Universitatea Ştefan cel Mare Suceava
Facultatea de Inginerie Electrică şi Ştiinţa Calculatoarelor

IMPLEMENTAREA, TESTAREA ŞI EVALUAREA UNUI UCP CU SISTEM DE OPERARE DE TIMP REAL ÎNGLOBAT

- raport de cercetare nr. 3 -

Coordonator ştiinţific,
prof. univ. dr. ing. Vasile-Gheorghiţă GĂITAN

Doctorand,
ing. MOISUC (CIOBANU) Elena-Eugenia

Suceava
- 2015 -

CUPRINS

ABREVIERI

ALM - Adaptive Logic Modules
ALU - Arithmetic Logic Unit
ASIC - Application Specific Integrated Circuit
BEQ - Branch if Equal
CLB - Configurable Logic Block
CPU - Central Processor Unit
DFF - D-Flip Flop
DIP - Dual Inline Package
EAB - Embedded Array Blocks
EX - Execute
FPGA - Field Programmable Gate Array
GPR - General Purpose Register
HDL - Hardware Description Language
HSE - Hardware Scheduler Engine
ID - Instruction Decode
IF - Instruction Fetch
IR - Instruction Registers
ISA - Instruction Set Architecture
J - Jump
JTAG - Joint Test Action Group
LAB - Logic Array Block
LB - Load Byte
LE - Logic Elements
LUT - LookUp Table
MEM - MEMory
MIPS - Microprocessor without Interlocked Pipeline Stages
MPRA - Multi Pipeline Register Architecture
PC - Program Counter
PIA - Programmable Interconnect Array
PLD - Programmable Logic Device
RAM - Random Access Memory
ROM - Read Only Memory
RTL - Register Transfer Level
RTS - Real-Time System
SB - Store Byte
SRAM - Static Random Access Memory
VHDL - Very high speed integrated circuit Hardware Description Language
WB - Write Back

INTRODUCERE

MPRA a fost implementată pentru acest studiu, în prima fază, pe kit-ul din dotare FPGA – QuartusII, Cyclone III. În acest Raport voi prezenta analizele implementărilor etajelor pipeline ale microcontrolerului cu arhitectură multi pipeline (MPRA) pe acest FPGA. Deoarece resursele nu sunt suficiente pentru a simula etajele pipeline (mai ales EXMEM, care este cel mai lung datorită ALU) multiplicate de 4 ori şi de 8 ori (cum se propune în acest studiu), voi realiza aceste simulări pe un kit Xilinx, Virtex6. Deocamdată am „forţat" implementarea pe un Cyclone V, cu resurse suficiente pentru a putea testa codurile Verilog şi a elimina erorile.

1. ARHITECTURA *MULTI PIPELINE REGISTER* (MPRA)

Arhitectura *Multi Pipeline Register* utilizează numai structura organizatorică a arhitecturii MIPS (*Microprocessor without Interlocked Pipeline Stages*). Arhitectura include un planificator funcţional într-un bloc funcțional al procesorului și oferă posibilitatea de a schimba contextul task-urilor în doar o jumătate de ciclu de ceas, eliminând astfel dezavantajele planificatoarelor software. Performanța MPRA nu constă în puterea de procesare, ci în viteza de comutare a contextelor task-urilor și viteza de execuție a algoritmului de planificare.

Modelul de bază pentru un procesor MIPS32 cu 5 stadii pipeline:

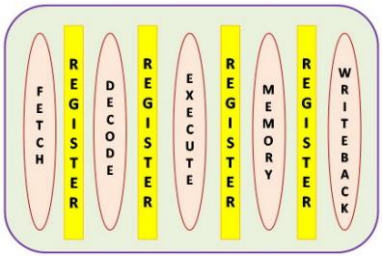
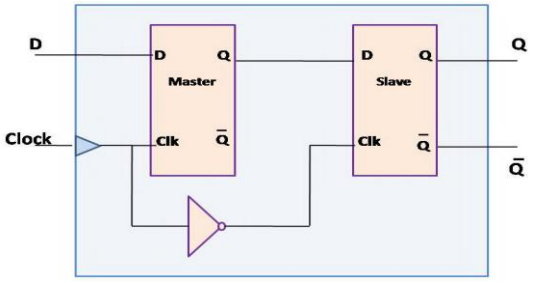


Figura 1-1 MIPS32 cu cinci stadii pipeline

Fig. 1-1 prezintă cele cinci etape și cele patru registre care stochează date între fiecare etapă. Această figură arată mai bine cum pot exista simultan cinci instrucțiuni în interiorul procesorului, fără a fi nevoie să aștepte finalizarea pentru o singură instrucțiune în fiecare etapă. Fiecare registru stochează datele pentru o instrucțiune specifică, eliberând informațiile necesare fiecărei etape, până când a trecut prin întreagul procesor.

Registrele inter-stadii sunt bistabili D master-slave (fig. 1-2). Masterul primeşte noile date de la etapa anterioară a instrucţiunii, în timp ce slave-ul furnizează datele următoarei etape.



Figură 1-2 Registrele pipeline dintre stadii

În cazul procesorului MIPS32 pipeline, la fiecare ciclu de ceas se iniţializează o nouă instrucţiune. Aici, fiecare ciclu de ceas înseamnă unul dintre stadiile de pipeline. Tabelul 1-1 reprezintă structura pipeline tipică, fiecărei instrucţiuni luându-i cinci clock-uri pentru a finaliza execuţia, iar hardware-ul va porni o nouă instrucţiune şi va executa câte o parte în fiecare etapă.

Tabel 1-1 Structura MIPS pipeline

Instruction number	Clock number								
	1	2	3	4	5	6	7	8	9
Instruction *i*	IF	ID	EX	MEM	WB				
Instruction *i+1*		IF	ID	EX	MEM	WB			
Instruction *i+2*			IF	ID	EX	MEM	WB		
Instruction *i+3*				IF	ID	EX	MEM	WB	
Instruction *i+4*					IF	ID	EX	MEM	WB

În MIPS există trei tipuri diferite de instrucţiuni: tipul ***R***, tipul ***I*** şi tipul ***J***:

Tabel 1-2 Formatul instrucţiunilor MIPS

Type	31	format (bits)				0
R	opcode (6)	rs (5)	rs (5)	rs (5)	shamt (5)	funct (6)
I	opcode (6)	rs (5)	rs (5)	immediate (16)		
J	opcode (6)	address (26)				

1. Instrucţiunile de tipul ***R***: prescurtarea pentru tipul ***register***. Acestea utilizează trei regiştri ca operanzi: doi ca sursă şi unul ca destinaţie.
2. Instrucţiunile de tipul ***I***: prescurtarea pentru tipul ***immediate***. Acestea utilizează doi operanzi regiştri şi un operand *16-bit immediate*.
3. Instrucţiunile de tipul ***J***: prescurtarea pentru tipul ***jump***. Se utilizează numai cu instrucţiunea *jump* şi foloseşte un singur operand *26-bit address*.

rs: *the first register source operand*.
rt: *the second register source operand*.
rd: *the register destination operand*, dă rezultatul operaţiunii.
shamt: *shift amount*, este folosit în instrucţiunea *shift* să menţină valoarea deplasată.
funct: *function*, selectează varianta specifică a operaţiunii în câmpul *op*.
imm: *the 16-bit address*, care este utilizat în instrucţiunile de transfer de date.
addr: *the 26-bit address*, care este folosit în instrucţiunile *jump*.

Tabel 1-3 Etapele pipeline din MIPS pentru diferite tipuri de instrucţiuni

***STORE* INSTRUCTION**				
IF	ID	EX	MEM	NOP
***LOAD* INSTRUCTION**				
IF	ID	EX	MEM	WB
***R* TYPE & ARITHMETIC *I* TYPE INSTRUCTION**				
IF	ID	EX	NOP	WB
***BRANCH* INSTRUCTION**				
IF	ID	EX	MEM	NOP
***JUMP* INSTRUCTION**				
IF	ID	NOP	NOP	NOP

Tabel 1-4 Operaţiile pe fiecare stadiu pipeline în arhitectura MIPS (*sursa:* [1])

Stage	Any instruction		
IF	IF/ID.IR ← Mem[PC] ; IF/ID.NPC, PC ← (1f ((EX/MEM.opcode = = branch) & EX/MEM.cond) {EX/MEM. ALUOutput} else {PC+4} ;		
ID	ID/EX.A ← Regs [IF/ID.IR [rs]] ; ID/EX.B ← Regs [IF/ID.IR [rt]] ; ID/EX.NPC ← IF/ID.NPC ; ID/EX.IR ← IF/ID.IR ; ID/EX.Imm ← sign-extend(IF/ID.IR[immediate field]) ;		
	***ALU* instruction**	***Load* or *Store* instruction**	***Branch* instruction**
EX	EX/MEM.IR ← ID/EX.IR ; EX/MEM.ALUOutput ← ID/EX.A func ID/EX.B ; or EX/MEM.ALUOutput ← ID/EX.A *op* ID/EX.Imm ;	EX/MEM.IR to ID/EX.IR ; EX/MEM.ALUOutput ← ID/EX.A + ID/EX.Imm ; EX/MEM.B ← ID/EX.B ;	EX/MEM.ALUOutput ← ID/EX.NPC + (ID/EX.Imm « 2) ; EX/MEM.cond ← (ID/EX.A = = 0) ;
MEM	MEM/WB.IR ← EX/MEM.IR ; MEM/WB.ALUOutput ← EX/MEM.ALUOutput ;	MEM/WB.IR ← EX/MEM.IR ; MEM/WB.LMD ← Mem[EX/MEM.ALIOutput] ; or Mem[EX/MEM.ALIOutput] ← EX/MEM.B ;	
WB	Regs[MEM/WB.IR [rd] ← MEM/WB.ALUOutput ; or Regs[MEM/WB.IR [rt]] ← MEM/WB.ALUOutput ;	For load only: Regs[MEM/WB.IR [rt]] ← MEM/WB.LMD ;	

Cercetarea prezentată în această lucrare se bazează pe o arhitectură funcțională de procesor cu regiştri multipipeline (MPRA), prezentată în**Error! Reference source not found.** [2] şi [2], care prevede un timp foarte redus pentru operaţiunile de comutare de context ca urmare a arhitecturii multipipeline.

MPRA este o arhitectură multipipeline, ceea ce înseamnă că fiecare task are propriul set de registre pipeline.

Tabel 1-5 Instrucțiunile MPRA

Mnemonic	Cod (6 biți)	Sursa 1 (5 biți)	Sursa 2 (5 biți)	Destinație (5 biți)	5biți	6biți	Obs.
Instrucțiuni Aritmetice							
nop	000000	xxxxx	xxxxx	xxxxx	xxxxx	xxxxxx	
add	000001	s1	s2	D	xxxxx	xxxxxx	Tip R
Adună operandul s2 cu s1 și depune rezultatul în registrul d							
sub	000010	s1	s2	D	xxxxx	xxxxxx	Tip R
Scade operandul s2 din s1 și depune rezultatul în registrul d							
mul	000011	s1	s2	D	xxxxx	xxxxxx	Tip R
Înmulțește operandul s2 cu s1 și depune rezultatul în registrul d							
div	000100	s1	s2	D	xxxxx	xxxxxx	Tip R
Împarte operandul s1 la s2 și depune rezultatul în registrul d							
sadd	000101	s1	s2	D	xxxxx	xxxxxx	Tip R
Adună cu semn operandul s2 și s1 și depune rezultatul în registrul d							
ssub	000110	s1	s2	D	xxxxx	xxxxxx	Tip R

Scade cu semn rezultatul s2 din s1 și depune rezultatul în registrul d							
smul	000111	s1	s2	D	xxxxx	xxxxxx	Tip R
Înmulțește cu semn s2 și s1 și depune rezultatul în registrul d							
sdiv	001000	s1	s2	D	xxxxx	xxxxxx	Tip R
Împarte cu semn s1 la s2 și depune rezultatul în d							
addi	001001	s1	d	Operand Imediat			Tip I
Adaugă un operand imediat cu valoarea din s1 și depune rezultatul în registrul d							
subi	001010	s1	d	Operand Imediat			Tip I
Scade un operand imediat din valoarea din s1 și depune rezultatul în registrul d							
muli	001011	s1	d	Operand Imediat			Tip I
Înmulțește un operand imediat cu valoarea din s1 și depune rezultatul în registrul d							
divi	001100	s1	d	Operand Imediat			Tip I
Împarte valoarea din s1 la un operand imediat și depune rezultatul în registrul d							
Instrucțiuni Logice							
and	001101	s1	s2	d	xxxxx	xxxxxx	Tip R
Execută un și logic între valorile regiștrilor s2 și s1 și depune rezultatul în d							
or	001110	s1	s2	d	xxxxx	xxxxxx	Tip R
Execută un sau logic între valorile regiștrilor s2 și s1 și depune rezultatul în d							
xor	001111	s1	s2	d	xxxxx	xxxxxx	Tip R
Execută un sau exclusiv între valorile regiștrilor s2 și s1 și depune rezultatul în d							
nxor	010000	s1	s2	d	xxxxx	xxxxxx	Tip R
Execută un sau exclusiv negat între valorile regiștrilor s2 și s1 și depune rezultatulîn d							
slr	010001	s1	s2	d	xxxxx	xxxxxx	Tip R
Deplasează spre stânga valoarea din s1 cu valoarea din s2 și depune rezultatul în registrul d							
srr	010010	s1	s2	d	xxxxx	xxxxxx	Tip R
Deplasează spre dreapta valoarea din s1 cu valoarea din s2 și depune rezultatul în registrul d							
andi	010011	s1	d	Operand Imediat			Tip I
Execută un și logic între valorile registrului s1 și un imediat și depune rezultatul în d							
ori	010100	s1	d	Operand Imediat			Tip I
Execută un sau logic între valorile registrului s1 și un imediat și depune rezultatul în d							
sli	010101	s1	d	Operand Imediat			Tip I
Execută o deplasare spre stânga a valorii registrului s1 cu valoarea operandului din instrucțiune și depunerezultatul în registrul d							
sri	010110	s1	d	Operand Imediat			Tip I
Execută o deplasare spre dreapta a valorii registrului s1 cu valoarea operandului din instrucțiune și depunerezultatul în registrul d							
Instrucțiuni pentru transferul datelor							
lw	010111	s1	d	Operand Imediat -Offset			Tip I
Încarcă o valoare în registrul d de la adresa specificată în registrul s1 + offset							
sw	011000	s1	d	Operand Imediat -Offset			Tip I
Scrie în memorie la adresa specificată de registrul s1 + offset valoarea din d							
Instrucțiuni de salt condiționat							
beq	011001	s1	s2	Operand Imediat –Offset			Tip I
Se sare dela PC curent la PC+offset dacă s1 = s2							
bne	011010	s1	s2	Operand Imediat –Offset			Tip I
Se sare de la PC curent la PC+offset dacă valorile celor doi regiștri sunt diferite							
bgt	011011	s1	s2	Operand Imediat –Offset			Tip I
Se sare de la PC curent la PC+offset dacă s1 > s2							
Blt	011100	s1	s2	Operand Imediat –Offset			Tip I
Se sare de la PC curent la PC+offset dacă s1 < s2							
bsgt	011101	s1	s2	Operand Imediat –Offset			Tip I

Se sare de la PC curent la PC+offset dacă s1 > s2 (comparație cu semn)							
bslt	011110	S1	S2	Operand Imediat –Offset			Tip I
Se sare de la PC curent la PC+offset dacă s1 < s2 (comparație cu semn)							
Instrucțiuni de salt necondiționat							
j	011111	Operand Imediat -Offset					
Sare la adresa PC + offset							
jr	100000	s1	xxxxx				
Sare la adresa PC + s1							
call	100001	Operand Imediat -Offset					
Sare la adresa PC + offset și salvează adresa de retur și regiștrii speciali în mod automat							
ret	100010	xxxxx					
Restaurează PC la valoarea memorată în urma instrucțiunii call și restaurează valorile anumitor regiștri							
Instrucțiuni speciale							
clra	100011	xxxxx	xxxxx	xxxxx	xxxxx	xxxxxx	
Reseteaza semnalele din ALU (divbyzero, ovfl) mai puțin registrul de ieșire							
signaln	100100	xxn7n6n5	n4n3n2n1n0	xxxxx	xxxxx	xxxxxx	
Instrucțiune ce se inserează la sfârșitul buclei principale a unui task pentru a semnaliza că execuția acestuia s-a terminat							
schedn	100101	c0xn7n6n5	n4n3n2n1n0	xxxxx	xxxxx	xxxxx	
Cere planificatorului lansarea în execuție a unui anumit task. Numărul task-ului ia valori cuprinse între 0 și 255. Execuția acestei instrucțiuni de către planificator poate fi sau nu guvernată de regulile priorităților prestabilite în planificator. *n7:n0 –task-ul ce trebuie lăsat în execuție ia valori între 0 și 255* *C0 - 0 mod forțat* *- 1 respectă regulile de planificare din planificator*							
eni	100110	xxxxx	xxxxx	xxxxx	xxxxx	xxxxxx	
Activează sistemul general de întreruperi							
disi	100111	xxxxx	xxxxx	xxxxx	xxxxx	xxxxxx	
Dezactivează sistemul general de întreruperi							
scheden	101000	xxxxx	xxxxx	xxxxx	xxxxx	xxxxxx	
Activează HSE							
scheddis	101001	xxxxx	xxxxx	xxxxx	xxxxx	xxxxxx	
Dezactivează HSE							
schedenn	101010	xxn7n6n5	n4n3n2n1n0	xxxxx	xxxxx	xxxxxx	
Activează pentru planificare task-ul n							
scheddisn	101011	xxn7n6n5	n4n3n2n1n0	xxxxx	xxxxx	xxxxxx	
Dezactivează pentru planificare task-ul n							
debug	101100	Hxn7n6n5	n4n3n2n1n0				
Scrie valoarea n7:n0 pe portul de debug *H (Halt) - 1 oprește execuția procesorului și activează modul debug. În acest mod se pot citi și scrie toți regiștrii din procesor prin comenzi trimise pe un port serial dedicat;* *- 0 trimite valoarea pe portul de debug fără a opri execuția procesorului*							
UNDEF	101101	xxxxx					
Instrucțiuni nedefinite. Execuția unor astfel de instrucțiuni determină plasarea în mod automat în starea fail-safe							
UNDEF	……	xxxxx					
UNDEF	111110	xxxxx					
nop	111111	xxxxx					

x – Valoare neimportantă, TIP I – Instrucțiune, TIP R – Registru

MPRA a fost implementată, în prima fază, pe FPGA – QuartusII, Cyclone III.

Figură 1-3 Cyclone III de la ALTERA

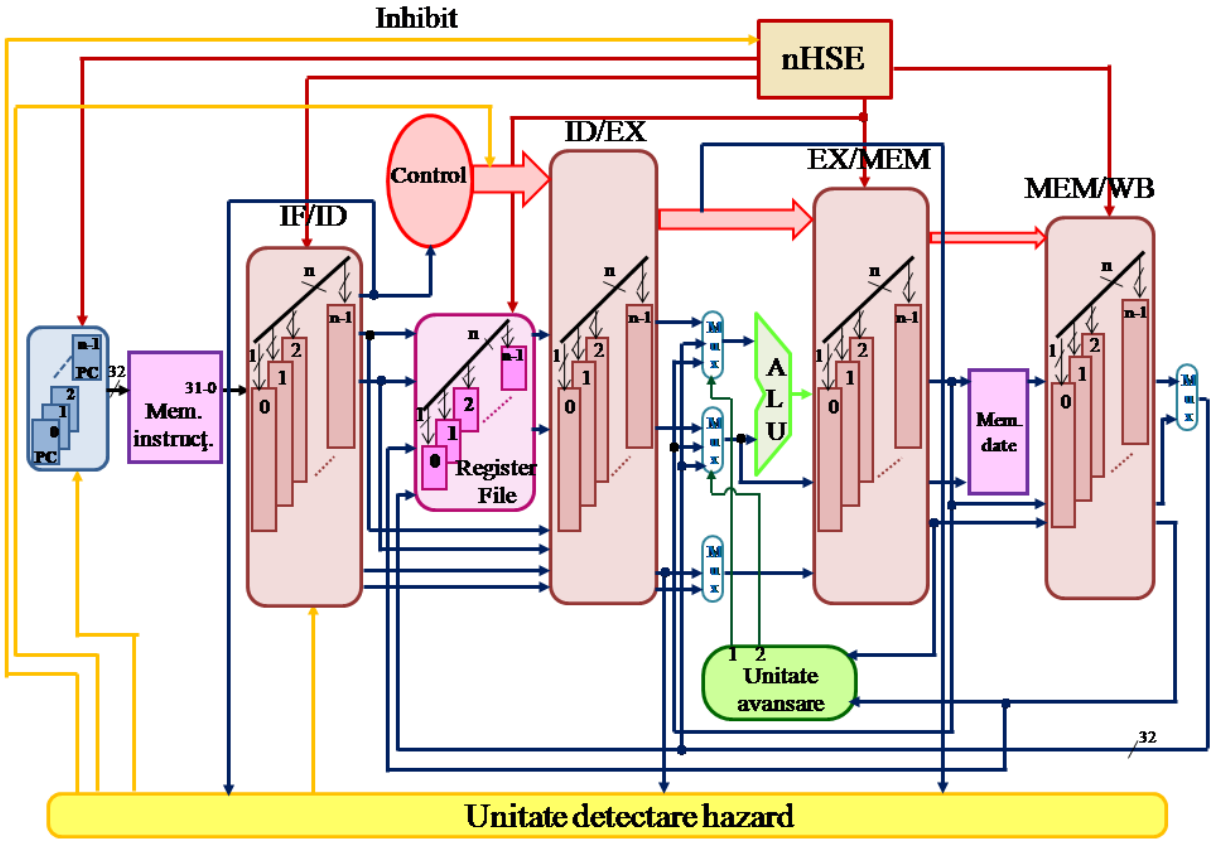
Dispozitivele FPGA (*Field Programmable Gate Array*) oferă elemente integrate cu complexitatea aplicaţiei de circuite integrate orientate (ASIC - *Application Specific Integrated Circuit*), cu avantajul de programabilitate sau, mai bine zis, de configurare [3]. Dispozitivele FPGA permit proiectarea de arhitecturi hardware specializate cu avantajul flexibilităţii mediului programabil ce poate fi implementat [4]. Pe baza acestor dispozitive, suporturile hardware pentru RTOS pot fi ușor puse în aplicare [6], [7]. În prezent, dispozitivele FPGA [7] [8], [9], [10] sunt larg răspândite, sunt mult mai ieftine și au capacități mai mari, echivalentul a milioane de porți logice**Error! Reference source not found.**. Din acest motiv, acest studiu propune un suport implementat hardware.

