na adrese registra (src_decoder) do registra DATA_OUT, pričom register DADDR bude rovná addrese, kam sa má hodnota v pamäti uložiť. Ak chceme pristúpiť k zápisu v externej pamäti, musíme nastaviť register DWR na hodnotu 1.
Čítanie z dátovej pamäte (RD) prebieha privedením hodnoty z registra DATA_IN. V prípade, že do dekodera príde inštrukcia RD, do banky registra sa odošle adresa a destinácia z inštrukcie, taktiež sa nastaví kontrolný bit read na 1. Následne sa v banke registrov odošle adresa registra v pamäti a aktivuje sa counter, ktorý čaká, kým príde na vstup hodnota DATA_IN, ktorú priradí k registru podľa destinačnej adresy.

- Dekoder

Názov	Input/Output	Veľkosť [bit]
INSTR_IN	Input	32
clk	Input	1
RST	Input	1
dst	Output	1
dstLd	Output	32
src	Output	6
src2	Output	6
const	Output	6
opcode	Output	6
jump	Output	31
ctrl_PC	Output	4

write_BR	Output	1
read_BR	Output	1
write_ALU	Output	1
ld	Output	1

Na vstupe získa 32-bitovú inštrukciu, ktorú má za úlohu rozparsovať a získať potrebné údaje na základe ktorých bude prebiehať ďalšie korektné vykonávanie príkazov v submoduloch. V module kontrolujeme 7-bitovú časť INSTR_IN[31:25], cez ktorú následne určujeme, či ide o aritmeticko-logickú operáciu, skok na inú konkrétnu inštrukciu, uloženie konštanty do registra, zápis/čítanie do/z externej pamäte alebo skok na iný branch.

- Program counter

Názov	Input/Output	Veľkosť [bit]
clk	Input	1
RST	Input	1
data_in	Input	32
dctrl	Input	2
jump	Input	32
IADDR	Output	32

Modul pracuje v dvoch módoch. Buď prebehne zvýšenie inštrukcie o jedna (bežná inštrukcia) alebo o konštantu (inštrukcia jump). Inštrukciu skoku o konštantu, ak je rovná adresa v registri sa mi nepodarilo implementovať.

- Delay register

Názov	Input/Output	Veľkosť [bit]
clk	Input	1
RST	Input	1
dst	Input	6
write_alu	Input	1

dst_delay	Output	6
write_alu_delay	Output	1

Vytvorí oneskorenie signalu o jeden hodinový signál clk pre registre dst a write_alu, ktoré vychádzajú z dekodera a slúžia na aritmeticko-logické operácie.

- Delay2 register

Názov	Input/Output	Veľkosť [bit]
clk	Input	1
RST	Input	1
dst	Input	6
write_alu	Input	1
dst_delay2	Output	6
write_alu_delay2	Output	1