Multi Pipeline Register Architecture (MPRA), prezentată în [11] și [12], a fost modificată în [13] și transformat în *n*-task *MPRA* (*nMPRA*). Ideea de bază a fost de a replica registrele pipeline și de a crea mai multe instanțe ale procesorului, numite *semi CPU*, pentru a avea un semi-procesor pentru fiecare task *i* ($sCPU_i$).

nMPRA constă dintr-o structură hardware originală utilizată pentru planificarea task-ului, static și dinamic, și oferă managementul unitar al evenimentelor. Scopul a fost de a îmbunatăţi prin hardware performanţele RTOS pentru microcontrolere, pentru a comuta mai rapid între task-uri, pentru a îmbunătăți timpul de răspuns la evenimente externe, pentru a îmbunătăți comportamentul întreruperilor, care sunt tratate ca evenimente în acest caz și pentru a oferi mai multe tipuri de bază de comunicare inter-task-uri (mesaje, mutex-uri etc.).

PC-urile pentru fiecare $sCPU_i$ se vor încărca cu adresa unei celule capcană unde se va transfera controlul, iar de aici se poate sări la software-ul specific fiecărie $sCPU_i$. Fiecare $sCPU_i$ va avea acces atât la o memorie locală cu date și instrucțiuni, cât și la o memorie globală cu date și instrucțiuni.

1.1. *nMPRA*



Figură 1-4 *nMPRA* (*sursa*:Error! Reference source not found.**)**

Arhitectura *nMPRA* este prezentată în Fig. 1-4. MPRA a fost transformată în *nMPRA*, adică a fost multiplicată de *n* ori; pentru fiecare task avem câte un set de regiștrii pipeline (IFID, IDEX, EXMEM, MEMWB), câte un Program Counter (PC) și câte un Banked Register File. Datorită faptului că fiecare task are propriul set de registre pipeline și propriul set de registre generale comutarea de context se poate face într-un ciclu procesor, iar răspunsul la un eveniment extern poate fi întârziat maximum 1,5 cicli procesor. Din aceste motive arhitectura este foarte rapidă. Celelalte resurse sunt folosite în comun de către toate taskurile. O instanță a acestui procesor o vom numi „semi CPU" ($sCPU_i$) pentru task-ul *i* (*i=0, ..., n-1*) şi va executa un singur task ($task_i$).

Toate $sCPU_i$ sunt identice cu excepția $sCPU_0$ care va fi singura activă după reset, singura care va executa instrucțiuni supervizor și singura care va avea acces la *registrele de monitorizare* ale *nMPRA*. $sCPU_0$ are întotdeauna prioritatea 0 și este cea mai prioritară *sCPU* [13] [14].

Arhitectura propusă constituie o metodă eficientă de optimizare a procesorului, având numeroase avantaje: reacția la un eveniment nu depășește 1,5 cicli-procesor dacă evenimentul apărut este atașat unui task mai prioritar decât taskul curent; nu resetează banda de asamblare, nu necesită salvare/restaurare de context, accelerează execuția prin apeluri de subrutine cu

copierea automată a parametrilor și comutarea setului de regiștrii, stivă locală de mare viteză; instrucțini puternice pentru partajarea resurselor, sincronizarea și comunicația între taskuri.

În arhitectura MPRA dimensiunea memoriei consumate pentru implementarea fișierului de regiștri este direct proporțională cu numărul de task-uri.

O implementare a arhitecturii MPRA ce funcționează la 50 MHz este capabilă să realizeze o **comutare a contextelor într-un interval de timp cuprins între 20 și 60ns**. Răspunsul la un eveniment extern este cuprins în același interval.

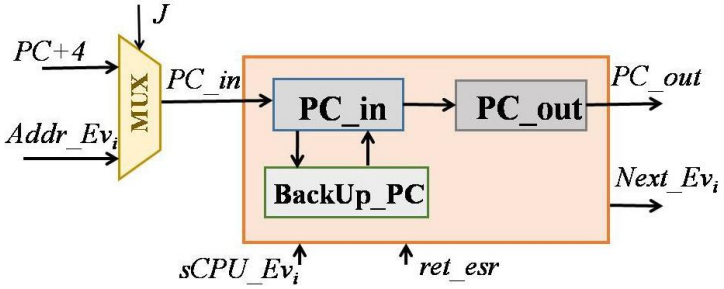
1.2. Etajele şi regiştrii pipeline

MPRA utilizează o linie de asamblare cu 5 etaje pentru îmbunătățirea performanțelor de calcul:

- **IF** – Etapa de citire a instrucțiunii din memorie
- **ID** – Etapa de decodificare a instrucțiunii
- **EX** – Etapa de calcul aritmetic/logic sau a adresei de salt
- **MEM** – Ciclul de acces a memoriei de date (scriere/citire)
- **WB** – Etapa de scriere a rezultatelor în fișierul de regiștri

Acest lucru permite în acest caz rularea a până la 5 instrucțiuni în paralel în mod secvențial pentru ca raportul de execuție să fie în final de 1 instrucțiune/ciclu-procesor. Acesta este un caz ideal. Eficiența benzii de asamblare este însă diminuată de fiecare dată când programul efectuează un salt la o altă adresă. În acest caz toate instrucțiunile care sunt încărcate în banda de asamblare trebuie ignorate pentru a nu schimba logica programului. În etajul IF (Instruction Fetch) adresa din registrul PC (Program Counter) este folosită pentru a adresa memoria program și pentru a citi următoarea instrucțiune ce urmează a fi executată.

Atunci când are loc un eveniment, task-ul asociat devine gata pentru execuție și, dacă are o prioritate mai mare decât task-ul în starea de execuție, atunci registrul *Program Counter* corespunzător $sCPU_i$ (PC_i) (vezi fig. 1-5) este setat cu conținutul registrului-capcană (adresa rutinei asociate pentru evenimentul produs). Pentru a putea implementa redirectarea automată către rutinele de tratare a evenimentelor, PC_i trebuie modificat astfel încât să salveze intern adresa de revenire dintr-o rutină de tratare a unui eveniment [16] [16].



Figură 1-5 Arhitectura *Program Counter* (*sursa*: [16])

În interiorul PC_i vom avea un registru, numit *BackUp_PC*, în care se salvează adresa curentă din PC_i în momentul apariției unui eveniment ce trebuie tratat, semnalizat de semnalul $sCPU\_Ev_i$. În PC_i se va încărca automat, folosind registrul capcană corespunzător, adresa rutinei de tratare a evenimentului activ cel mai prioritar. Revenirea din rutina de tratare a evenimentului este semnalizată prin execuția instrucțiunii *retesr* (*return from the event service routine*), care detemină activarea semnalului *ret_esr* ce semnalizează PC_i să încarce adresa de revenire din registrul *BackUp_PC*, continuând astfel execuția normală a programului. Salvarea adresei de revenire în registrul *BackUp_PC* determină dezactivarea semnalului $NextEv_i$, indicând astfel că se tratează un eveniment și niciun alt eveniment nu mai poate fi

tratat până când nu se termină tratarea evenimentului curent. După revenirea din rutina de tratare a evenimentului curent, semnalul $NextEv_i$ va fi reactivat pentru a semnala că se poate trece la tratarea evenimentului următor [16]. În fig. 1-5 este prezentată structura PC_i.

Regiștrii pipeline pe care arhitectura MPRA îi folosește sunt: **IFID, IDEX, EXMEM, MEMWB**. La acestea patru se adaugă și **PC** care nu este considerat un registru de pipeline dar este gestionat de către HSE în aceeași manieră ca și regiștrii pipeline.

Scopul unui registru pipeline este de a captura date de la o etapă pipeline și de a le furniza următoarei etape pipeline. Acest lucru creează cel puțin un ciclu de ceas întârziere, dar reduce lungimea pentru calea combinatorie a semnalelor, ceea ce permite viteze mai mari de ceas.

La fiecare comutare a contextului, HSE remapează acești 5 regiștri pentru a restaura starea internă a procesorului corespunzătoare acelui task. La fiecare ciclu de ceas informația stocată în regiștrii pipeline trece prin logica etajului, este prelucrată și este depusă în următorul registru de pipeline. Valoarea adresei instrucțiunii memorată în registrul PC este folosită pentru adresarea memoriei de cod. Rezultatul citit din memorie este depus apoi în registrul intermediar IFID.

În registrul IDEX se regăsesc următoarele date: instrucțiunea completă care este copiată din registrul IFID, datele obținute în urma adresării Banked Register File și semnalele de control pentru etajele EX, MEM, WB obținute în urma decodificării instrucțiunii.

În etapa ID are loc decodarea instrucțiunii și adresarea fișierului de regiștri indiferent de tipul instrucțiunii. Informația extrasă din fișierul de regiștri va fi valorificată dacă instrucțiunea este de tipul R sau I, și va fi ignorată dacă este o instrucțiune de salt sau o instrucțiune specială.

În etapa EX de execuție a instrucțiunilor are loc calculul operațiilor aritmetice, logice, evaluarea condițiilor și generarea adreselor de salt pentru instrucțiunile de salt.

Rezultatul operațiilor de adunare este depus în registrul EXMEM pentru o stocare temporară. La proiectarea sistemului trebuie luat în calcul că pentru fiecare task din cele 256 posibile trebuie alocată memorie pentru toți regiștrii pipeline. Semnalele destinate etajelor MEM și WB sunt memorate în registrul EXMEM având aceeași consistență ca în registrul IDEX.

În etajul pipeline MEM are loc accesul memoriei de date pentru scriere sau citire. Validarea operației de scriere sau citire este făcută de semnalele *memread* și *memwrite* ce provin de la unitatea de control. Datele citite din memorie sunt depuse în registrul MEMWB pentru stocare temporară (în eventualitatea că HSE schimbă contextele) și utilizare ulterioară. Multiplexorul este folosit pentru selecția datei care este scrisă în fișierul de regiștri. În aceste condiții, multiplexorul va selecta câmpul *ALUout* dacă instrucțiunea a fost o operație matematică sau logică sau *RDData* dacă operația a fost una de citire a memoriei. Adresarea memoriei se face în permanență cu rezultatul din unitatea ALU. Aceasta calculează adresa ca fiind rezultatul citit din regiștrii de lucru la care se adaugă un deplasament aliniat în prelabil pe 32 biți. Valoarea selectată de multiplexor este folosită în etapa WB pentru adresarea și stocarea datelor în Register File.

Blocurile de memorie sunt de fapt unul și același. Accesul concurent pentru scrierea și citirea datelor și instrucțiunilor în același ciclu de ceas în etajele IF și MEM nu ridică probleme deoarece memoria folosită în această implementare este o memorie multiport ce permite accese multiple. În implementarea actuală, memoria poate suporta două citiri simultane sau o citire și o scriere. Deși memoria este una mixtă și conține atât date cât și cod se realizează totuși o separare a celor două.

În arhitectura MPRA, ALU efectuează următoarele operații:
- operații matematice cu și fără semn;
- operații logice (incluzând operații decizionale pentru instrucțiunile de salt condiționat);
- calculează deplasamentul în cazul instrucțiunilor de tip lw sau sw.

ALU nu calculează adresa de salt pentru instrucțiunile de salt condiționat sau necondiționat. Pentru aceste operații MPRA utilizează două sumatoare simple dedicate. În tabelul 1-6 se prezintă codul generat de unitatea de control și operația executată.

Tabel 1-6 Codul operațiilor ALU generate de unitatea de control

Cod	Operație
00001	*Adunare*
00010	*Scădere*
00011	*Înmulțire*
00100	*Împărțire*
00101	*Complementare*
00110	*AND*
00111	*OR*
01000	*XOR*
01001	*NXOR*
01010	*SHL*
01011	*SHR*
01100	*sadd (adunare cu semn)*
01101	*ssub (scădere cu semn)*
01110	*smul (înmulțire cu semn)*
01111	*sdiv (împărțire cu semn)*
00000	*Păstrează ieșirile ALU din operația precedentă*
10000	*Setează valorile implicite –toate ieșirile unitățiiALU sunt setate pe valoarea 0*
--------	*Ieșiri implicite*
11111	*Ieșiri implicite*

Rezultatul pe 32 biți al operației aritmetice sau logice este depus în câmpul ALUout din registrul pipeline EXMEM de unde va fi ulterior preluat și depus fie în fișierul de regiștri dacă operația a fost una matematică, fie va fi folosit pentru adresarea memoriei de date dacă instrucțiunea a fost de tipul *lw* sau *sw*. Rezultatele operațiilor decizionale sunt evaluate în etajul MEM în paralel cu semnalul *branch* generat de unitatea de control. Operanzii unității ALU pot proveni fie din registrul IDEX, fie de la ieșirea din memoria de date sau chiar de la ieșirea ALU după avansarea în prealabil a datelor de către unitățile de forwarding.

În arhitectura MPRA rezultatul împărțirii la 0 este 0 și este semnalizat întotdeauna cu excepție de către ALU. Excepția poate fi sau nu tratată.

1.3. Banked Register File

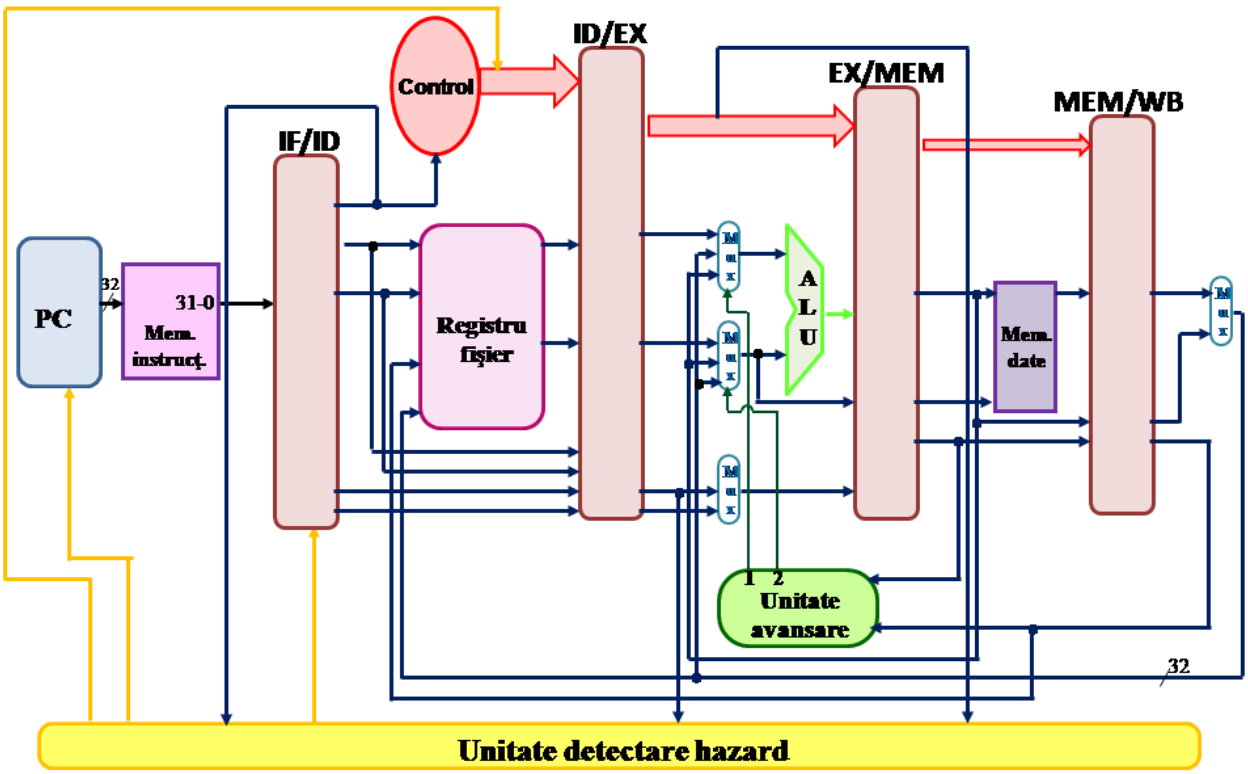
Register File a fost special proiectat pentru a suporta comutarea rapidă a contextelor task-urilor. În arhitecturile clasice salvarea și restaurarea contextelor era materializată în ciclii de acces la memoria externă. Durata schimbului contextelor era dependentă de numărul regiștrilor salvați, dimensiunea lor și lățimea magistralei de date ce interconectează procesorul și memoria RAM. MPRA nu utilizează conceptul de stivă așa cum este el folosit în arhitecturile existente, dar o parte din această funcționalitate este inclusă în Register File doar pentru menținearea unei ordini a apelului de funcții. Fișierul de regiștri implementează 32 de regiștri pe 32 biți pentru fiecare context al unei funcții. Fiecare task suportă o înlănțuire a

apelurilor funcțiilor până la 10 nivele. Contextul unei funcții poartă numele de set de regiștri, iar totalitatea regiștrilor unui task compun un bank de regiștri.

Fiecare $sCPU_i$ (fiecare task i) are un Register File organizat pe bancuri, care au fost prevăzute pentru a accelera procesul de apelare și revenire din procedură. La un apel de procedură se va folosi bancul imediat superior, iar la revenirea din procedură se va folosi bancul imediat inferior. Un banc de regiștrii conține 32 de regiștri organizați ca 18 regiștrii de uz general, 4 regiștrii speciali și 10 regiștrii pentru parametrii**Error! Reference source not found.**. Fișierul de regiștrii și memoria de program și date sunt ambele implementate în tehnologie multi port ceea ce permite accese multiple în același ciclu de ceas. Fișierul de regiștrii permite două operații de citire și una de scriere, în timp ce memoria de program și date permite doar două citiri simultane sau o citire și o scriere. Comutarea contextelor task-urilor presupune atât comutarea regiștrilor din fișierul de regiștri, cât și cea a fișierelor pipeline, ambele acțiuni fiind dirijate de unitatea HSE.

2. IMPLEMENTAREA ETAJELOR PIPELINE PE CYCLONE III

2.1. Structura organizațională MIPS32 utilizată pentru MPRA



Figură 2-1 Arhitectura *MIPS32*

Pentru implementare și analiză s-a utilizat codul verilog pentru procesorul MIPS32r1 latest [17].

	Source Clock(s)	Destination Clock(s)	Delay Added in ns
1	clock	clock	19.3

PowerPlay Power Analyzer Status	Successful - Sat Sep 26 23:08:23 2015
Quartus II 64-Bit Version	15.0.0 Build 145 04/22/2015 SJ Web Edition
Revision Name	Processor
Top-level Entity Name	Processor
Family	Cyclone V
Device	5CGXFC9E6F31C7
Power Models	Final
Total Thermal Power Dissipation	530.98 mW
Core Dynamic Thermal Power Dissipation	0.00 mW
Core Static Thermal Power Dissipation	518.51 mW
I/O Thermal Power Dissipation	12.47 mW
Power Estimation Confidence	Low: user provided insufficient toggle rate data

Processor-Flow Summary

```
Flow Summary report for Processor
Tue Sep 22 23:19:13 2015
Quartus II 64-Bit Version 15.0.0 Build 145 04/22/2015 SJ Web Edition

+---------------------------------------------------------------------------------+
; Flow Summary                                                                    ;
+-----------------------------------+---------------------------------------------+
; Flow Status                       ; Successful - Wed Sep 16 20:56:55 2015       ;
; Quartus II 64-Bit Version         ; 15.0.0 Build 145 04/22/2015 SJ Web Edition ;
; Revision Name                     ; Processor                                   ;
; Top-level Entity Name             ; Processor                                   ;
; Family                            ; Cyclone V                                   ;
; Device                            ; 5CGXFC9E6F31C7                              ;
; Timing Models                     ; Final                                       ;
; Logic utilization (in ALMs)       ; 2,957 / 113,560 ( 3 % )                     ;
; Total registers                   ; 1858                                        ;
; Total pins                        ; 180 / 536 ( 34 % )                          ;
; Total virtual pins                ; 0                                           ;
; Total block memory bits           ; 0 / 12,492,800 ( 0 % )                      ;
; Total DSP Blocks                  ; 6 / 342 ( 2 % )                             ;
; Total HSSI RX PCSs                ; 0 / 12 ( 0 % )                              ;
; Total HSSI PMA RX Deserializers   ; 0 / 12 ( 0 % )                              ;
; Total HSSI TX PCSs                ; 0 / 12 ( 0 % )                              ;
; Total HSSI PMA TX Serializers     ; 0 / 12 ( 0 % )                              ;
; Total PLLs                        ; 0 / 20 ( 0 % )                              ;
; Total DLLs                        ; 0 / 4 ( 0 % )                               ;
+-----------------------------------+---------------------------------------------+
```

Processor-Resource Usage Summary

Resource Usage Summary report for Processor
Tue Sep 22 23:20:53 2015
Quartus II 64-Bit Version 15.0.0 Build 145 04/22/2015 SJ Web Edition

; Fitter Resource Usage Summary

Resource	Usage	%
Logic utilization (ALMs needed / total ALMs on device)	2,957 / 113,560	3 %
ALMs needed [=A-B+C]	2,957	
[A] ALMs used in final placement [=a+b+c+d]	3,203 / 113,560	3 %
[a] ALMs used for LUT logic and registers	282	
[b] ALMs used for LUT logic	2,280	
[c] ALMs used for registers	641	
[d] ALMs used for memory (up to half of total ALMs)	0	
[B] Estimate of ALMs recoverable by dense packing	307 / 113,560	< 1 %
[C] Estimate of ALMs unavailable [=a+b+c+d]	61 / 113,560	< 1 %
[a] Due to location constrained logic	0	
[b] Due to LAB-wide signal conflicts	4	
[c] Due to LAB input limits	57	
[d] Due to virtual I/Os	0	
Difficulty packing design	Low	
Total LABs: partially or completely used	391 / 11,356	3 %
-- Logic LABs	391	
-- Memory LABs (up to half of total LABs)	0	
Combinational ALUT usage for logic	3,778	
-- 7 input functions	108	
-- 6 input functions	1,881	
-- 5 input functions	634	
-- 4 input functions	247	
-- <=3 input functions	908	
Combinational ALUT usage for route-throughs	250	
Dedicated logic registers	1,858	
-- By type:		
-- Primary logic registers	1,846 / 227,120	< 1 %
-- Secondary logic registers	12 / 227,120	< 1 %
-- By function:		
-- Design implementation registers	1,858	
-- Routing optimization registers	0	
Virtual pins	0	
I/O pins	180 / 536	34 %
-- Clock pins	7 / 16	44 %
-- Dedicated input pins	0 / 35	0 %
Global signals	1	
M10K blocks	0 / 1,220	0 %
Total MLAB memory bits	0	
Total block memory bits	0 / 12,492,800	0 %
Total block memory implementation bits	0 / 12,492,800	0 %
Total DSP Blocks	6 / 342	2 %
Fractional PLLs	0 / 8	0 %
Global clocks	1 / 16	6 %
Quadrant clocks	0 / 88	0 %
Horizontal periphery clocks	0 / 24	0 %
SERDES Transmitters	0 / 140	0 %
SERDES Receivers	0 / 140	0 %
JTAGs	0 / 1	0 %
ASMI blocks	0 / 1	0 %
CRC blocks	0 / 1	0 %
Remote update blocks	0 / 1	0 %
Oscillator blocks	0 / 1	0 %
Hard IPs	0 / 2	0 %
Standard RX PCSs	0 / 12	0 %
HSSI PMA RX Deserializers	0 / 12	0 %
Standard TX PCSs	0 / 12	0 %
HSSI PMA TX Serializers	0 / 12	0 %
Channel PLLs	0 / 12	0 %
Impedance control blocks	0 / 3	0 %
Hard Memory Controllers	0 / 2	0 %
Average interconnect usage (total/H/V)	1.4% / 1.4% / 1.5%	
Peak interconnect usage (total/H/V)	33.4% / 34.9% / 28.8%	
Maximum fan-out	1858	
Highest non-global fan-out	1798	
Total fan-out	27305	
Average fan-out	4.37	

2.1.1. IFID

IFID_Stage-Flow Summary

```
Flow Summary report for IFID_Stage
Tue Sep 22 23:06:03 2015
Quartus II 64-Bit Version 15.0.0 Build 145 04/22/2015 SJ Web Edition
```

Flow Summary	
Flow Status	Successful - Mon Sep 21 21:45:44 2015
Quartus II 64-Bit Version	15.0.0 Build 145 04/22/2015 SJ Web Edition
Revision Name	IFID_Stage
Top-level Entity Name	IFID_Stage
Family	Cyclone V
Device	5CEFA9F31C7
Timing Models	Final
Logic utilization (in ALMs)	26 / 113,560 (< 1 %)
Total registers	98
Total pins	200 / 480 (42 %)
Total virtual pins	0
Total block memory bits	0 / 12,492,800 (0 %)
Total DSP Blocks	0 / 342 (0 %)
Total HSSI RX PCSs	0
Total HSSI PMA RX Deserializers	0
Total HSSI TX PCSs	0
Total HSSI PMA TX Serializers	0
Total PLLs	0 / 8 (0 %)
Total DLLs	0 / 4 (0 %)

IFID_Stage-General Register Statistics

```
General Register Statistics report for IFID_Stage
Tue Sep 22 23:06:36 2015
Quartus II 64-Bit Version 15.0.0 Build 145 04/22/2015 SJ Web Edition
```

General Register Statistics	
Statistic	Value
Total registers	98
Number of registers using Synchronous Clear	98
Number of registers using Synchronous Load	0
Number of registers using Asynchronous Clear	0
Number of registers using Asynchronous Load	0
Number of registers using Clock Enable	98
Number of registers using Preset	0

```
Multiplexer Restructuring Statistics (Restructuring Performed) report for IFID_Stage
Tue Sep 22 23:06:44 2015
Quartus II 64-Bit Version 15.0.0 Build 145 04/22/2015 SJ Web Edition
```

Multiplexer Restructuring Statistics (Restructuring Performed)

Multiplexer Inputs	Bus Width	Baseline Area	Area if Restructured	Saving if Restructured	Registered	Example Multiplexer Output
3:1	34 bits	68 LEs	0 LEs	68 LEs	Yes	|IFID_Stage|ID_PCAdd4[2]~reg0
3:1	32 bits	64 LEs	0 LEs	64 LEs	Yes	|IFID_Stage|ID_RestartPC[17]~reg0
4:1	32 bits	64 LEs	0 LEs	64 LEs	Yes	|IFID_Stage|ID_Instruction[0]~reg0

IFID_Stage-Post-Synthesis Netlist Statistics for Top Partition
Post-Synthesis Netlist Statistics for Top Partition report for IFID_Stage
Tue Sep 22 23:06:51 2015
Quartus II 64-Bit Version 15.0.0 Build 145 04/22/2015 SJ Web Edition

Post-Synthesis Netlist Statistics for Top Partition	
Type	Count
arriav_ff	98
ENA SCLR	98
arriav_lcell_comb	3
normal	3
2 data inputs	1
3 data inputs	2
boundary_port	200
Max LUT depth	1.00
Average LUT depth	0.04

IFID_Stage-Resource Usage Summary
Resource Usage Summary report for IFID_Stage
Tue Sep 22 23:07:24 2015
Quartus II 64-Bit Version 15.0.0 Build 145 04/22/2015 SJ Web Edition

Fitter Resource Usage Summary		
Resource	Usage	%
Logic utilization (ALMs needed / total ALMs on device)	26 / 113,560	< 1 %
ALMs needed [=A-B+C]	26	
[A] ALMs used in final placement [=a+b+c+d]	51 / 113,560	< 1 %
[a] ALMs used for LUT logic and registers	0	
[b] ALMs used for LUT logic	2	
[c] ALMs used for registers	49	
[d] ALMs used for memory (up to half of total ALMs)	0	
[B] Estimate of ALMs recoverable by dense packing	25 / 113,560	< 1 %
[C] Estimate of ALMs unavailable [=a+b+c+d]	0 / 113,560	0 %
[a] Due to location constrained logic	0	
[b] Due to LAB-wide signal conflicts	0	
[c] Due to LAB input limits	0	
[d] Due to virtual I/Os	0	
Difficulty packing design	Low	
Total LABs: partially or completely used	15 / 11,356	< 1 %
-- Logic LABs	15	
-- Memory LABs (up to half of total LABs)	0	
Combinational ALUT usage for logic	4	
-- 7 input functions	0	
-- 6 input functions	0	
-- 5 input functions	0	
-- 4 input functions	0	
-- <=3 input functions	4	
Combinational ALUT usage for route-throughs	68	
Dedicated logic registers	98	
-- By type:		
-- Primary logic registers	98 / 227,120	< 1 %
-- Secondary logic registers	0 / 227,120	0 %
-- By function:		
-- Design implementation registers	98	
-- Routing optimization registers	0	

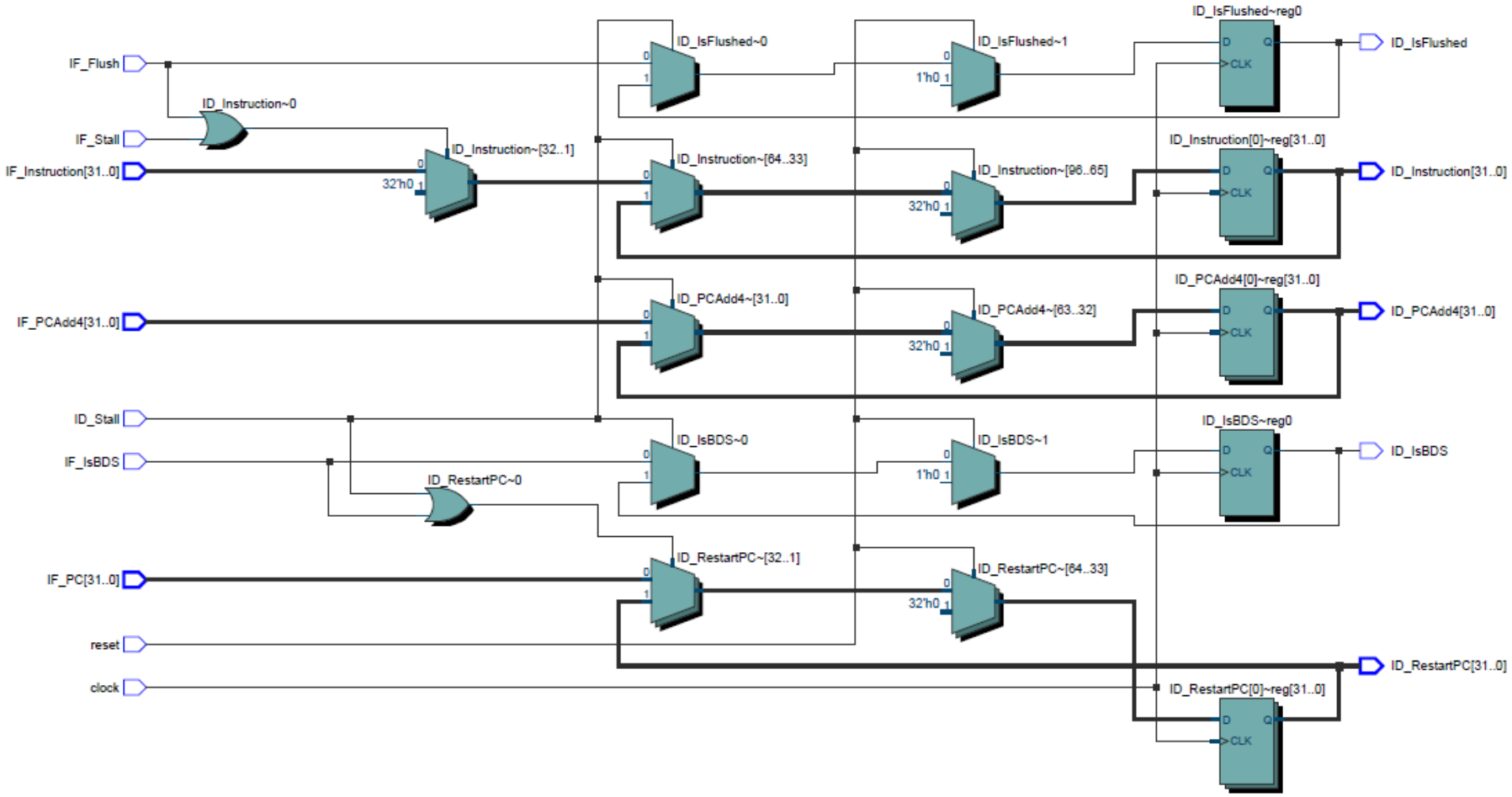
Virtual pins	0	
I/O pins	200 / 480	42 %
-- Clock pins	4 / 12	33 %
-- Dedicated input pins	0 / 11	0 %
Global signals	1	
M10K blocks	0 / 1,220	0 %
Total MLAB memory bits	0	
Total block memory bits	0 / 12,492,800	0 %
Total block memory implementation bits	0 / 12,492,800	0 %
Total DSP Blocks	0 / 342	0 %
Fractional PLLs	0 / 8	0 %
Global clocks	1 / 16	6 %
Quadrant clocks	0 / 88	0 %
Horizontal periphery clocks	0 / 24	0 %
SERDES Transmitters	0 / 140	0 %
SERDES Receivers	0 / 140	0 %
JTAGs	0 / 1	0 %
ASMI blocks	0 / 1	0 %
CRC blocks	0 / 1	0 %
Remote update blocks	0 / 1	0 %
Oscillator blocks	0 / 1	0 %
Impedance control blocks	0 / 3	0 %
Hard Memory Controllers	0 / 2	0 %
Average interconnect usage (total/H/V)	0.1% / 0.1% / 0.1%	
Peak interconnect usage (total/H/V)	0.7% / 0.6% / 1.3%	
Maximum fan-out	98	
Highest non-global fan-out	69	
Total fan-out	767	
Average fan-out	1.34	

IFID_Stage-Routing Usage Summary

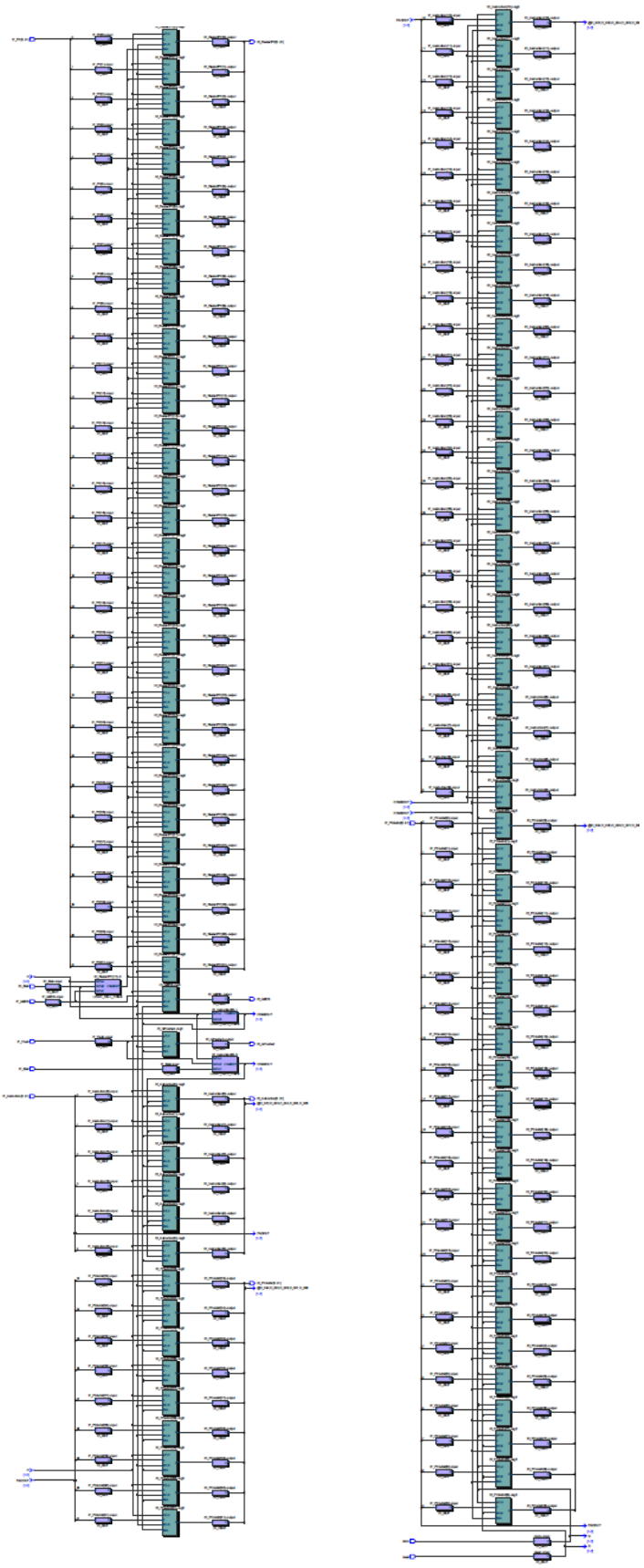
Routing Usage Summary report for IFID_Stage
Tue Sep 22 23:08:25 2015
Quartus II 64-Bit Version 15.0.0 Build 145 04/22/2015 SJ Web Edition

Routing Usage Summary

Routing Resource Type	Usage
Block interconnects	226 / 721,028 (< 1 %)
C12 interconnects	44 / 30,780 (< 1 %)
C2 interconnects	153 / 303,248 (< 1 %)
C4 interconnects	146 / 140,620 (< 1 %)
DQS bus muxes	0 / 35 (0 %)
DQS-18 I/O buses	0 / 35 (0 %)
DQS-9 I/O buses	0 / 35 (0 %)
Direct links	0 / 721,028 (0 %)
Global clocks	1 / 16 (6 %)
Horizontal periphery clocks	0 / 96 (0 %)
Local interconnects	1 / 227,120 (< 1 %)
Quadrant clocks	0 / 88 (0 %)
R14 interconnects	119 / 30,168 (< 1 %)
R14/C12 interconnect drivers	148 / 53,952 (< 1 %)
R3 interconnects	164 / 331,920 (< 1 %)
R6 interconnects	176 / 622,512 (< 1 %)
Spine clocks	4 / 480 (< 1 %)
Wire stub REs	0 / 40,996 (0 %)



Figură 2-2 RTL IFID



Figură 2-3 Tehnology map IFID

2.1.2. IDEX

IDEX_Stage-Routing Usage Summary

Routing Usage Summary report for IDEX_Stage
Tue Sep 22 23:02:00 2015
Quartus II 64-Bit Version 15.0.0 Build 145 04/22/2015 SJ Web Edition

Routing Usage Summary	
Routing Resource Type	Usage
Block interconnects	424 / 721,028 (< 1 %)
C12 interconnects	109 / 30,780 (< 1 %)
C2 interconnects	267 / 303,248 (< 1 %)
C4 interconnects	381 / 140,620 (< 1 %)
DQS bus muxes	0 / 35 (0 %)
DQS-18 I/O buses	0 / 35 (0 %)
DQS-9 I/O buses	0 / 35 (0 %)
Direct links	18 / 721,028 (< 1 %)
Global clocks	1 / 16 (6 %)
Horizontal periphery clocks	0 / 96 (0 %)
Local interconnects	2 / 227,120 (< 1 %)
Quadrant clocks	0 / 88 (0 %)
R14 interconnects	149 / 30,168 (< 1 %)
R14/C12 interconnect drivers	219 / 53,952 (< 1 %)
R3 interconnects	219 / 331,920 (< 1 %)
R6 interconnects	224 / 622,512 (< 1 %)
Spine clocks	6 / 480 (1 %)
Wire stub REs	0 / 40,996 (0 %)

Multiplexer Restructuring Statistics (Restructuring Performed) report for IDEX_Stage
Tue Sep 22 23:00:31 2015
Quartus II 64-Bit Version 15.0.0 Build 145 04/22/2015 SJ Web Edition

Multiplexer Restructuring Statistics (Restructuring Performed)						
Multiplexer Inputs	Bus Width	Baseline Area	Area if Restructured	Saving if Restructured	Registered	Example Multiplexer Output
3:1	139 bits	278 LEs	0 LEs	278 LEs	Yes	|IDEX_Stage|EX_RestartPC[1]~reg0
4:1	15 bits	30 LEs	0 LEs	30 LEs	Yes	|IDEX_Stage|EX_ALUOp[0]~reg0

IDEX_Stage-Post-Synthesis Netlist Statistics for Top Partition

Post-Synthesis Netlist Statistics for Top Partition report for IDEX_Stage
Tue Sep 22 23:00:38 2015
Quartus II 64-Bit Version 15.0.0 Build 145 04/22/2015 SJ Web Edition

Post-Synthesis Netlist Statistics for Top Partition	
Type	Count
arriav_ff	154
ENA SCLR	154
arriav_lcell_comb	3
normal	3
2 data inputs	2
3 data inputs	1
boundary_port	339
Max LUT depth	1.00
Average LUT depth	0.03

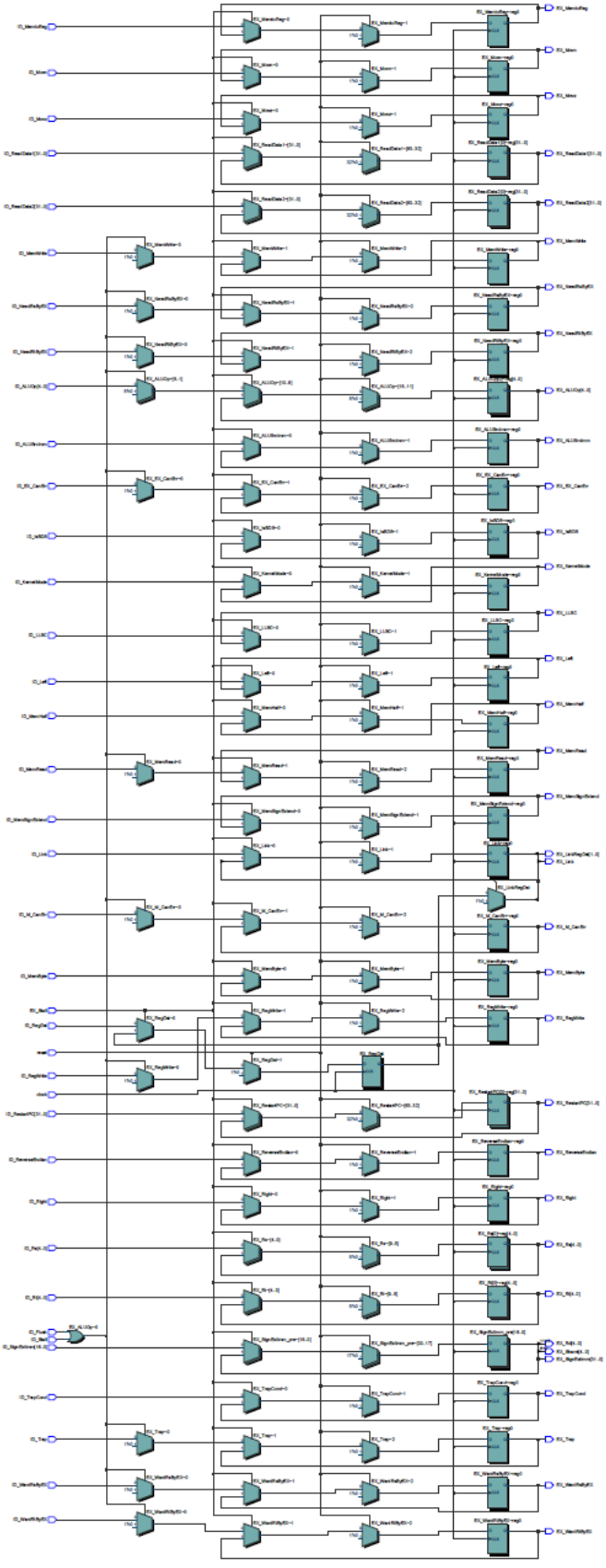
IDEX_Stage-Partition Statistics

Partition Statistics report for IDEX_Stage
Tue Sep 22 23:01:21 2015
Quartus II 64-Bit Version 15.0.0 Build 145 04/22/2015 SJ Web Edition