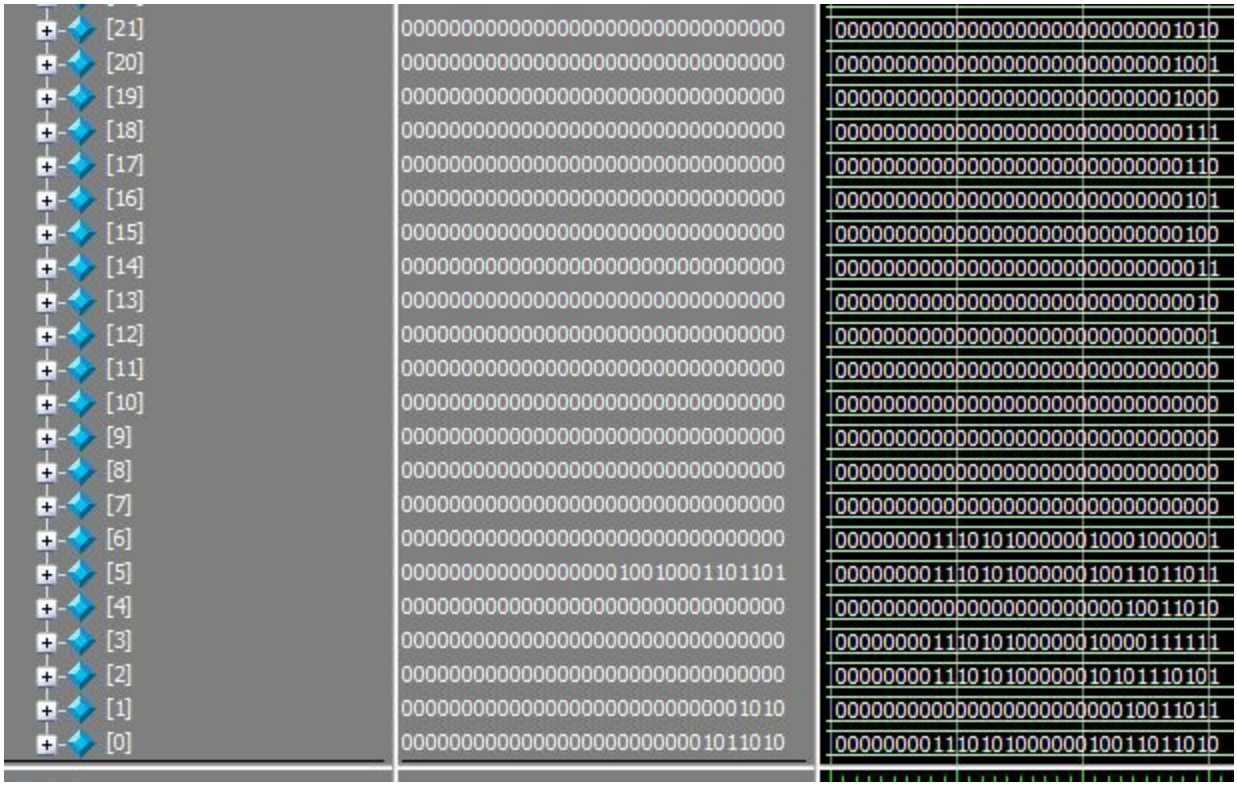
Vytvorí oneskorenie signalu o ďalší jeden hodinový signál clk. Výsledkom je oneskorenie signálu o 2 takty clk. Dôvod oneskorenia je totožný ako pri Delay registry, potrebujeme oneskoriť takt o 2 takty, preto boli implementované dva registre.

3. Simulácie

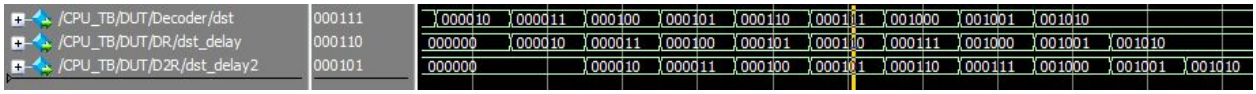
Príklady vykonávania ALU operácii v čase - na obrázkoch môžeme vidieť vykonávané inštrukcie z modulu CPU_TB, konkrétne od inštrukcie ‘instr[18]’, teda ADD:

Priebeh zápisu (LOAD - LD) konštanty (const_in) na adresu registra (REGISTERS[dstLd_in]). Taktiež môžeme vidieť zmenu kontrolného bitu (ld) v prvom riadku:

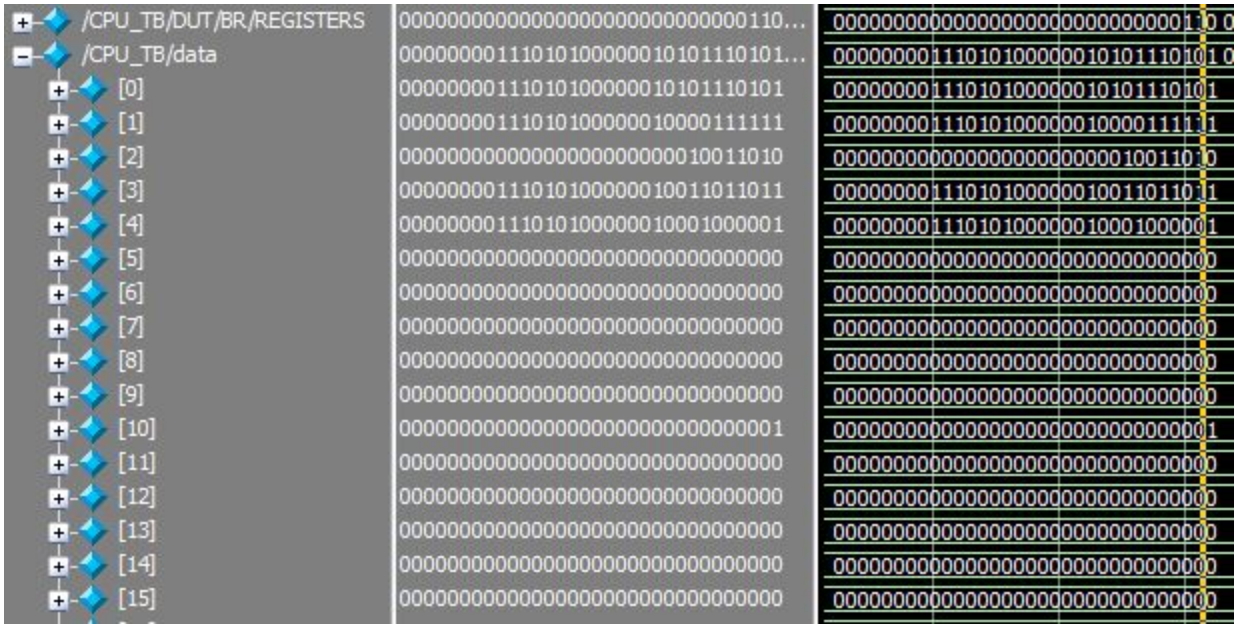
Na nasledujúcej simulácii môžeme vidieť zapísané hodnoty v pamäti registrov, ktoré sa zapísali počas behu CPU_TB. Zápisy sú výsledkom inštrukcii aritmeticko-logických operácii (ADD,SUB,AND,OR,XOR) a zápisu konštánt (LD) do pamäte.