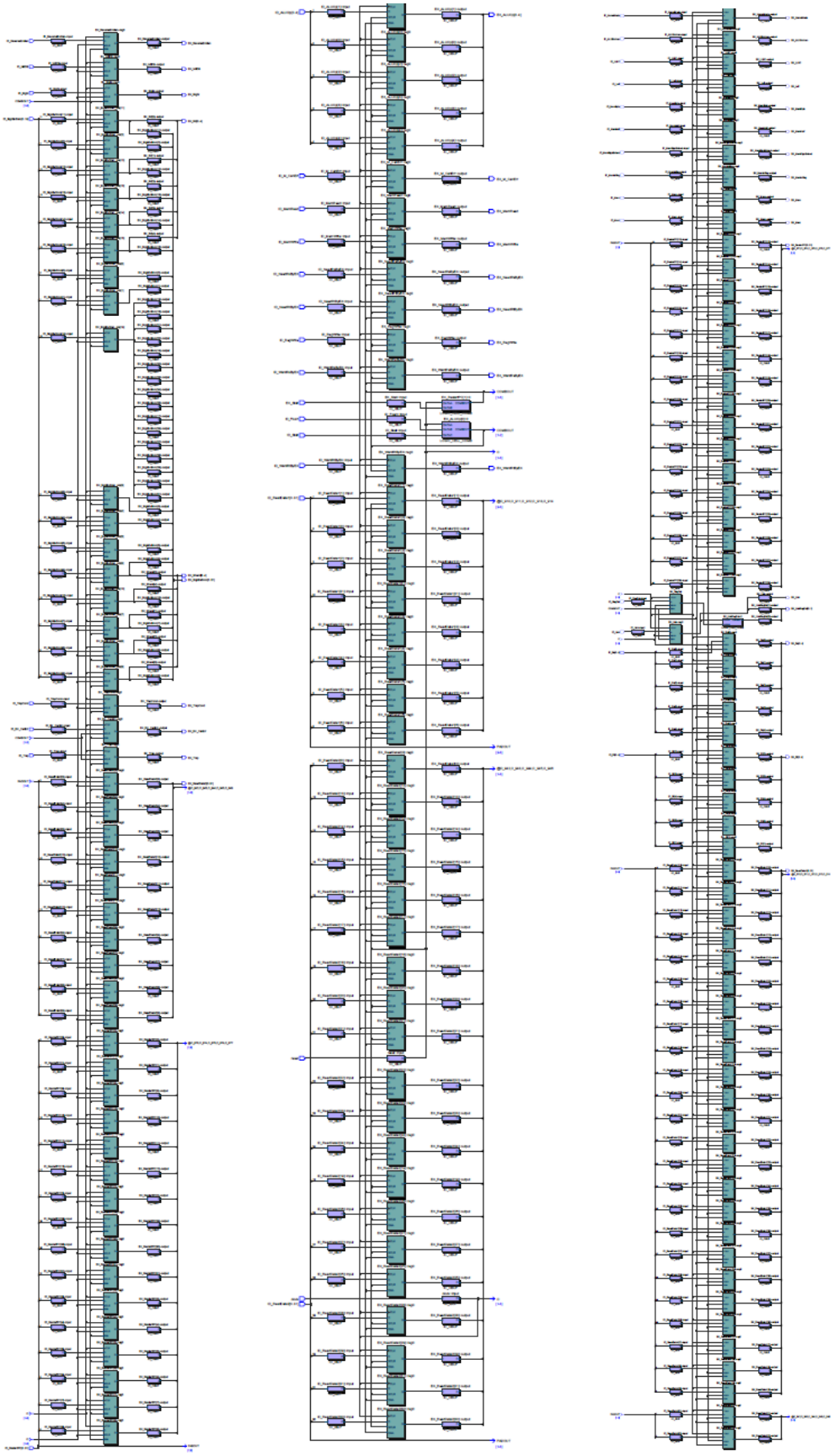
; Fitter Partition Statistics

Statistic	Top	hard_block:auto_generated_inst
Logic utilization (ALMs needed / total ALMs on device)	40 / 113560 (< 1 %)	0 / 113560 (0 %)
ALMs needed [=A-B+C]	40	0
[A] ALMs used in final placement [=a+b+c+d]	80 / 113560 (< 1 %)	0 / 113560 (0 %)
[a] ALMs used for LUT logic and registers	1	0
[b] ALMs used for LUT logic	2	0
[c] ALMs used for registers	77	0
[d] ALMs used for memory (up to half of total ALMs)	0	0
[B] Estimate of ALMs recoverable by dense packing	40 / 113560 (< 1 %)	0 / 113560 (0 %)
[C] Estimate of ALMs unavailable [=a+b+c+d]	0 / 113560 (0 %)	0 / 113560 (0 %)
[a] Due to location constrained logic	0	0
[b] Due to LAB-wide signal conflicts	0	0
[c] Due to LAB input limits	0	0
[d] Due to virtual I/Os	0	0
Difficulty packing design	Low	Low
Total LABs: partially or completely used	46 / 11356 (< 1 %)	0 / 11356 (0 %)
-- Logic LABs	46	0
-- Memory LABs (up to half of total LABs)	0	0
Combinational ALUT usage for logic	4	0
-- 7 input functions	0	0
-- 6 input functions	0	0
-- 5 input functions	0	0
-- 4 input functions	0	0
-- <=3 input functions	4	0
Combinational ALUT usage for route-throughs	101	0
Memory ALUT usage	0	0
-- 64-address deep	0	0
-- 32-address deep	0	0
Dedicated logic registers	0	0
-- By type:		
-- Primary logic registers	154 / 227120 (< 1 %)	0 / 227120 (0 %)

-- Secondary logic registers	0 / 227120 (0 %)	0 / 227120 (0 %)
-- By function:		
-- Design implementation registers	154	0
-- Routing optimization registers	0	0
Virtual pins	0	0
I/O pins	339	0
I/O registers	0	0
Total block memory bits	0	0
Total block memory implementation bits	0	0
Clock enable block	1 / 128 (< 1 %)	0 / 128 (0 %)
Connections		
-- Input Connections	0	0
-- Registered Input Connections	0	0
-- Output Connections	0	0
-- Registered Output Connections	0	0
Internal Connections		
-- Total Connections	1244	0
-- Registered Connections	181	0
External Connections		
-- Top	0	0
-- hard_block:auto_generated_inst	0	0
Partition Interface		
-- Input Ports	159	0
-- Output Ports	180	0
-- Bidir Ports	0	0
Registered Ports		
-- Registered Input Ports	0	0
-- Registered Output Ports	0	0
Port Connectivity		
-- Input Ports driven by GND	0	0
-- Output Ports driven by GND	0	0
-- Input Ports driven by VCC	0	0
-- Output Ports driven by VCC	0	0
-- Input Ports with no Source	0	0
-- Output Ports with no Source	0	0
-- Input Ports with no Fanout	0	0



Figură 2-4 RTL IDEX



Figură 2-5 Tehnology map IDEX

2.1.3. EXMEM

EXMEM_Stage-Flow Summary

Flow Summary	
Flow Status	Successful - Mon Sep 21 21:09:04 2015
Quartus II 64-Bit Version	15.0.0 Build 145 04/22/2015 SJ Web Edition
Revision Name	EXMEM_Stage
Top-level Entity Name	EXMEM_Stage
Family	Cyclone V
Device	5CGXBC9E7F31C8
Timing Models	Final
Logic utilization (in ALMs)	32 / 113,560 (< 1 %)
Total registers	117
Total pins	242 / 536 (45 %)
Total virtual pins	0
Total block memory bits	0 / 12,492,800 (0 %)
Total DSP Blocks	0 / 342 (0 %)
Total HSSI RX PCSs	0 / 12 (0 %)
Total HSSI PMA RX Deserializers	0 / 12 (0 %)
Total HSSI TX PCSs	0 / 12 (0 %)
Total HSSI PMA TX Serializers	0 / 12 (0 %)
Total PLLs	0 / 20 (0 %)
Total DLLs	0 / 4 (0 %)

Multiplexer Restructuring Statistics (Restructuring Performed) report for EXMEM_Stage
Tue Sep 22 22:53:18 2015
Quartus II 64-Bit Version 15.0.0 Build 145 04/22/2015 SJ Web Edition

Multiplexer Restructuring Statistics (Restructuring Performed)

Multiplexer Inputs	Bus Width	Baseline Area	Area if Restructured	Saving if Restructured	Registered	Example Multiplexer Output
3:1	112 bits	224 LEs	0 LEs	224 LEs	Yes	|EXMEM_Stage|M_RestartPC[2]~reg0
4:1	4 bits	8 LEs	8 LEs	0 LEs	Yes	|EXMEM_Stage|M_MemRead~reg0

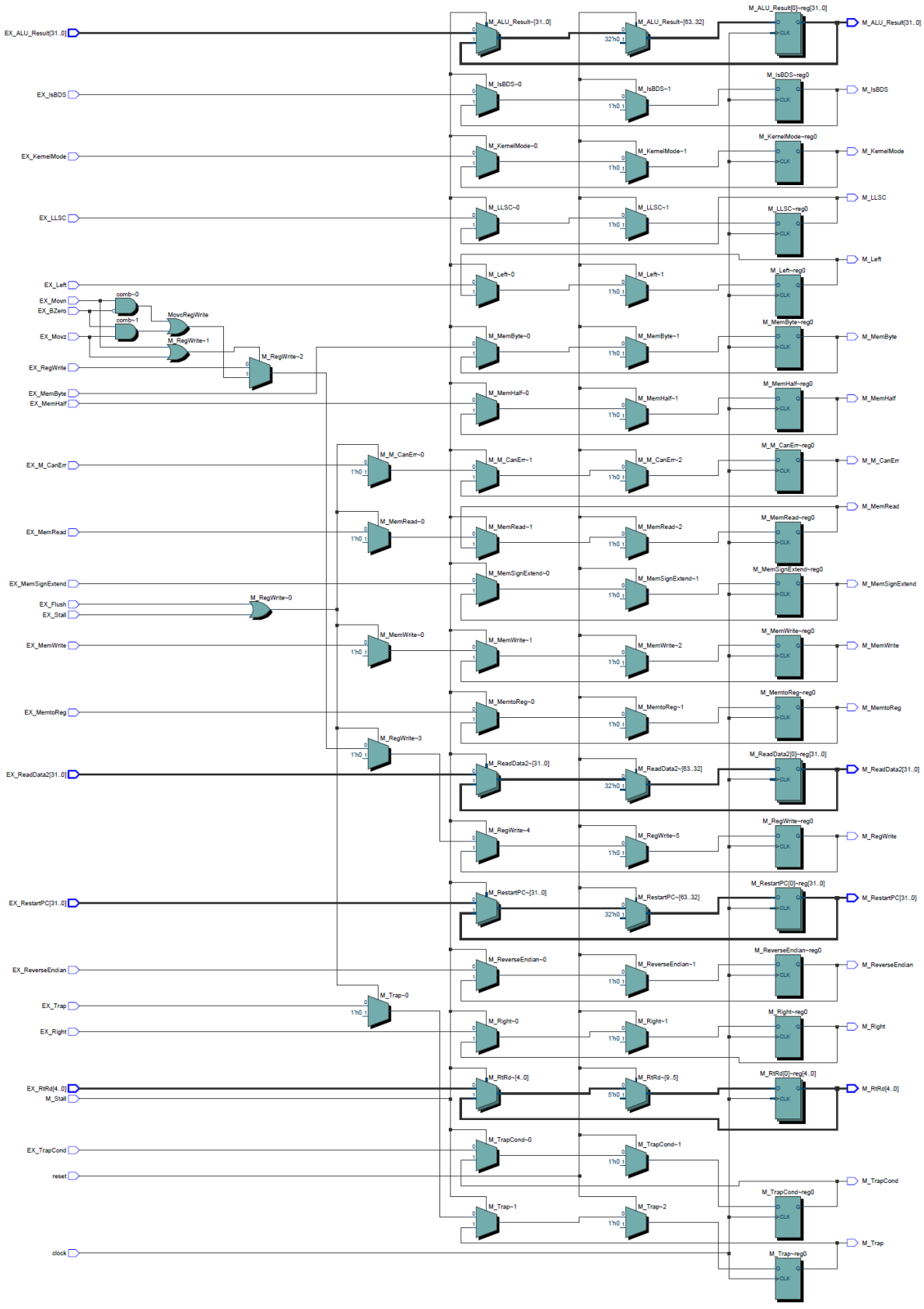
EXMEM_Stage-Post-Synthesis Netlist Statistics for Top Partition
Post-Synthesis Netlist Statistics for Top Partition report for EXMEM_Stage
Tue Sep 22 22:53:26 2015
Quartus II 64-Bit Version 15.0.0 Build 145 04/22/2015 SJ Web Edition

Post-Synthesis Netlist Statistics for Top Partition	
Type	Count
arriav_ff	117
ENA	4
ENA SCLR	112
SCLR	1
arriav_lcell_comb	7
normal	7
2 data inputs	1
4 data inputs	5
5 data inputs	1
boundary_port	242
Max LUT depth	2.00
Average LUT depth	0.10

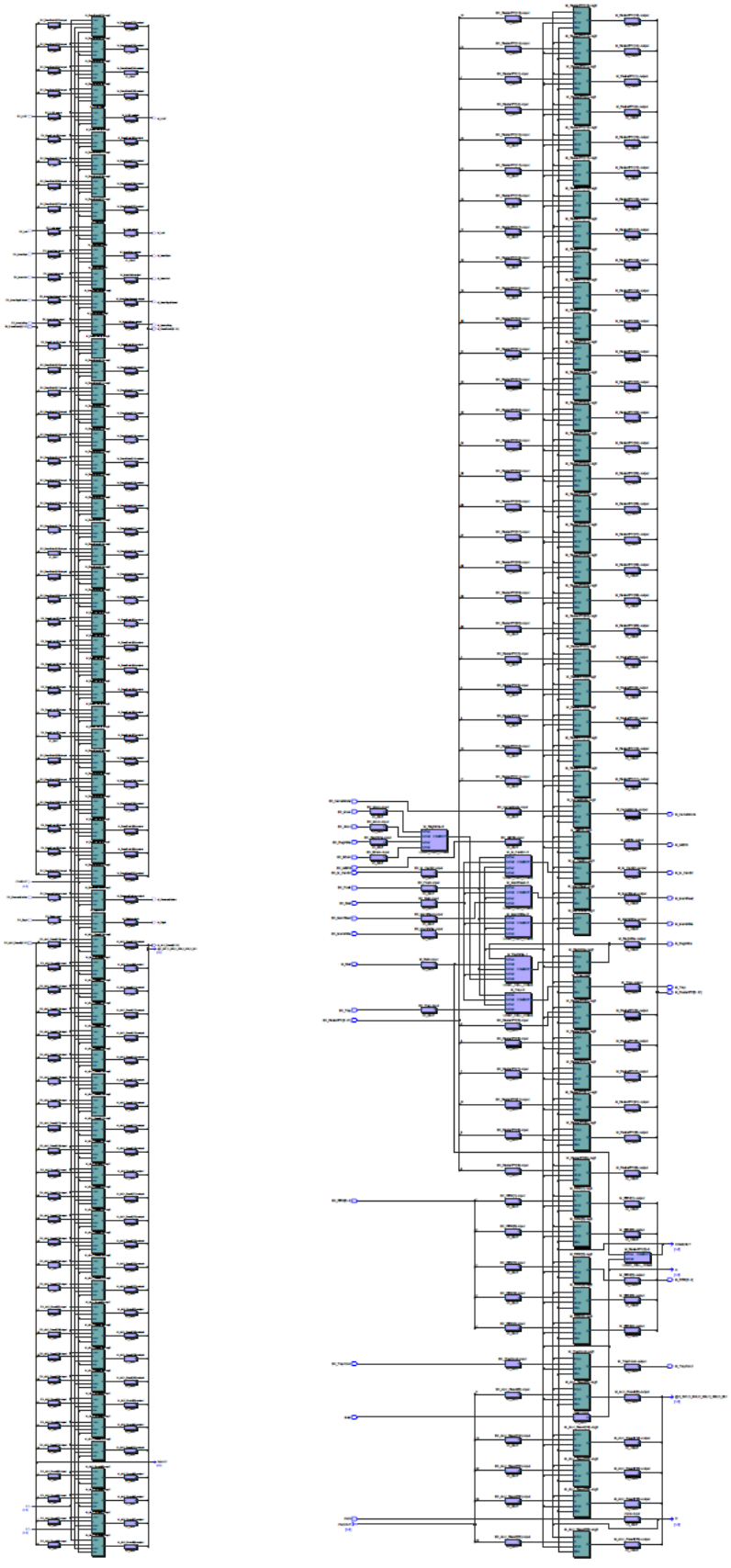
EXMEM_Stage-Partition Statistics

Partition Statistics report for EXMEM_Stage
Tue Sep 22 22:54:18 2015
Quartus II 64-Bit Version 15.0.0 Build 145 04/22/2015 SJ Web Edition

Fitter Partition Statistics		
Statistic	Top	hard_block:auto_generated_inst
Logic utilization (ALMs needed / total ALMs on device)	32 / 113560 (< 1 %)	0 / 113560 (0 %)
ALMs needed [=A-B+C]	32	0
[A] ALMs used in final placement [=a+b+c+d]	61 / 113560 (< 1 %)	0 / 113560 (0 %)
[a] ALMs used for LUT logic and registers	3	0
[b] ALMs used for LUT logic	2	0
[c] ALMs used for registers	56	0
[d] ALMs used for memory (up to half of total ALMs)	0	0
[B] Estimate of ALMs recoverable by dense packing	29 / 113560 (< 1 %)	0 / 113560 (0 %)
[C] Estimate of ALMs unavailable [=a+b+c+d]	0 / 113560 (0 %)	0 / 113560 (0 %)
[a] Due to location constrained logic	0	0
[b] Due to LAB-wide signal conflicts	0	0
[c] Due to LAB input limits	0	0
[d] Due to virtual I/Os	0	0
Difficulty packing design	Low	Low
Total LABs: partially or completely used	18 / 11356 (< 1 %)	0 / 11356 (0 %)
-- Logic LABs	18	0
-- Memory LABs (up to half of total LABs)	0	0
Combinational ALUT usage for logic	8	0
-- 7 input functions	0	0
-- 6 input functions	0	0
-- 5 input functions	1	0
-- 4 input functions	5	0
-- <=3 input functions	2	0
Combinational ALUT usage for route-throughs	63	0
Memory ALUT usage	0	0
-- 64-address deep	0	0
-- 32-address deep	0	0
Dedicated logic registers	0	0
-- By type:		
-- Primary logic registers	117 / 227120 (< 1 %)	0 / 227120 (0 %)
-- Secondary logic registers	0 / 227120 (0 %)	0 / 227120 (0 %)
-- By function:		
-- Design implementation registers	117	0
-- Routing optimization registers	0	0
Virtual pins	0	0
I/O pins	242	0
I/O registers	0	0
Total block memory bits	0	0
Total block memory implementation bits	0	0
Clock enable block	1 / 128 (< 1 %)	0 / 128 (0 %)
Connections		
-- Input Connections	0	0
-- Registered Input Connections	0	0
-- Output Connections	0	0
-- Registered Output Connections	0	0
Internal Connections		
-- Total Connections	913	0
-- Registered Connections	118	0
External Connections		
-- Top	0	0
-- hard_block:auto_generated_inst	0	0
Partition Interface		
-- Input Ports	125	0
-- Output Ports	117	0
-- Bidir Ports	0	0
Registered Ports		
-- Registered Input Ports	0	0
-- Registered Output Ports	0	0
Port Connectivity		
-- Input Ports driven by GND	0	0
-- Output Ports driven by GND	0	0
-- Input Ports driven by VCC	0	0
-- Output Ports driven by VCC	0	0
-- Input Ports with no Source	0	0
-- Output Ports with no Source	0	0
-- Input Ports with no Fanout	0	0
-- Output Ports with no Fanout	0	0



Figură 2-6 RTL EXMEM



Figură 2-7 Tehnology map EXMEM

2.1.4. MEMWB

MEMWB_Stage-Flow Summary

Flow Summary report for MEMWB_Stage
Tue Sep 22 23:11:58 2015
Quartus II 64-Bit Version 15.0.0 Build 145 04/22/2015 SJ Web Edition

Flow Summary	
Flow Status	Successful - Mon Sep 21 21:54:36 2015
Quartus II 64-Bit Version	15.0.0 Build 145 04/22/2015 SJ Web Edition
Revision Name	MEMWB_Stage
Top-level Entity Name	MEMWB_Stage
Family	Cyclone V
Device	5CEFA9F31C7
Timing Models	Final
Logic utilization (in ALMs)	19 / 113,560 (< 1 %)
Total registers	71
Total pins	147 / 480 (31 %)
Total virtual pins	0
Total block memory bits	0 / 12,492,800 (0 %)
Total DSP Blocks	0 / 342 (0 %)
Total HSSI RX PCSs	0
Total HSSI PMA RX Deserializers	0
Total HSSI TX PCSs	0
Total HSSI PMA TX Serializers	0
Total PLLs	0 / 8 (0 %)
Total DLLs	0 / 4 (0 %)

Multiplexer Restructuring Statistics (Restructuring Performed) report for MEMWB_Stage
Tue Sep 22 23:12:32 2015
Quartus II 64-Bit Version 15.0.0 Build 145 04/22/2015 SJ Web Edition

Multiplexer Restructuring Statistics (Restructuring Performed)

Multiplexer Inputs	Bus Width	Baseline Area	Area if Restructured	Saving if Restructured	Registered	Example Multiplexer Output
3:1	70 bits	140 LEs	0 LEs	140 LEs	Yes	|MEMWB_Stage|WB_ReadData[6]~reg0

MEMWB_Stage-Post-Synthesis Netlist Statistics for Top Partition

Post-Synthesis Netlist Statistics for Top Partition report for MEMWB_Stage
Tue Sep 22 23:12:39 2015
Quartus II 64-Bit Version 15.0.0 Build 145 04/22/2015 SJ Web Edition

Post-Synthesis Netlist Statistics for Top Partition

Type	Count
arriav_ff	71
ENA SCLR	70
SCLR	1
arriav_lcell_comb	2
normal	2
2 data inputs	1
5 data inputs	1
boundary_port	147
Max LUT depth	1.00
Average LUT depth	0.05

MEMWB_Stage-Partition Statistics

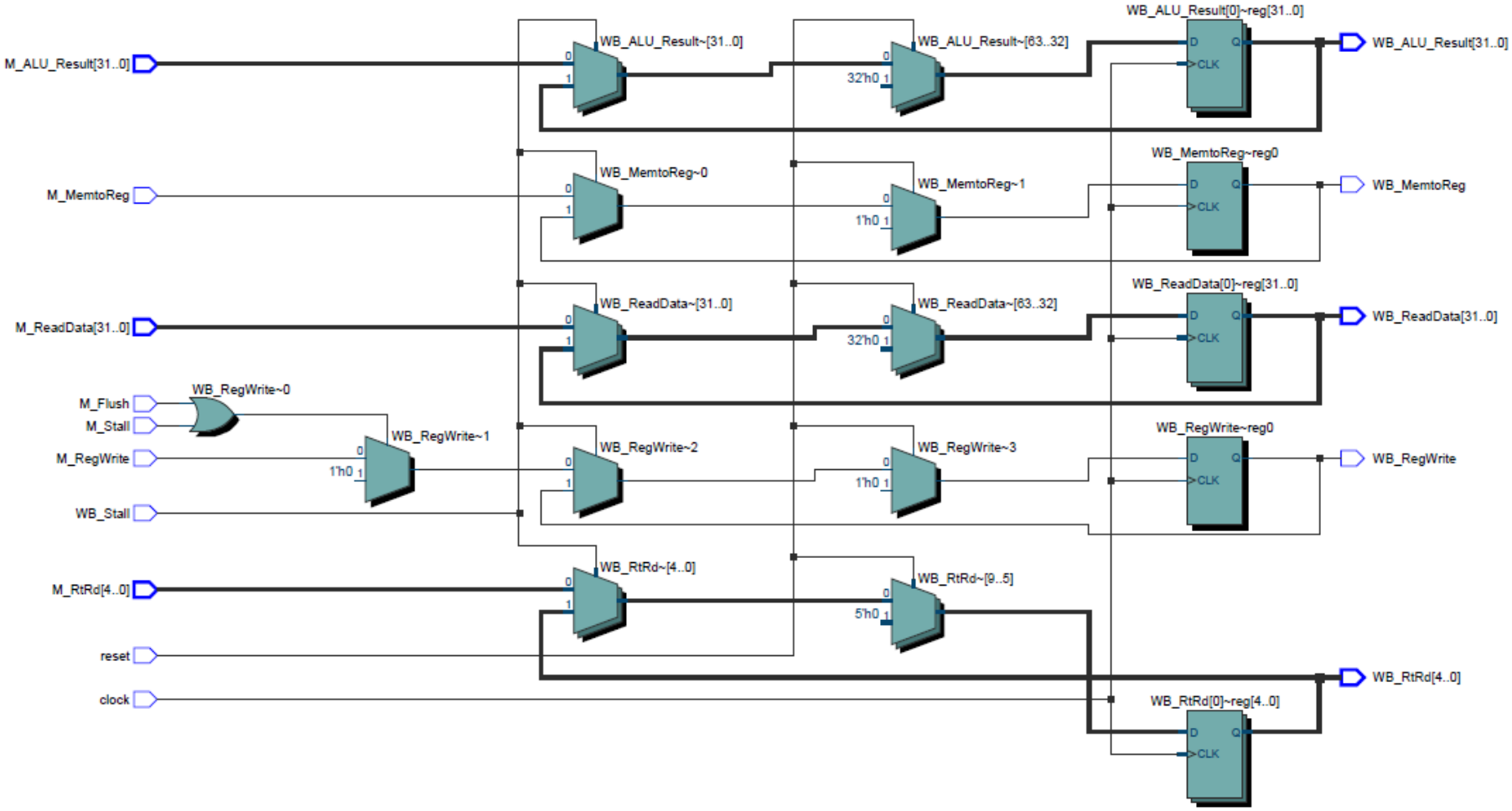
Partition Statistics report for MEMWB_Stage
Tue Sep 22 23:13:16 2015
Quartus II 64-Bit Version 15.0.0 Build 145 04/22/2015 SJ Web Edition

Fitter Partition Statistics		
Statistic	Top	hard_block:auto_generated_inst
Logic utilization (ALMs needed / total ALMs on device)	19 / 113560 (< 1 %)	0 / 113560 (0 %)
ALMs needed [=A-B+C]	19	0
[A] ALMs used in final placement [=a+b+c+d]	38 / 113560 (< 1 %)	0 / 113560 (0 %)
[a] ALMs used for LUT logic and registers	1	0
[b] ALMs used for LUT logic	2	0
[c] ALMs used for registers	35	0
[d] ALMs used for memory (up to half of total ALMs)	0	0
[B] Estimate of ALMs recoverable by dense packing	19 / 113560 (< 1 %)	0 / 113560 (0 %)
[C] Estimate of ALMs unavailable [=a+b+c+d]	0 / 113560 (0 %)	0 / 113560 (0 %)
[a] Due to location constrained logic	0	0
[b] Due to LAB-wide signal conflicts	0	0
[c] Due to LAB input limits	0	0
[d] Due to virtual I/Os	0	0
Difficulty packing design	Low	Low
Total LABs: partially or completely used	10 / 11356 (< 1 %)	0 / 11356 (0 %)
-- Logic LABs	10	0
-- Memory LABs (up to half of total LABs)	0	0
Combinational ALUT usage for logic	3	0
-- 7 input functions	0	0
-- 6 input functions	0	0
-- 5 input functions	1	0
-- 4 input functions	0	0
-- <=3 input functions	2	0
Combinational ALUT usage for route-throughs	42	0
Memory ALUT usage	0	0
-- 64-address deep	0	0
-- 32-address deep	0	0
Dedicated logic registers	0	0
-- By type:		
-- Primary logic registers	71 / 227120 (< 1 %)	0 / 227120 (0 %)
-- Secondary logic registers	0 / 227120 (0 %)	0 / 227120 (0 %)
-- By function:		
-- Design implementation registers	71	0
-- Routing optimization registers	0	0
Virtual pins	0	0
I/O pins	147	0
I/O registers	0	0
Total block memory bits	0	0
Total block memory implementation bits	0	0
Clock enable block	1 / 128 (< 1 %)	0 / 128 (0 %)
Connections		
-- Input Connections	0	0
-- Registered Input Connections	0	0
-- Output Connections	0	0
-- Registered Output Connections	0	0
Internal Connections		
-- Total Connections	551	0
-- Registered Connections	72	0
External Connections		
-- Top	0	0
-- hard_block:auto_generated_inst	0	0
Partition Interface		
-- Input Ports	76	0
-- Output Ports	71	0
-- Bidir Ports	0	0
Registered Ports		
-- Registered Input Ports	0	0
-- Registered Output Ports	0	0
Port Connectivity		
-- Input Ports driven by GND	0	0
-- Output Ports driven by GND	0	0
-- Input Ports driven by VCC	0	0
-- Output Ports driven by VCC	0	0
-- Input Ports with no Source	0	0
-- Output Ports with no Source	0	0
-- Input Ports with no Fanout	0	0

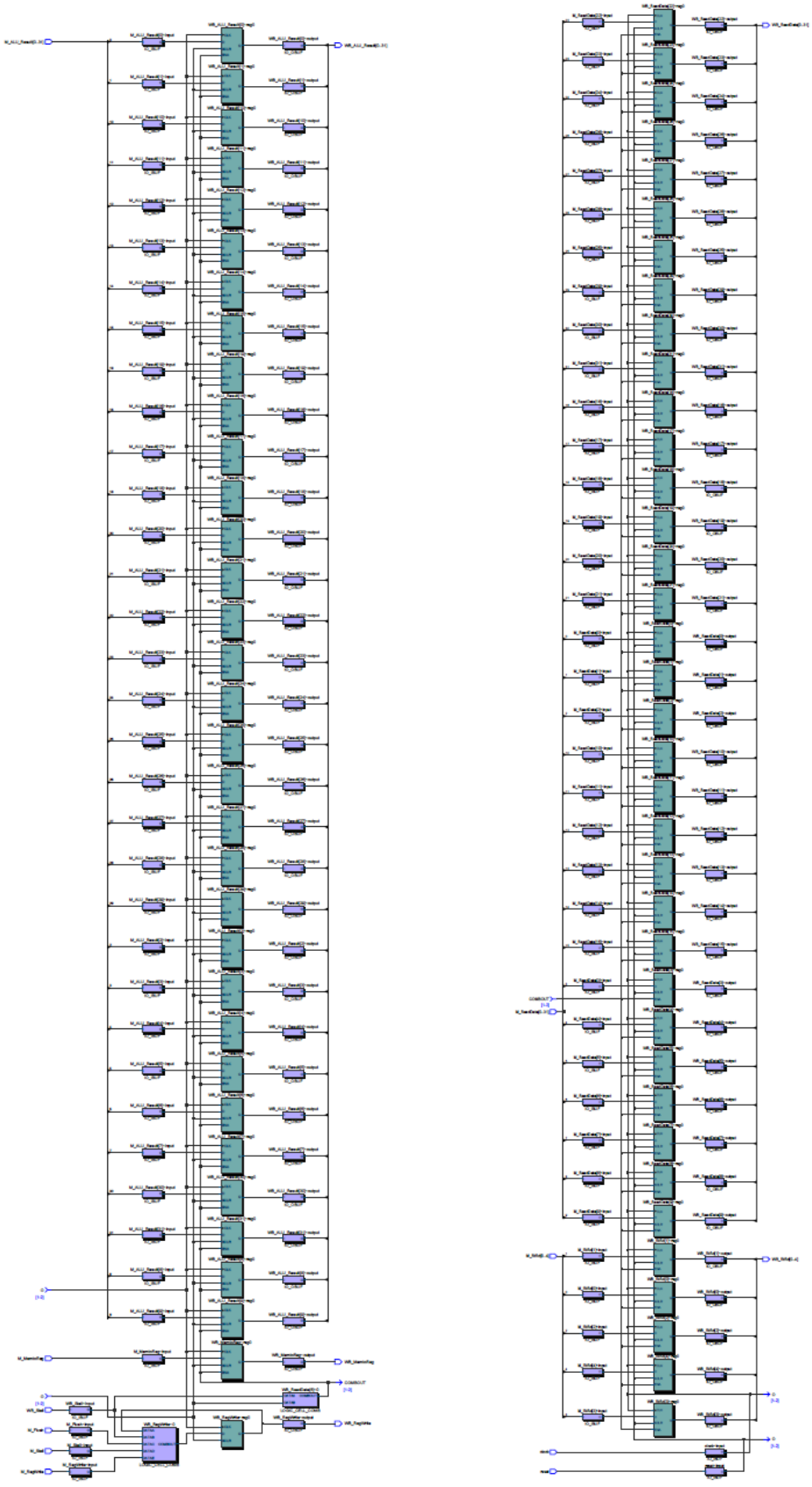
MEMWB_Stage-Routing Usage Summary

Routing Usage Summary report for MEMWB_Stage
Tue Sep 22 23:13:50 2015
Quartus II 64-Bit Version 15.0.0 Build 145 04/22/2015 SJ Web Edition

Routing Usage Summary	
Routing Resource Type	Usage
Block interconnects	163 / 721,028 (< 1 %)
C12 interconnects	0 / 30,780 (0 %)
C2 interconnects	116 / 303,248 (< 1 %)
C4 interconnects	30 / 140,620 (< 1 %)
DQS bus muxes	0 / 35 (0 %)
DQS-18 I/O buses	0 / 35 (0 %)
DQS-9 I/O buses	0 / 35 (0 %)
Direct links	0 / 721,028 (0 %)
Global clocks	1 / 16 (6 %)
Horizontal periphery clocks	0 / 96 (0 %)
Local interconnects	0 / 227,120 (0 %)
Quadrant clocks	0 / 88 (0 %)
R14 interconnects	9 / 30,168 (< 1 %)
R14/C12 interconnect drivers	8 / 53,952 (< 1 %)
R3 interconnects	103 / 331,920 (< 1 %)
R6 interconnects	61 / 622,512 (< 1 %)
Spine clocks	4 / 480 (< 1 %)
Wire stub REs	0 / 40,996 (0 %)

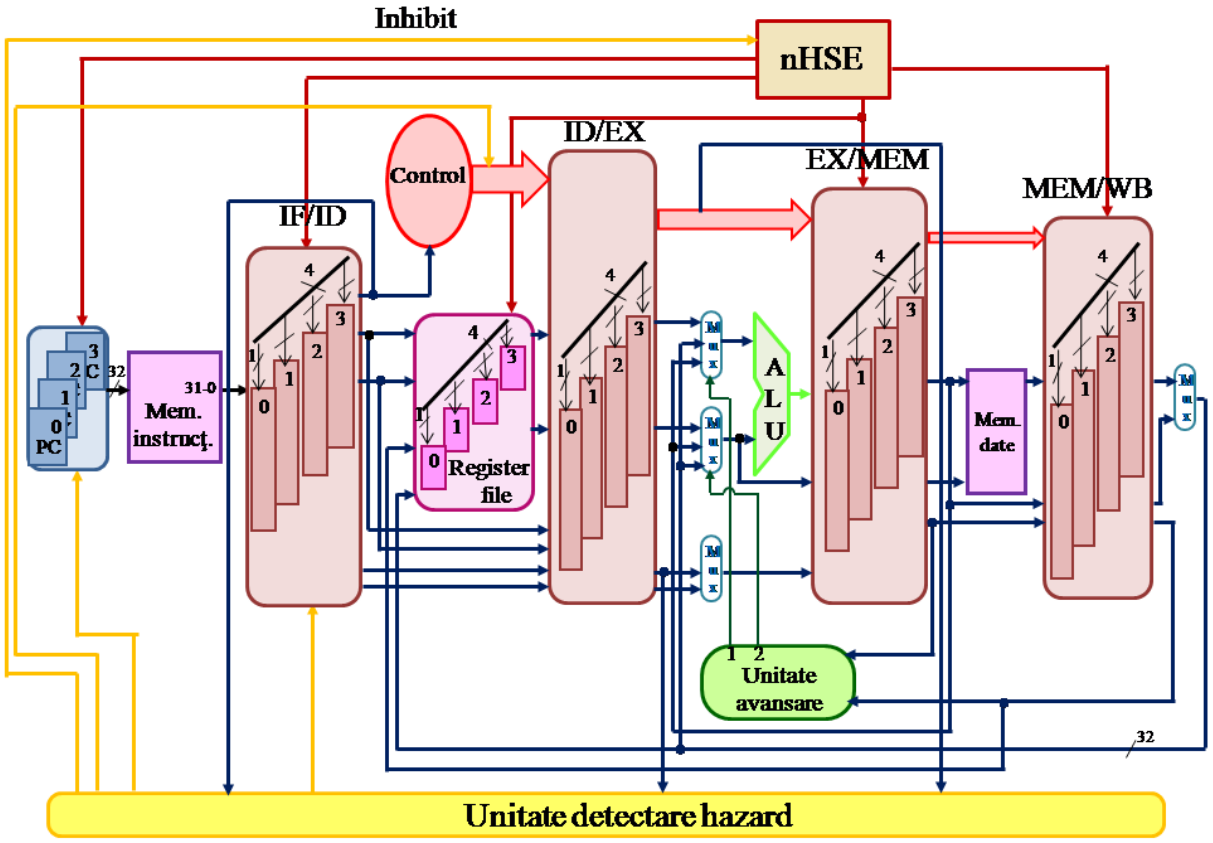


Figură 2-8 RTL MEMWB



Figură 2-9 Tehnology map MEMWB

2.2. 4MPRA



Figură 2-10 *4MPRA*

	Source Clock(s)	Destination Clock(s)	Delay Added in ns
1	clock	clock	3.7

PowerPlay Power Analyzer Status	Successful - Sat Sep 26 21:28:27 2015
Quartus II 64-Bit Version	15.0.0 Build 145 04/22/2015 SJ Web Edition
Revision Name	Processor4
Top-level Entity Name	Processor4
Family	Cyclone V
Device	5CGXFC9E7F31C8
Power Models	Final
Total Thermal Power Dissipation	536.17 mW
Core Dynamic Thermal Power Dissipation	0.00 mW
Core Static Thermal Power Dissipation	518.57 mW
I/O Thermal Power Dissipation	17.60 mW
Power Estimation Confidence	Low: user provided insufficient toggle rate data

	Source Register	Destination Register	Delay Added in ns
1	Register:IODataIn|Q[15]	Register:IODataReg|Q[15]	0.175
2	Register:IODataIn|Q[26]	Register:IODataReg|Q[26]	0.166
3	Register:IODataIn|Q[23]	Register:IODataReg|Q[23]	0.161
4	Register:IODataIn|Q[0]	Register:IODataReg|Q[0]	0.153
5	Register:IODataIn|Q[28]	Register:IODataReg|Q[28]	0.145
6	Register:IODataIn|Q[18]	Register:IODataReg|Q[18]	0.143
7	Register:IODataIn|Q[7]	Register:IODataReg|Q[7]	0.141
8	Register:IODataIn|Q[25]	Register:IODataReg|Q[25]	0.138
9	Register:IODataIn|Q[22]	Register:IODataReg|Q[22]	0.138
10	Register:IODataIn|Q[12]	Register:IODataReg|Q[12]	0.138
11	Register:IODataIn|Q[5]	Register:IODataReg|Q[5]	0.136
12	Register:IODataIn|Q[2]	Register:IODataReg|Q[2]	0.133
13	Register:IODataIn|Q[14]	Register:IODataReg|Q[14]	0.103
14	Register:IODataIn|Q[3]	Register:IODataReg|Q[3]	0.103
15	Register:IODataIn|Q[24]	Register:IODataReg|Q[24]	0.102
16	Register:IODataIn|Q[31]	Register:IODataReg|Q[31]	0.100
17	Register:IODataIn|Q[19]	Register:IODataReg|Q[19]	0.100
18	Register:IODataIn|Q[10]	Register:IODataReg|Q[10]	0.100
19	Register:IODataIn|Q[9]	Register:IODataReg|Q[9]	0.100
20	Register:IODataIn|Q[20]	Register:IODataReg|Q[20]	0.100
21	Register:IODataIn|Q[17]	Register:IODataReg|Q[17]	0.100
22	Register:IODataIn|Q[16]	Register:IODataReg|Q[16]	0.100
23	Register:IODataIn|Q[8]	Register:IODataReg|Q[8]	0.100
24	Register:IODataIn|Q[4]	Register:IODataReg|Q[4]	0.098
25	Register:IODataIn|Q[30]	Register:IODataReg|Q[30]	0.096
26	Register:IODataIn|Q[13]	Register:IODataReg|Q[13]	0.096
27	Register:IODataIn|Q[1]	Register:IODataReg|Q[1]	0.096
28	Register:IODataIn|Q[21]	Register:IODataReg|Q[21]	0.091
29	Register:IODataIn|Q[11]	Register:IODataReg|Q[11]	0.091
30	Register:IODataIn|Q[29]	Register:IODataReg|Q[29]	0.089
31	Register:IODataIn|Q[27]	Register:IODataReg|Q[27]	0.089
32	Register:IODataIn|Q[6]	Register:IODataReg|Q[6]	0.089

Processor4-Flow Summary

```
Flow Summary report for Processor4
Tue Sep 22 22:42:32 2015
Quartus II 64-Bit Version 15.0.0 Build 145 04/22/2015 SJ Web Edition

+---------------------------------------------------------------------------------+
; Flow Summary                                                                    ;
+-----------------------------------+---------------------------------------------+
; Flow Status                       ; Successful - Sat Sep 19 19:01:43 2015       ;
; Quartus II 64-Bit Version         ; 15.0.0 Build 145 04/22/2015 SJ Web Edition ;
; Revision Name                     ; Processor4                                  ;
; Top-level Entity Name             ; Processor4                                  ;
; Family                            ; Cyclone V                                   ;
; Device                            ; 5CGXFC9E7F31C8                              ;
; Timing Models                     ; Final                                       ;
; Logic utilization (in ALMs)       ; 17 / 113,560 ( < 1 % )                      ;
; Total registers                   ; 64                                          ;
; Total pins                        ; 302 / 536 ( 56 % )                          ;
; Total virtual pins                ; 0                                           ;
; Total block memory bits           ; 0 / 12,492,800 ( 0 % )                      ;
; Total DSP Blocks                  ; 0 / 342 ( 0 % )                             ;
; Total HSSI RX PCSs                ; 0 / 12 ( 0 % )                              ;
; Total HSSI PMA RX Deserializers   ; 0 / 12 ( 0 % )                              ;
; Total HSSI TX PCSs                ; 0 / 12 ( 0 % )                              ;
; Total HSSI PMA TX Serializers     ; 0 / 12 ( 0 % )                              ;
; Total PLLs                        ; 0 / 20 ( 0 % )                              ;
; Total DLLs                        ; 0 / 4 ( 0 % )                               ;
+-----------------------------------+---------------------------------------------+
```