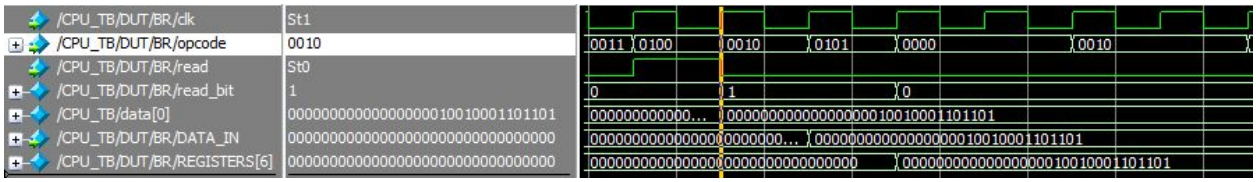
Oneskorené privedenie destinačného registra z dekodera do banky registra cez moduly delay_reg a delay2_reg. Vďaka tomu máme zabezpečené korektné ukladanie aritmeticko_logických operácii na správe miesto v pamäti registrov.



Zápis hodnôt z banky registra do dátovej pamäte podľa CPU_TB:



Čítanie z dátovej pamäte. Opcode čítania predtavuje hodnotu 0010, následne sa registre odošlú do banky registrov pošle aktivácia inštrukcie RD (read=1) a čaká sa na register z dátovej pamäte. Proces môžeme vidieť na obrázku nižšie:



4. Výsledky

Quartus-prime výsledky:

	Task	Time
✔	Compile Design	00:05:21
✔	Analysis & Synthesis	00:00:33
✔	Fitter (Place & Route)	00:03:57
✔	Assembler (Generate programming files)	00:00:26
✔	Timing Analysis	00:00:25

Flow Status	Successful - Sat Dec 12 17:45:42 2020
Quartus Prime Version	20.1.1 Build 720 11/11/2020 SJ Lite Edition
Revision Name	QuartusCPU
Top-level Entity Name	CPU
Family	Cyclone V
Device	5CGXFC9E6F35I7
Timing Models	Final
Logic utilization (in ALMs)	2,542 / 113,560 (2 %)
Total registers	2387
Total pins	165 / 616 (27 %)

	Fmax	Restricted Fmax	Clock Name
1	117.25 MHz	117.25 MHz	clk

Finálne zhodnotenie projektu:

V rámci projektu sa mi podarilo implementovať nasledujúce inštrukcie:
NOP, JMP, ADD, SUB, AND, OR, XOR, LD, WR a RD.
Inštrukcie SHROL, SHROR, BEQ a BNE som neimplementoval z dôvodu nesprávneho nastavenia time managementu a teda nemal som časové kapacity na dokončenie inštrukcii.
Finálne výsledky projektu hodnotím pozitívne, až na 4 inštrukcie sa mi podarilo pochopiť a implementovať základné operácie CPU hardvarskej architektúry. F_{max} procesora činí hodnotu 117.25 MHz, rýchlosť je dosiahnutá aj vďaka implementácii pipeliningu pri aritmeticko-logických operáciach.