2.2.1. IFID4

```
1
2   module  IFID4_Stage(
3
4           input       sCPU0,
5           input       sCPU1,
6           input       sCPU2,
7           input       sCPU3,
8           input       [ 1: 0]selMIPS,
9           input       clock,
10          input       reset,
11          input       IF_Flush,
12          input       IF_Stall,
13          input       ID_Stall,
14          // Control Signals
15          input       [ 31: 0]IF_Instruction,
16          // Data Signals
17          input       [ 31: 0]IF_PCAdd4,
18          input       [ 31: 0]IF_PC,
19
20          input       IF_IsBDS,
21          // ------------------
22          output      [ 31: 0]ID_Instruction,
23          output      [ 31: 0]ID_PCAdd4,
24          output      [ 31: 0]ID_RestartPC,
25          output      ID_IsBDS,
26          output      ID_IsFlushed
27          );
28
29  sCPU_IFID_StageIFID0(
30          .if1(clock&sCPU0),
31          .if2(reset),
32          .if3(IF_Flush),
33          .if4(IF_Stall),
34          .if5(ID_Stall),
35          // Control Signals
36          .if6(IF_Instruction),
37          // Data Signals
38          .if7(IF_PCAdd4),
39          .if8(IF_PC),
40          .if9(IF_IsBDS),
41          // ------------------
42          .id1(ID_Instruction0),
43          .id2(ID_PCAdd40),
44          .id3(ID_RestartPC0),
45          .id4(ID_IsBDS0),
46          .id5(ID_IsFlushed0)
47          );
48
49          wire    [ 31: 0]ID_Instruction0;
50          wire    [ 31: 0]ID_PCAdd40;
51          wire    [ 31: 0]ID_RestartPC0;
52          wire    ID_IsBDS0;
53          wire    ID_IsFlushed0;
54
55  sCPU_IFID_StageIFID1(
56          .if1(clock&sCPU1),
57          .if2(reset),
58          .if3(IF_Flush),
59          .if4(IF_Stall),
60          .if5(ID_Stall),
61          // Control Signals
62          .if6(IF_Instruction),
63          // Data Signals
64          .if7(IF_PCAdd4),
65          .if8(IF_PC),
66          .if9(IF_IsBDS),
67          // ------------------
68          .id1(ID_Instruction1),
69          .id2(ID_PCAdd41),
70          .id3(ID_RestartPC1),
71          .id4(ID_IsBDS1),
72          .id5(ID_IsFlushed1)
73          );
74
75          wire    [ 31: 0]ID_Instruction1;
76          wire    [ 31: 0]ID_PCAdd41;
77          wire    [ 31: 0]ID_RestartPC1;
78          wire    ID_IsBDS1;
79          wire    ID_IsFlushed1;
80
81  sCPU_IFID_StageIFID2(
82          .if1(clock&sCPU2),
83          .if2(reset),
84          .if3(IF_Flush),
85          .if4(IF_Stall),
86          .if5(ID_Stall),
87          // Control Signals
88          .if6(IF_Instruction),
89          // Data Signals
90          .if7(IF_PCAdd4),
91          .if8(IF_PC),
92          .if9(IF_IsBDS),
93          // ------------------
94          .id1(ID_Instruction2),
95          .id2(ID_PCAdd42),
96          .id3(ID_RestartPC2),
97          .id4(ID_IsBDS2),
98          .id5(ID_IsFlushed2)
99          );
100
101         wire    [ 31: 0]ID_Instruction2;
102         wire    [ 31: 0]ID_PCAdd42;
103         wire    [ 31: 0]ID_RestartPC2;
104         wire    ID_IsBDS2;
105         wire    ID_IsFlushed2;
106
107 sCPU_IFID_StageIFID3(
108         .if1(clock&sCPU3),
109         .if2(reset),
110         .if3(IF_Flush),
111         .if4(IF_Stall),
112         .if5(ID_Stall),
113         // Control Signals
114         .if6(IF_Instruction),
115         // Data Signals
116         .if7(IF_PCAdd4),
117         .if8(IF_PC),
118         .if9(IF_IsBDS),
119         // ------------------
120         .id1(ID_Instruction3),
121         .id2(ID_PCAdd43),
122         .id3(ID_RestartPC3),
123         .id4(ID_IsBDS3),
124         .id5(ID_IsFlushed3)
125         );
126
127         wire    [ 31: 0]ID_Instruction3;
128         wire    [ 31: 0]ID_PCAdd43;
129         wire    [ 31: 0]ID_RestartPC3;
130         wire    ID_IsBDS3;
131         wire    ID_IsFlushed3;
132
133 Mux4#(.WIDTH(   32))ID_Instruction_Mux(
134             .sel(selMIPS),
135             .in0(ID_Instruction0),
136             .in1(ID_Instruction1),
137             .in2(ID_Instruction2),
138             .in3(ID_Instruction3),
139             .out(ID_Instruction)
140             );
141
142 Mux4#(.WIDTH(   32))ID_PCAdd4_Mux(
143             .sel(selMIPS),
144             .in0(ID_PCAdd40),
145             .in1(ID_PCAdd41),
146             .in2(ID_PCAdd42),
147             .in3(ID_PCAdd43),
148             .out(ID_PCAdd4)
149             );
150
151 Mux4#(.WIDTH(   32))ID_RestartPC_Mux(
152             .sel(selMIPS),
153             .in0(ID_RestartPC0),
154             .in1(ID_RestartPC1),
155             .in2(ID_RestartPC2),
156             .in3(ID_RestartPC3),
157             .out(ID_RestartPC)
158             );
159
160 Mux4#(.WIDTH(   1))ID_IsBDS_Mux(
161             .sel(selMIPS),
162             .in0(ID_IsBDS0),
163             .in1(ID_IsBDS1),
164             .in2(ID_IsBDS2),
165             .in3(ID_IsBDS3),
166             .out(ID_IsBDS)
167             );
168
169 Mux4#(.WIDTH(   1))ID_IsFlushed_Mux(
170             .sel(selMIPS),
171             .in0(ID_IsFlushed0),
172             .in1(ID_IsFlushed1),
173             .in2(ID_IsFlushed2),
174             .in3(ID_IsFlushed3),
175             .out(ID_IsFlushed)
176             );
177
178 endmodule
179
```

Codul verilog pentru IFID4

IFID4_Stage-Flow Summary

Flow Summary report for IFID4_Stage
Tue Sep 22 22:05:23 2015
Quartus II 64-Bit Version 15.0.0 Build 145 04/22/2015 SJ Web Edition

Flow Summary	
Flow Status	Successful - Sat Sep 19 11:37:15 2015
Quartus II 64-Bit Version	15.0.0 Build 145 04/22/2015 SJ Web Edition
Revision Name	IFID4_Stage
Top-level Entity Name	IFID4_Stage
Family	Cyclone V
Device	5CGXBC9E7F31C8
Timing Models	Final
Logic utilization (in ALMs)	1 / 113,560 (< 1 %)
Total registers	0
Total pins	206 / 536 (38 %)
Total virtual pins	0
Total block memory bits	0 / 12,492,800 (0 %)
Total DSP Blocks	0 / 342 (0 %)
Total HSSI RX PCSs	0 / 12 (0 %)
Total HSSI PMA RX Deserializers	0 / 12 (0 %)
Total HSSI TX PCSs	0 / 12 (0 %)
Total HSSI PMA TX Serializers	0 / 12 (0 %)
Total PLLs	0 / 20 (0 %)
Total DLLs	0 / 4 (0 %)

IFID4_Stage-Partition Statistics

Partition Statistics report for IFID4_Stage
Tue Sep 22 22:07:29 2015
Quartus II 64-Bit Version 15.0.0 Build 145 04/22/2015 SJ Web Edition

Fitter Partition Statistics		
Statistic	Top	hard_block:auto_generated_inst
Logic utilization (ALMs needed / total ALMs on device)	1 / 113560 (< 1 %)	0 / 113560 (0 %)
ALMs needed [=A-B+C]	1	0
[A] ALMs used in final placement [=a+b+c+d]	1 / 113560 (< 1 %)	0 / 113560 (0 %)
[a] ALMs used for LUT logic and registers	0	0
[b] ALMs used for LUT logic	1	0
[c] ALMs used for registers	0	0
[d] ALMs used for memory (up to half of total ALMs)	0	0
[B] Estimate of ALMs recoverable by dense packing	0 / 113560 (0 %)	0 / 113560 (0 %)
[C] Estimate of ALMs unavailable [=a+b+c+d]	0 / 113560 (0 %)	0 / 113560 (0 %)
[a] Due to location constrained logic	0	0
[b] Due to LAB-wide signal conflicts	0	0
[c] Due to LAB input limits	0	0
[d] Due to virtual I/Os	0	0
Difficulty packing design	Low	Low
Total LABs: partially or completely used	1 / 11356 (< 1 %)	0 / 11356 (0 %)
-- Logic LABs	1	0
-- Memory LABs (up to half of total LABs)	0	0
Combinational ALUT usage for logic	1	0
-- 7 input functions	0	0
-- 6 input functions	0	0
-- 5 input functions	0	0
-- 4 input functions	0	0
-- <=3 input functions	1	0
Combinational ALUT usage for route-throughs	0	0
Memory ALUT usage	0	0
-- 64-address deep	0	0
-- 32-address deep	0	0
Dedicated logic registers	0	0
-- By type:		
-- Primary logic registers	0 / 227120 (0 %)	0 / 227120 (0 %)
-- Secondary logic registers	0 / 227120 (0 %)	0 / 227120 (0 %)
-- By function:		

-- Design implementation registers	0	0
-- Routing optimization registers	0	0
Virtual pins	0	0
I/O pins	206	0
I/O registers	0	0
Total block memory bits	0	0
Total block memory implementation bits	0	0
Connections		
-- Input Connections	0	0
-- Registered Input Connections	0	0
-- Output Connections	0	0
-- Registered Output Connections	0	0
Internal Connections		
-- Total Connections	206	0
-- Registered Connections	0	0
External Connections		
-- Top	0	0
-- hard_block:auto_generated_inst	0	0
Partition Interface		
-- Input Ports	108	0
-- Output Ports	98	0
-- Bidir Ports	0	0
Registered Ports		
-- Registered Input Ports	0	0
-- Registered Output Ports	0	0
Port Connectivity		
-- Input Ports driven by GND	0	0
-- Output Ports driven by GND	0	0
-- Input Ports driven by VCC	0	0
-- Output Ports driven by VCC	0	0
-- Input Ports with no Source	0	0
-- Output Ports with no Source	0	0
-- Input Ports with no Fanout	0	0
-- Output Ports with no Fanout	0	0

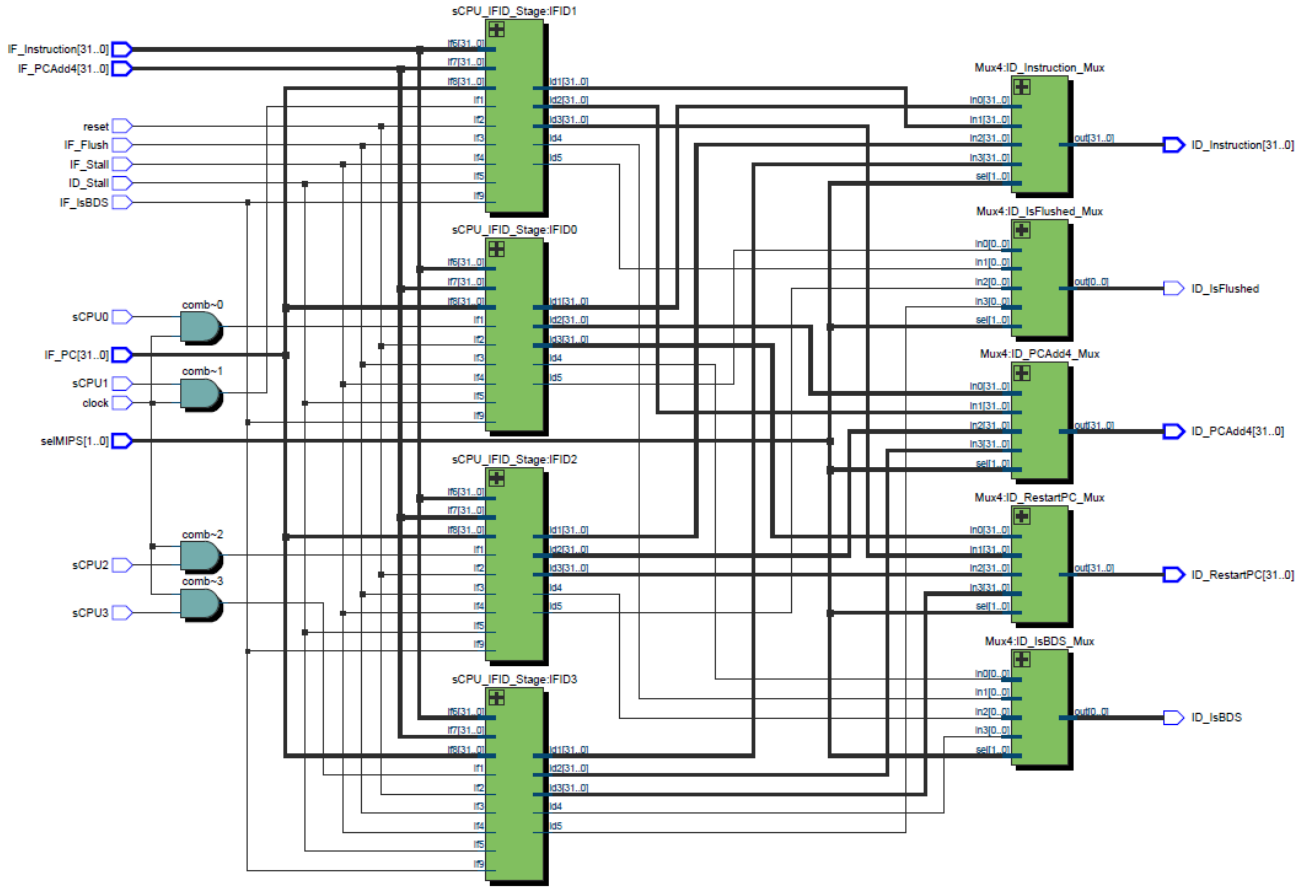
IFID4_Stage-Resource Usage Summary

Resource Usage Summary report for IFID4_Stage
Tue Sep 22 22:07:18 2015
Quartus II 64-Bit Version 15.0.0 Build 145 04/22/2015 SJ Web Edition

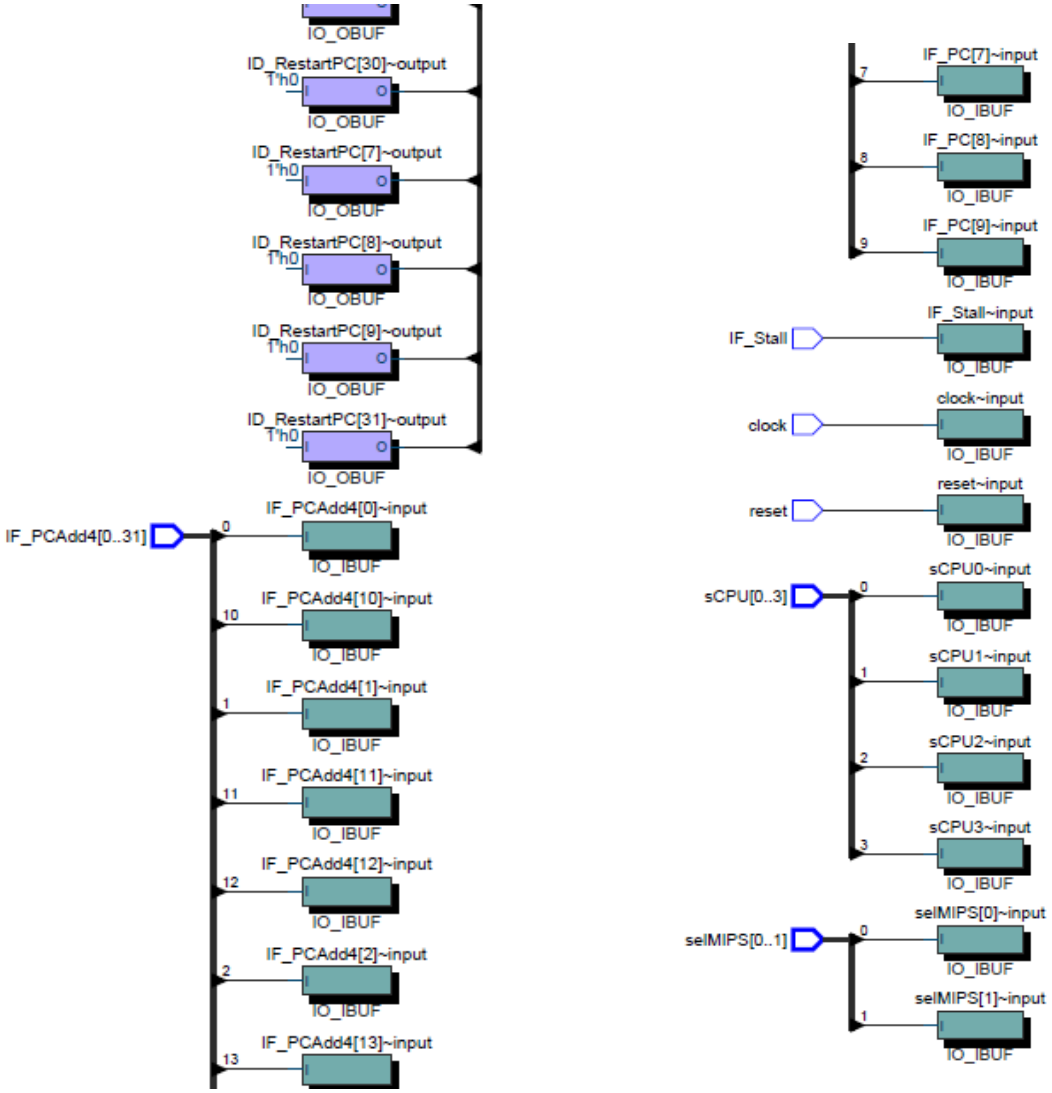
Fitter Resource Usage Summary

Resource	Usage	%
Logic utilization (ALMs needed / total ALMs on device)	1 / 113,560	< 1 %
ALMs needed [=A-B+C]	1	
[A] ALMs used in final placement [=a+b+c+d]	1 / 113,560	< 1 %
[a] ALMs used for LUT logic and registers	0	
[b] ALMs used for LUT logic	1	
[c] ALMs used for registers	0	
[d] ALMs used for memory (up to half of total ALMs)	0	
[B] Estimate of ALMs recoverable by dense packing	0 / 113,560	0 %
[C] Estimate of ALMs unavailable [=a+b+c+d]	0 / 113,560	0 %
[a] Due to location constrained logic	0	
[b] Due to LAB-wide signal conflicts	0	
[c] Due to LAB input limits	0	
[d] Due to virtual I/Os	0	
Difficulty packing design	Low	
Total LABs: partially or completely used	1 / 11,356	< 1 %
-- Logic LABs	1	
-- Memory LABs (up to half of total LABs)	0	
Combinational ALUT usage for logic	1	
-- 7 input functions	0	
-- 6 input functions	0	
-- 5 input functions	0	
-- 4 input functions	0	
-- <=3 input functions	1	
Combinational ALUT usage for route-throughs	0	
Dedicated logic registers	0	
-- By type:		
-- Primary logic registers	0 / 227,120	0 %
-- Secondary logic registers	0 / 227,120	0 %
-- By function:		
-- Design implementation registers	0	
-- Routing optimization registers	0	
Virtual pins	0	
I/O pins	206 / 536	38 %
-- Clock pins	5 / 16	31 %

-- Dedicated input pins	0 / 35	0 %
Global signals	0	
M10K blocks	0 / 1,220	0 %
Total MLAB memory bits	0	
Total block memory bits	0 / 12,492,800	0 %
Total block memory implementation bits	0 / 12,492,800	0 %
Total DSP Blocks	0 / 342	0 %
Fractional PLLs	0 / 8	0 %
Global clocks	0 / 16	0 %
Quadrant clocks	0 / 88	0 %
Horizontal periphery clocks	0 / 24	0 %
SERDES Transmitters	0 / 140	0 %
SERDES Receivers	0 / 140	0 %
JTAGs	0 / 1	0 %
ASMI blocks	0 / 1	0 %
CRC blocks	0 / 1	0 %
Remote update blocks	0 / 1	0 %
Oscillator blocks	0 / 1	0 %
Standard RX PCSs	0 / 12	0 %
HSSI PMA RX Deserializers	0 / 12	0 %
Standard TX PCSs	0 / 12	0 %
HSSI PMA TX Serializers	0 / 12	0 %
Channel PLLs	0 / 12	0 %
Impedance control blocks	0 / 3	0 %
Average interconnect usage (total/H/V)	0.0% / 0.0% / 0.0%	
Peak interconnect usage (total/H/V)	0.0% / 0.0% / 0.0%	
Maximum fan-out	1	
Highest non-global fan-out	1	
Total fan-out	206	
Average fan-out	0.50	



Figură 2-11 RTL IFID4



Figură 2-12 Parţial Tehnology map IFID4

2.2.2. IDEX4

```
module  IDEX4_Stage(
        input       sCPU0,
        input       sCPU1,
        input       sCPU2,
        input       sCPU3,
        input       [ 1: 0]selMIPS,
        input       clock,
        input       reset,
        input       ID_Flush,
        input       ID_Stall,
        input       EX_Stall,
        // Control Signals
        input       ID_Link,
        input       ID_RegDst,
        input       ID_ALUSrcImm,
        input       [ 4: 0]ID_ALUOp,
        input       ID_Movn,
        input       ID_Movz,
        input       ID_LLSC,
        input       ID_MemRead,
        input       ID_MemWrite,
        input       ID_MemByte,
        input       ID_MemHalf,
        input       ID_MemSignExtend,
        input       ID_Left,
        input       ID_Right,
        input       ID_RegWrite,
        input       ID_MemtoReg,
        input       ID_ReverseEndian,
        // Hazard & Forwarding
        input       [ 4: 0]ID_Rs,
        input       [ 4: 0]ID_Rt,
        input       ID_WantRsByEX,
        input       ID_NeedRsByEX,
        input       ID_WantRtByEX,
        input       ID_NeedRtByEX,
        // Exception Control/Info
        input       ID_KernelMode,
        input       [ 31: 0]ID_RestartPC,
        input       ID_IsBDS,
        input       ID_Trap,
        input       ID_TrapCond,
        input       ID_EX_CanErr,
        input       ID_M_CanErr,
        // Data Signals
        input       [ 31: 0]ID_ReadData1,
        input       [ 31: 0]ID_ReadData2,
        input       [ 16: 0]ID_SignExtImm,
        // ----------------
        output      EX_Link,
        output      [ 1: 0]EX_LinkRegDst,
        output      EX_ALUSrcImm,
        output      [ 4: 0]EX_ALUOp,
        output      EX_Movn,
        output      EX_Movz,
        output      EX_LLSC,
        output      EX_MemRead,
        output      EX_MemWrite,
        output      EX_MemByte,
        output      EX_MemHalf,
        output      EX_MemSignExtend,
        output      EX_Left,
        output      EX_Right,
        output      EX_RegWrite,
        output      EX_MemtoReg,
        output      EX_ReverseEndian,
        output      [ 4: 0]EX_Rs,
        output      [ 4: 0]EX_Rt,
        output      EX_WantRsByEX,
        output      EX_NeedRsByEX,
        output      EX_WantRtByEX,
        output      EX_NeedRtByEX,
        output      EX_KernelMode,
        output      [ 31: 0]EX_RestartPC,
        output      EX_IsBDS,
        output      EX_Trap,
        output      EX_TrapCond,
        output      EX_EX_CanErr,
        output      EX_M_CanErr,
        output      [ 31: 0]EX_ReadData1,
        output      [ 31: 0]EX_ReadData2,
        output      [ 31: 0]EX_SignExtImm
        output      [ 4: 0]EX_Rd,
        output      [ 4: 0]EX_Shamt
        );



sCPU_IDEX_StageIDEXzero(
        .id1(clock&sCPU0),
        .id2(reset),
        .id3(ID_Flush),
        .id4(ID_Stall),
        .id5(EX_Stall),
        // Control Signals
        .id6(ID_Link),
        .id7(ID_RegDst),
        .id8(ID_ALUSrcImm),
        .id9(ID_ALUOp),
        .id10(ID_Movn),
        .id11(ID_Movz),
        .id12(ID_LLSC),
        .id13(ID_MemRead),
        .id14(ID_MemWrite),
        .id15(ID_MemByte),
        .id16(ID_MemHalf),
        .id17(ID_MemSignExtend),
        .id18(ID_Left),
        .id19(ID_Right),
        .id20(ID_RegWrite),
        .id21(ID_MemtoReg),
        .id22(ID_ReverseEndian),
        // Hazard & Forwarding
        .id23(ID_Rs),
        .id24(ID_Rt),
        .id25(ID_WantRsByEX),
        .id26(ID_NeedRsByEX),
        .id27(ID_WantRtByEX),
        .id28(ID_NeedRtByEX),
        // Exception Control/Info
        .id29(ID_KernelMode),
        .id30(ID_RestartPC),
        .id31(ID_IsBDS),
        .id32(ID_Trap),
        .id33(ID_TrapCond),
        .id34(ID_EX_CanErr),
        .id35(ID_M_CanErr),
        // Data Signals
        .id36(ID_ReadData1),
        .id37(ID_ReadData2),
        .id38(ID_SignExtImm),
        // ----------------
        .ex1(EX_Link0),
        .ex2(EX_LinkRegDst0),
        .ex3(EX_ALUSrcImm0),
        .ex4(EX_ALUOp0),
        .ex5(EX_Movn0),
        .ex6(EX_Movz0),
        .ex7(EX_LLSC0),
        .ex8(EX_MemRead0),
        .ex9(EX_MemWrite0),
        .ex10(EX_MemByte0),
        .ex11(EX_MemHalf0),
        .ex12(EX_MemSignExtend0),
        .ex13(EX_Left0),
        .ex14(EX_Right0),
        .ex15(EX_RegWrite0),
        .ex16(EX_MemtoReg0),
        .ex17(EX_ReverseEndian0),
        .ex18(EX_Rs0),
        .ex19(EX_Rt0),
        .ex20(EX_WantRsByEX0),
        .ex21(EX_NeedRsByEX0),
        .ex22(EX_WantRtByEX0),
        .ex23(EX_NeedRtByEX0),
        .ex24(EX_KernelMode0),
        .ex25(EX_RestartPC0),
        .ex26(EX_IsBDS0),
        .ex27(EX_Trap0),
        .ex28(EX_TrapCond0),
        .ex29(EX_EX_CanErr0),
        .ex30(EX_M_CanErr0),
        .ex31(EX_ReadData10),
        .ex32(EX_ReadData20),
        .ex33(EX_SignExtImm0),
        .ex34(EX_Rd0),
        .ex35(EX_Shamt0)
        );


        wire    EX_Link0;
        wire    [ 1: 0]EX_LinkRegDst0;
        wire    EX_ALUSrcImm0;
        wire    [ 4: 0]EX_ALUOp0;
        wire    EX_Movn0;
        wire    EX_Movz0;
        wire    EX_LLSC0;
        wire    EX_MemRead0;
        wire    EX_MemWrite0;
        wire    EX_MemByte0;
        wire    EX_MemHalf0;
        wire    EX_MemSignExtend0;
        wire    EX_Left0;
        wire    EX_Right0;
```

```
187         wire    EX_RegWrite0;
188         wire    EX_MemtoReg0;
189         wire    EX_ReverseEndian0;
190         wire    [ 4: 0]EX_Rs0;
191         wire    [ 4: 0]EX_Rt0;
192         wire    EX_WantRsByEX0;
193         wire    EX_NeedRsByEX0;
194         wire    EX_WantRtByEX0;
195         wire    EX_NeedRtByEX0;
196         wire    EX_KernelMode0;
197         wire    [ 31: 0]EX_RestartPC0;
198         wire    EX_IsBDS0;
199         wire    EX_Trap0;
200         wire    EX_TrapCond0;
201         wire    EX_EX_CanErr0;
202         wire    EX_M_CanErr0;
203         wire    [ 31: 0]EX_ReadData10;
204         wire    [ 31: 0]EX_ReadData20;
205         wire    [ 31: 0]EX_SignExtImm0;
206         wire    [ 4: 0]EX_Rd0;
207         wire    [ 4: 0]EX_Shamt0;
208
209
210
211     sCPU_IDEX_StageIDEXunu(
212         .id1(clock&sCPU1),
213         .id2(reset),
214         .id3(ID_Flush),
215         .id4(ID_Stall),
216         .id5(EX_Stall),
217         // Control Signals
218         .id6(ID_Link),
219         .id7(ID_RegDst),
220         .id8(ID_ALUSrcImm),
221         .id9(ID_ALUOp),
222         .id10(ID_Movn),
223         .id11(ID_Movz),
224         .id12(ID_LLSC),
225         .id13(ID_MemRead),
226         .id14(ID_MemWrite),
227         .id15(ID_MemByte),
228         .id16(ID_MemHalf),
229         .id17(ID_MemSignExtend),
230         .id18(ID_Left),
231         .id19(ID_Right),
232         .id20(ID_RegWrite),
233         .id21(ID_MemtoReg),
234         .id22(ID_ReverseEndian),
235         // Hazard & Forwarding
236         .id23(ID_Rs),
237         .id24(ID_Rt),
238         .id25(ID_WantRsByEX),
239         .id26(ID_NeedRsByEX),
240         .id27(ID_WantRtByEX),
241         .id28(ID_NeedRtByEX),
242         // Exception Control/Info
243         .id29(ID_KernelMode),
244         .id30(ID_RestartPC),
245         .id31(ID_IsBDS),
246         .id32(ID_Trap),
247         .id33(ID_TrapCond),
248         .id34(ID_EX_CanErr),
249         .id35(ID_M_CanErr),
250         // Data Signals
251         .id36(ID_ReadData1),
252         .id37(ID_ReadData2),
253         .id38(ID_SignExtImm),
254         // ----------------
255         .ex1(EX_Link1),
256         .ex2(EX_LinkRegDst1),
257         .ex3(EX_ALUSrcImm1),
258         .ex4(EX_ALUOp1),
259         .ex5(EX_Movn1),
260         .ex6(EX_Movz1),
261         .ex7(EX_LLSC1),
262         .ex8(EX_MemRead1),
263         .ex9(EX_MemWrite1),
264         .ex10(EX_MemByte1),
265         .ex11(EX_MemHalf1),
266         .ex12(EX_MemSignExtend1),
267         .ex13(EX_Left1),
268         .ex14(EX_Right1),
269         .ex15(EX_RegWrite1),
270         .ex16(EX_MemtoReg1),
271         .ex17(EX_ReverseEndian1),
272         .ex18(EX_Rs1),
273         .ex19(EX_Rt1),
274         .ex20(EX_WantRsByEX1),
275         .ex21(EX_NeedRsByEX1),
276         .ex22(EX_WantRtByEX1),
277         .ex23(EX_NeedRtByEX1),
278         .ex24(EX_KernelMode1),
279         .ex25(EX_RestartPC1),
280         .ex26(EX_IsBDS1),
281         .ex27(EX_Trap1),
282         .ex28(EX_TrapCond1),
283         .ex29(EX_EX_CanErr1),
284         .ex30(EX_M_CanErr1),
285         .ex31(EX_ReadData11),
286         .ex32(EX_ReadData21),
287         .ex33(EX_SignExtImm1),
288         .ex34(EX_Rd1),
289         .ex35(EX_Shamt1)
290         );
291
292
293         wire    EX_Link1;
294         wire    [ 1: 0]EX_LinkRegDst1;
295         wire    EX_ALUSrcImm1;
296         wire    [ 4: 0]EX_ALUOp1;
297         wire    EX_Movn1;
298         wire    EX_Movz1;
299         wire    EX_LLSC1;
300         wire    EX_MemRead1;
301         wire    EX_MemWrite1;
302         wire    EX_MemByte1;
303         wire    EX_MemHalf1;
304         wire    EX_MemSignExtend1;
305         wire    EX_Left1;
306         wire    EX_Right1;
307         wire    EX_RegWrite1;
308         wire    EX_MemtoReg1;
309         wire    EX_ReverseEndian1;
310         wire    [ 4: 0]EX_Rs1;
311         wire    [ 4: 0]EX_Rt1;
312         wire    EX_WantRsByEX1;
313         wire    EX_NeedRsByEX1;
314         wire    EX_WantRtByEX1;
315         wire    EX_NeedRtByEX1;
316         wire    EX_KernelMode1;
317         wire    [ 31: 0]EX_RestartPC1;
318         wire    EX_IsBDS1;
319         wire    EX_Trap1;
320         wire    EX_TrapCond1;
321         wire    EX_EX_CanErr1;
322         wire    EX_M_CanErr1;
323         wire    [ 31: 0]EX_ReadData11;
324         wire    [ 31: 0]EX_ReadData21;
325         wire    [ 31: 0]EX_SignExtImm1;
326         wire    [ 4: 0]EX_Rd1;
327         wire    [ 4: 0]EX_Shamt1;
328
329     sCPU_IDEX_StageIDEXdoi(
330         .id1(clock&sCPU2),
331         .id2(reset),
332         .id3(ID_Flush),
333         .id4(ID_Stall),
334         .id5(EX_Stall),
335         // Control Signals
336         .id6(ID_Link),
337         .id7(ID_RegDst),
338         .id8(ID_ALUSrcImm),
339         .id9(ID_ALUOp),
340         .id10(ID_Movn),
341         .id11(ID_Movz),
342         .id12(ID_LLSC),
343         .id13(ID_MemRead),
344         .id14(ID_MemWrite),
345         .id15(ID_MemByte),
346         .id16(ID_MemHalf),
347         .id17(ID_MemSignExtend),
348         .id18(ID_Left),
349         .id19(ID_Right),
350         .id20(ID_RegWrite),
351         .id21(ID_MemtoReg),
352         .id22(ID_ReverseEndian),
353         // Hazard & Forwarding
354         .id23(ID_Rs),
355         .id24(ID_Rt),
356         .id25(ID_WantRsByEX),
357         .id26(ID_NeedRsByEX),
358         .id27(ID_WantRtByEX),
359         .id28(ID_NeedRtByEX),
360         // Exception Control/Info
361         .id29(ID_KernelMode),
362         .id30(ID_RestartPC),
363         .id31(ID_IsBDS),
364         .id32(ID_Trap),
365         .id33(ID_TrapCond),
366         .id34(ID_EX_CanErr),
367         .id35(ID_M_CanErr),
368         // Data Signals
369         .id36(ID_ReadData1),
370         .id37(ID_ReadData2),
371         .id38(ID_SignExtImm),
372         // ----------------
```

```
373         .ex1(EX_Link2),
374         .ex2(EX_LinkRegDst2),
375         .ex3(EX_ALUSrcImm2),
376         .ex4(EX_ALUOp2),
377         .ex5(EX_Movn2),
378         .ex6(EX_Movz2),
379         .ex7(EX_LLSC2),
380         .ex8(EX_MemRead2),
381         .ex9(EX_MemWrite2),
382         .ex10(EX_MemByte2),
383         .ex11(EX_MemHalf2),
384         .ex12(EX_MemSignExtend2),
385         .ex13(EX_Left2),
386         .ex14(EX_Right2),
387         .ex15(EX_RegWrite2),
388         .ex16(EX_MemtoReg2),
389         .ex17(EX_ReverseEndian2),
390         .ex18(EX_Rs2),
391         .ex19(EX_Rt2),
392         .ex20(EX_WantRsByEX2),
393         .ex21(EX_NeedRsByEX2),
394         .ex22(EX_WantRtByEX2),
395         .ex23(EX_NeedRtByEX2),
396         .ex24(EX_KernelMode2),
397         .ex25(EX_RestartPC2),
398         .ex26(EX_IsBDS2),
399         .ex27(EX_Trap2),
400         .ex28(EX_TrapCond2),
401         .ex29(EX_EX_CanErr2),
402         .ex30(EX_M_CanErr2),
403         .ex31(EX_ReadData12),
404         .ex32(EX_ReadData22),
405         .ex33(EX_SignExtImm2),
406         .ex34(EX_Rd2),
407         .ex35(EX_Shamt2)
408         );
409
410
411
412         wire    EX_Link2;
413         wire    [ 1: 0]EX_LinkRegDst2;
414         wire    EX_ALUSrcImm2;
415         wire    [ 4: 0]EX_ALUOp2;
416         wire    EX_Movn2;
417         wire    EX_Movz2;
418         wire    EX_LLSC2;
419         wire    EX_MemRead2;
420         wire    EX_MemWrite2;
421         wire    EX_MemByte2;
422         wire    EX_MemHalf2;
423         wire    EX_MemSignExtend2;
424         wire    EX_Left2;
425         wire    EX_Right2;
426         wire    EX_RegWrite2;
427         wire    EX_MemtoReg2;
428         wire    EX_ReverseEndian2;
429         wire    [ 4: 0]EX_Rs2;
430         wire    [ 4: 0]EX_Rt2;
431         wire    EX_WantRsByEX2;
432         wire    EX_NeedRsByEX2;
433         wire    EX_WantRtByEX2;
434         wire    EX_NeedRtByEX2;
435         wire    EX_KernelMode2;
436         wire    [ 31: 0]EX_RestartPC2;
437         wire    EX_IsBDS2;
438         wire    EX_Trap2;
439         wire    EX_TrapCond2;
440         wire    EX_EX_CanErr2;
441         wire    EX_M_CanErr2;
442         wire    [ 31: 0]EX_ReadData12;
443         wire    [ 31: 0]EX_ReadData22;
444         wire    [ 31: 0]EX_SignExtImm2;
445         wire    [ 4: 0]EX_Rd2;
446         wire    [ 4: 0]EX_Shamt2;
447
448
449 sCPU_IDEX_StageIDEXtrei(
450         .id1(clock&sCPU3),
451         .id2(reset),
452         .id3(ID_Flush),
453         .id4(ID_Stall),
454         .id5(EX_Stall),
455         // Control Signals
456         .id6(ID_Link),
457         .id7(ID_RegDst),
458         .id8(ID_ALUSrcImm),
459         .id9(ID_ALUOp),
460         .id10(ID_Movn),
461         .id11(ID_Movz),
462         .id12(ID_LLSC),
463         .id13(ID_MemRead),
464         .id14(ID_MemWrite),
465         .id15(ID_MemByte),
466         .id16(ID_MemHalf),
467         .id17(ID_MemSignExtend),
468         .id18(ID_Left),
469         .id19(ID_Right),
470         .id20(ID_RegWrite),
471         .id21(ID_MemtoReg),
472         .id22(ID_ReverseEndian),
473         // Hazard & Forwarding
474         .id23(ID_Rs),
475         .id24(ID_Rt),
476         .id25(ID_WantRsByEX),
477         .id26(ID_NeedRsByEX),
478         .id27(ID_WantRtByEX),
479         .id28(ID_NeedRtByEX),
480         // Exception Control/Info
481         .id29(ID_KernelMode),
482         .id30(ID_RestartPC),
483         .id31(ID_IsBDS),
484         .id32(ID_Trap),
485         .id33(ID_TrapCond),
486         .id34(ID_EX_CanErr),
487         .id35(ID_M_CanErr),
488         // Data Signals
489         .id36(ID_ReadData1),
490         .id37(ID_ReadData2),
491         .id38(ID_SignExtImm),
492         // ----------------
493         .ex1(EX_Link3),
494         .ex2(EX_LinkRegDst3),
495         .ex3(EX_ALUSrcImm3),
496         .ex4(EX_ALUOp3),
497         .ex5(EX_Movn3),
498         .ex6(EX_Movz3),
499         .ex7(EX_LLSC3),
500         .ex8(EX_MemRead3),
501         .ex9(EX_MemWrite3),
502         .ex10(EX_MemByte3),
503         .ex11(EX_MemHalf3),
504         .ex12(EX_MemSignExtend3),
505         .ex13(EX_Left3),
506         .ex14(EX_Right3),
507         .ex15(EX_RegWrite3),
508         .ex16(EX_MemtoReg3),
509         .ex17(EX_ReverseEndian3),
510         .ex18(EX_Rs3),
511         .ex19(EX_Rt3),
512         .ex20(EX_WantRsByEX3),
513         .ex21(EX_NeedRsByEX3),
514         .ex22(EX_WantRtByEX3),
515         .ex23(EX_NeedRtByEX3),
516         .ex24(EX_KernelMode3),
517         .ex25(EX_RestartPC3),
518         .ex26(EX_IsBDS3),
519         .ex27(EX_Trap3),
520         .ex28(EX_TrapCond3),
521         .ex29(EX_EX_CanErr3),
522         .ex30(EX_M_CanErr3),
523         .ex31(EX_ReadData13),
524         .ex32(EX_ReadData23),
525         .ex33(EX_SignExtImm3),
526         .ex34(EX_Rd3),
527         .ex35(EX_Shamt3)
528         );
529
530
531         wire    EX_Link3;
532         wire    [ 1: 0]EX_LinkRegDst3;
533         wire    EX_ALUSrcImm3;
534         wire    [ 4: 0]EX_ALUOp3;
535         wire    EX_Movn3;
536         wire    EX_Movz3;
537         wire    EX_LLSC3;
538         wire    EX_MemRead3;
539         wire    EX_MemWrite3;
540         wire    EX_MemByte3;
541         wire    EX_MemHalf3;
542         wire    EX_MemSignExtend3;
543         wire    EX_Left3;
544         wire    EX_Right3;
545         wire    EX_RegWrite3;
546         wire    EX_MemtoReg3;
547         wire    EX_ReverseEndian3;
548         wire    [ 4: 0]EX_Rs3;
549         wire    [ 4: 0]EX_Rt3;
550         wire    EX_WantRsByEX3;
551         wire    EX_NeedRsByEX3;
552         wire    EX_WantRtByEX3;
553         wire    EX_NeedRtByEX3;
554         wire    EX_KernelMode3;
555         wire    [ 31: 0]EX_RestartPC3;
556         wire    EX_IsBDS3;
557         wire    EX_Trap3;
558         wire    EX_TrapCond3;
```

```
559         wire    EX_EX_CanErr3;
560         wire    EX_M_CanErr3;
561         wire    [ 31: 0]EX_ReadData13;
562         wire    [ 31: 0]EX_ReadData23;
563         wire    [ 31: 0]EX_SignExtImm3;
564         wire    [ 4: 0]EX_Rd3;
565         wire    [ 4: 0]EX_Shamt3;
566
567     Mux4#(.WIDTH(    1))EX_Link_Mux(
568                 .sel(selMIPS),
569                 .in0(EX_Link0),
570                 .in1(EX_Link1),
571                 .in2(EX_Link2),
572                 .in3(EX_Link3),
573                 .out(EX_Link)
574                 );
575
576     Mux4#(.WIDTH(    2))EX_LinkRegDst_Mux(
577                 .sel(selMIPS),
578                 .in0(EX_LinkRegDst0),
579                 .in1(EX_LinkRegDst1),
580                 .in2(EX_LinkRegDst2),
581                 .in3(EX_LinkRegDst3),
582                 .out(EX_LinkRegDst)
583                 );
584
585     Mux4#(.WIDTH(    1))EX_ALUSrcImm_Mux(
586                 .sel(selMIPS),
587                 .in0(EX_ALUSrcImm0),
588                 .in1(EX_ALUSrcImm1),
589                 .in2(EX_ALUSrcImm2),
590                 .in3(EX_ALUSrcImm3),
591                 .out(EX_ALUSrcImm)
592                 );
593
594     Mux4#(.WIDTH(    5))EX_ALUOp_Mux(
595                 .sel(selMIPS),
596                 .in0(EX_ALUOp0),
597                 .in1(EX_ALUOp1),
598                 .in2(EX_ALUOp2),
599                 .in3(EX_ALUOp3),
600                 .out(EX_ALUOp)
601                 );
602
603     Mux4#(.WIDTH(    1))EX_Movn_Mux(
604                 .sel(selMIPS),
605                 .in0(EX_Movn0),
606                 .in1(EX_Movn1),
607                 .in2(EX_Movn2),
608                 .in3(EX_Movn3),
609                 .out(EX_Movn)
610                 );
611
612     Mux4#(.WIDTH(    1))EX_Movz_Mux(
613                 .sel(selMIPS),
614                 .in0(EX_Movz0),
615                 .in1(EX_Movz1),
616                 .in2(EX_Movz2),
617                 .in3(EX_Movz3),
618                 .out(EX_Movz)
619                 );
620
621     Mux4#(.WIDTH(    1))EX_LLSC_Mux(
622                 .sel(selMIPS),
623                 .in0(EX_LLSC0),
624                 .in1(EX_LLSC1),
625                 .in2(EX_LLSC2),
626                 .in3(EX_LLSC3),
627                 .out(EX_LLSC)
628                 );
629
630     Mux4#(.WIDTH(    1))EX_MemRead_Mux(
631                 .sel(selMIPS),
632                 .in0(EX_MemRead0),
633                 .in1(EX_MemRead1),
634                 .in2(EX_MemRead2),
635                 .in3(EX_MemRead3),
636                 .out(EX_MemRead)
637                 );
638
639     Mux4#(.WIDTH(    1))EX_MemWrite_Mux(
640                 .sel(selMIPS),
641                 .in0(EX_MemWrite0),
642                 .in1(EX_MemWrite1),
643                 .in2(EX_MemWrite2),
644                 .in3(EX_MemWrite3),
645                 .out(EX_MemWrite)
646                 );
647
648     Mux4#(.WIDTH(    1))EX_MemByte_Mux(
649                 .sel(selMIPS),
650                 .in0(EX_MemByte0),
651                 .in1(EX_MemByte1),
652                 .in2(EX_MemByte2),
653                 .in3(EX_MemByte3),
654                 .out(EX_MemByte)
655                 );
656
657     Mux4#(.WIDTH(    1))EX_MemHalf_Mux(
658                 .sel(selMIPS),
659                 .in0(EX_MemHalf0),
660                 .in1(EX_MemHalf1),
661                 .in2(EX_MemHalf2),
662                 .in3(EX_MemHalf3),
663                 .out(EX_MemHalf)
664                 );
665
666     Mux4#(.WIDTH(    1))EX_MemSignExtend_Mux(
667                 .sel(selMIPS),
668                 .in0(EX_MemSignExtend0),
669                 .in1(EX_MemSignExtend1),
670                 .in2(EX_MemSignExtend2),
671                 .in3(EX_MemSignExtend3),
672                 .out(EX_MemSignExtend)
673                 );
674
675     Mux4#(.WIDTH(    1))EX_Left_Mux(
676                 .sel(selMIPS),
677                 .in0(EX_Left0),
678                 .in1(EX_Left1),
679                 .in2(EX_Left2),
680                 .in3(EX_Left3),
681                 .out(EX_Left)
682                 );
683
684     Mux4#(.WIDTH(    1))EX_Right_Mux(
685                 .sel(selMIPS),
686                 .in0(EX_Right0),
687                 .in1(EX_Right1),
688                 .in2(EX_Right2),
689                 .in3(EX_Right3),
690                 .out(EX_Right)
691                 );
692
693     Mux4#(.WIDTH(    1))EX_RegWrite_Mux(
694                 .sel(selMIPS),
695                 .in0(EX_RegWrite0),
696                 .in1(EX_RegWrite1),
697                 .in2(EX_RegWrite2),
698                 .in3(EX_RegWrite3),
699                 .out(EX_RegWrite)
700                 );
701
702     Mux4#(.WIDTH(    1))EX_MemtoReg_Mux(
703                 .sel(selMIPS),
704                 .in0(EX_MemtoReg0),
705                 .in1(EX_MemtoReg1),
706                 .in2(EX_MemtoReg2),
707                 .in3(EX_MemtoReg3),
708                 .out(EX_MemtoReg)
709                 );
710
711     Mux4#(.WIDTH(    1))EX_ReverseEndian_Mux(
712                 .sel(selMIPS),
713                 .in0(EX_ReverseEndian0),
714                 .in1(EX_ReverseEndian1),
715                 .in2(EX_ReverseEndian2),
716                 .in3(EX_ReverseEndian3),
717                 .out(EX_ReverseEndian)
718                 );
719
720     Mux4#(.WIDTH(    5))EX_Rs_Mux(
721                 .sel(selMIPS),
722                 .in0(EX_Rs0),
723                 .in1(EX_Rs1),
724                 .in2(EX_Rs2),
725                 .in3(EX_Rs3),
726                 .out(EX_Rs)
727                 );
728
729     Mux4#(.WIDTH(    5))EX_Rt_Mux(
730                 .sel(selMIPS),
731                 .in0(EX_Rt0),
732                 .in1(EX_Rt1),
733                 .in2(EX_Rt2),
734                 .in3(EX_Rt3),
735                 .out(EX_Rt)
736                 );
737
738     Mux4#(.WIDTH(    1))EX_WantRsByEX_Mux(
739                 .sel(selMIPS),
740                 .in0(EX_WantRsByEX0),
741                 .in1(EX_WantRsByEX1),
742                 .in2(EX_WantRsByEX2),
743                 .in3(EX_WantRsByEX3),
744                 .out(EX_WantRsByEX)
```

```
745             );
746
747     Mux4#(.WIDTH(    1))EX_NeedRsByEX_Mux
748             .sel(selMIPS),
749             .in0(EX_NeedRsByEX0),
750             .in1(EX_NeedRsByEX1),
751             .in2(EX_NeedRsByEX2),
752             .in3(EX_NeedRsByEX3),
753             .out(EX_NeedRsByEX)
754             );
755
756     Mux4#(.WIDTH(    1))EX_WantRtByEX_Mux(
757             .sel(selMIPS),
758             .in0(EX_WantRtByEX0),
759             .in1(EX_WantRtByEX1),
760             .in2(EX_WantRtByEX2),
761             .in3(EX_WantRtByEX3),
762             .out(EX_WantRtByEX)
763             );
764
765     Mux4#(.WIDTH(    1))EX_NeedRtByEX_Mux(
766             .sel(selMIPS),
767             .in0(EX_NeedRtByEX0),
768             .in1(EX_NeedRtByEX1),
769             .in2(EX_NeedRtByEX2),
770             .in3(EX_NeedRtByEX3),
771             .out(EX_NeedRtByEX)
772             );
773
774     Mux4#(.WIDTH(    1))EX_KernelMode_Mux(
775             .sel(selMIPS),
776             .in0(EX_KernelMode0),
777             .in1(EX_KernelMode1),
778             .in2(EX_KernelMode2),
779             .in3(EX_KernelMode3),
780             .out(EX_KernelMode)
781             );
782
783     Mux4#(.WIDTH(    32))EX_RestartPC_Mux(
784             .sel(selMIPS),
785             .in0(EX_RestartPC0),
786             .in1(EX_RestartPC1),
787             .in2(EX_RestartPC2),
788             .in3(EX_RestartPC3),
789             .out(EX_RestartPC)
790             );
791
792     Mux4#(.WIDTH(    1))EX_IsBDS_Mux(
793             .sel(selMIPS),
794             .in0(EX_IsBDS0),
795             .in1(EX_IsBDS1),
796             .in2(EX_IsBDS2),
797             .in3(EX_IsBDS3),
798             .out(EX_IsBDS)
799             );
800
801     Mux4#(.WIDTH(    1))EX_Trap_Mux(
802             .sel(selMIPS),
803             .in0(EX_Trap0),
804             .in1(EX_Trap1),
805             .in2(EX_Trap2),
806             .in3(EX_Trap3),
807             .out(EX_Trap)
808             );
809
810     Mux4#(.WIDTH(    1))EX_TrapCond_Mux(
811             .sel(selMIPS),
812             .in0(EX_TrapCond0),
813             .in1(EX_TrapCond1),
814             .in2(EX_TrapCond2),
815             .in3(EX_TrapCond3),
816             .out(EX_TrapCond)
817             );
818
819     Mux4#(.WIDTH(    1))EX_EX_CanErr_Mux(
820             .sel(selMIPS),
821             .in0(EX_EX_CanErr0),
822             .in1(EX_EX_CanErr1),
823             .in2(EX_EX_CanErr2),
824             .in3(EX_EX_CanErr3),
825             .out(EX_EX_CanErr)
826             );
827
828     Mux4#(.WIDTH(    1))EX_M_CanErr_Mux(
829             .sel(selMIPS),
830             .in0(EX_M_CanErr0),
831             .in1(EX_M_CanErr1),
832             .in2(EX_M_CanErr2),
833             .in3(EX_M_CanErr3),
834             .out(EX_M_CanErr)
835             );
836
837     Mux4#(.WIDTH(    32))EX_ReadData1_Mux(
838             .sel(selMIPS),
839             .in0(EX_ReadData10),
840             .in1(EX_ReadData11),
841             .in2(EX_ReadData12),
842             .in3(EX_ReadData13),
843             .out(EX_ReadData1)
844             );
845
846     Mux4#(.WIDTH(    32))EX_ReadData2_Mux(
847             .sel(selMIPS),
848             .in0(EX_ReadData20),
849             .in1(EX_ReadData21),
850             .in2(EX_ReadData22),
851             .in3(EX_ReadData23),
852             .out(EX_ReadData2)
853             );
854
855     Mux4#(.WIDTH(    32))EX_SignExtImm_Mux(
856             .sel(selMIPS),
857             .in0(EX_SignExtImm0),
858             .in1(EX_SignExtImm1),
859             .in2(EX_SignExtImm2),
860             .in3(EX_SignExtImm3),
861             .out(EX_SignExtImm)
862             );
863
864     Mux4#(.WIDTH(    5))EX_Rd_Mux(
865             .sel(selMIPS),
866             .in0(EX_Rd0),
867             .in1(EX_Rd1),
868             .in2(EX_Rd2),
869             .in3(EX_Rd3),
870             .out(EX_Rd)
871             );
872
873     Mux4#(.WIDTH(    5))EX_Shamt_Mux(
874             .sel(selMIPS),
875             .in0(EX_Shamt0),
876             .in1(EX_Shamt1),
877             .in2(EX_Shamt2),
878             .in3(EX_Shamt3),
879             .out(EX_Shamt)
880             );
881
882
883         // ---------------
884
885
886
887
888         endmodule
889
```

Figură 2-13 Codul verilog pentru IDEX4

IDEX4_Stage-Flow Summary

Flow Summary report for IDEX4_Stage
Tue Sep 22 21:43:29 2015
Quartus II 64-Bit Version 15.0.0 Build 145 04/22/2015 SJ Web Edition

Flow Summary	
Flow Status	Successful - Sat Sep 19 11:26:48 2015
Quartus II 64-Bit Version	15.0.0 Build 145 04/22/2015 SJ Web Edition
Revision Name	IDEX4_Stage
Top-level Entity Name	IDEX4_Stage
Family	Cyclone V
Device	5CGXBC7D7F31C8
Timing Models	Final
Logic utilization (in ALMs)	285 / 56,480 (< 1 %)
Total registers	616
Total pins	345 / 522 (66 %)
Total virtual pins	0
Total block memory bits	0 / 7,024,640 (0 %)
Total DSP Blocks	0 / 156 (0 %)
Total HSSI RX PCSs	0 / 9 (0 %)
Total HSSI PMA RX Deserializers	0 / 9 (0 %)
Total HSSI TX PCSs	0 / 9 (0 %)
Total HSSI PMA TX Serializers	0 / 9 (0 %)
Total PLLs	0 / 16 (0 %)
Total DLLs	0 / 4 (0 %)

Multiplexer Restructuring Statistics (Restructuring Performed) report for IDEX4_Stage
Tue Sep 22 21:52:17 2015
Quartus II 64-Bit Version 15.0.0 Build 145 04/22/2015 SJ Web Edition

Multiplexer Restructuring Statistics (Restructuring Performed)

Multiplexer Inputs	Bus Width	Baseline Area	Area if Restructured	Saving if Restructured	Registered	Example Multiplexer Output
3:1	616 bits	1232 LEs	0 LEs	1232 LEs	Yes	|IDEX4_Stage|sCPU_IDEX_Stage:IDEXdoi|ex7
4:1	153 bits	306 LEs	306 LEs	0 LEs	No	|IDEX4_Stage|Mux4:EX_MemSignExtend_Mux|Mux0

IDEX4_Stage-Post-Synthesis Netlist Statistics for Top Partition

Post-Synthesis Netlist Statistics for Top Partition report for IDEX4_Stage
Tue Sep 22 21:47:32 2015
Quartus II 64-Bit Version 15.0.0 Build 145 04/22/2015 SJ Web Edition

Post-Synthesis Netlist Statistics for Top Partition	
Type	Count
arriav_ff	616
ENA SCLR	616
arriav_lcell_comb	177
extend	1
7 data inputs	1
normal	176
2 data inputs	8
3 data inputs	15
6 data inputs	153
boundary_port	345
Max LUT depth	2.00
Average LUT depth	0.87

IDEX4_Stage-Resource Usage Summary

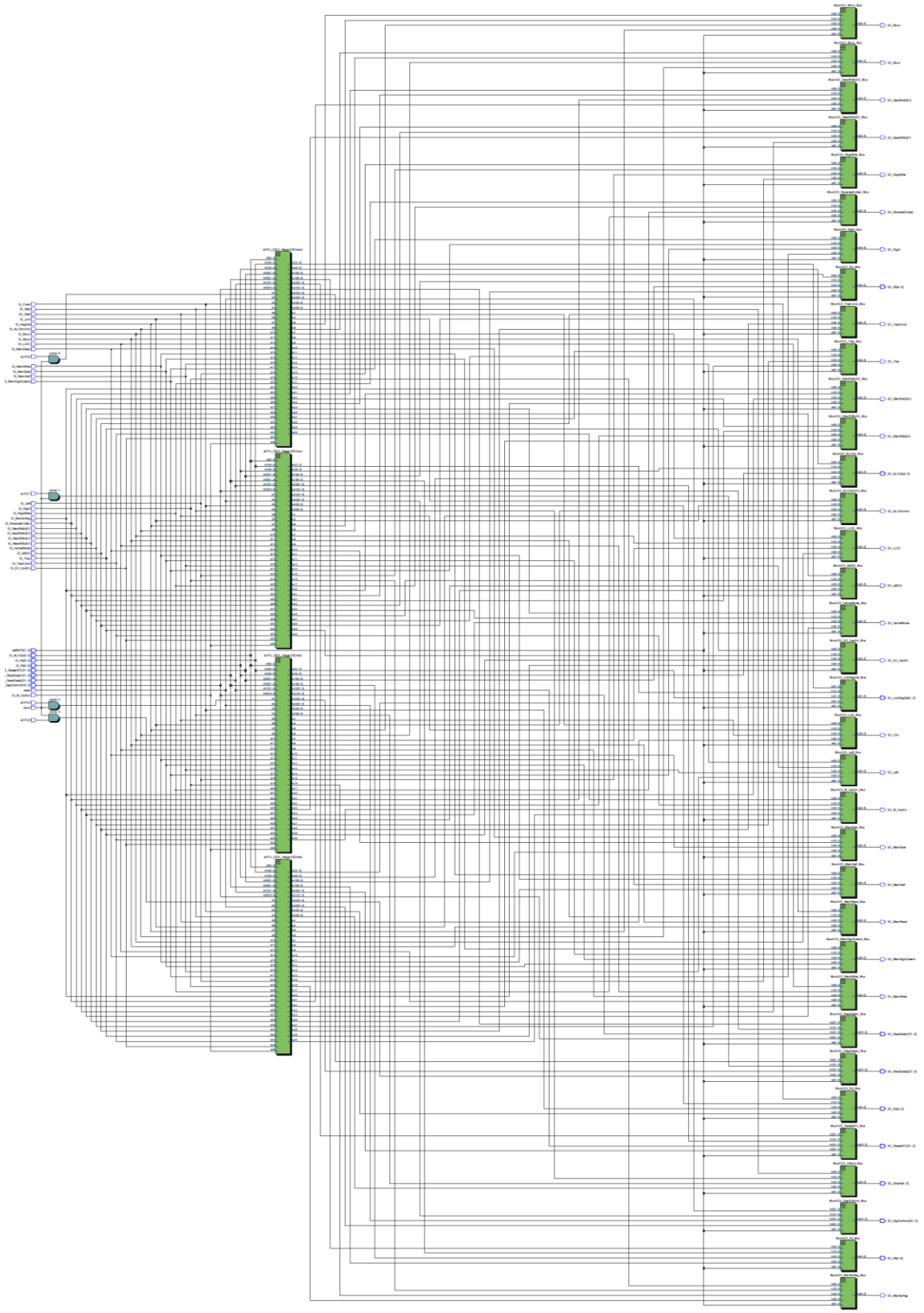
Resource Usage Summary report for IDEX4_Stage
Tue Sep 22 21:49:15 2015
Quartus II 64-Bit Version 15.0.0 Build 145 04/22/2015 SJ Web Edition

Fitter Resource Usage Summary

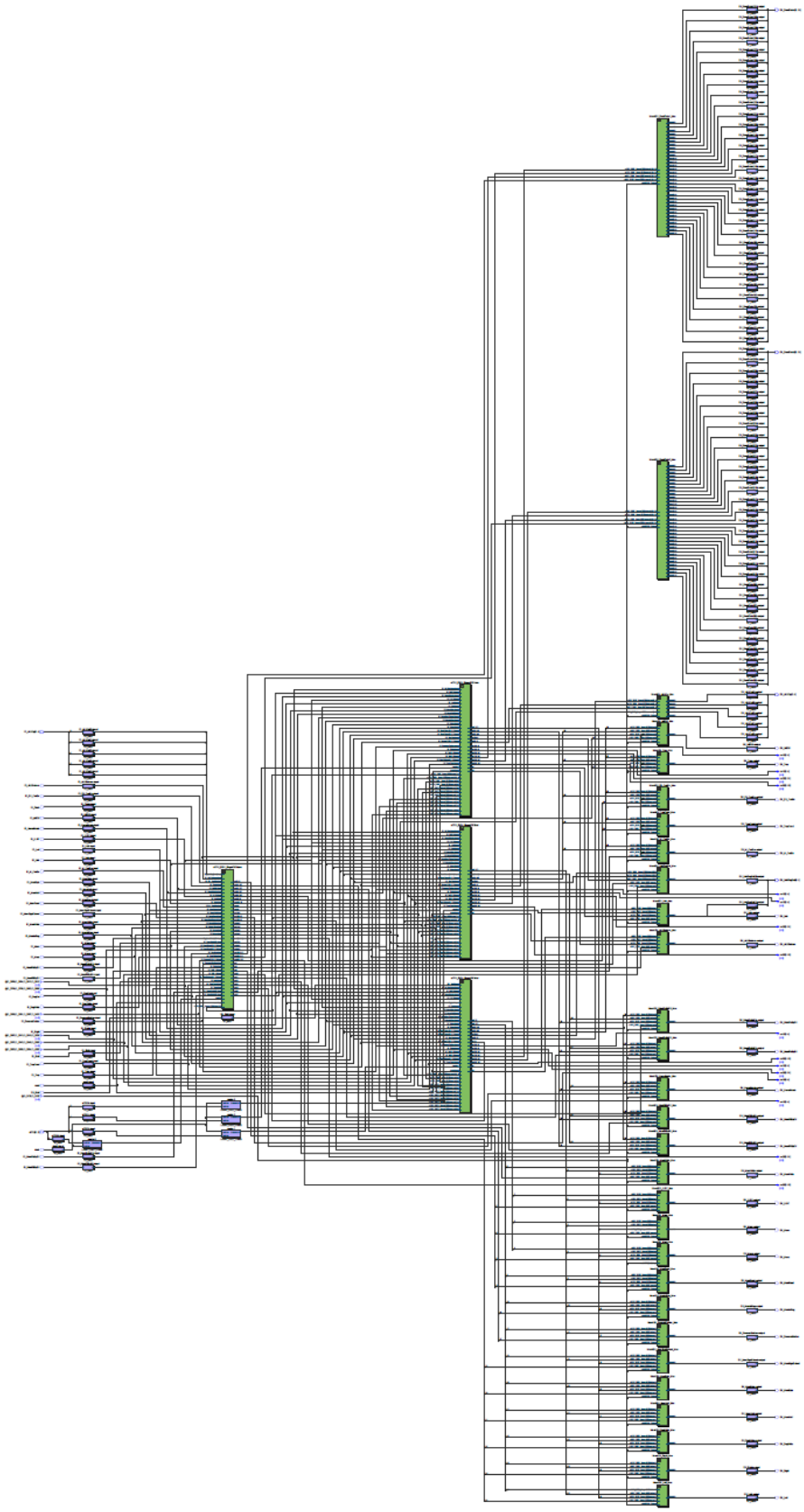
Resource	Usage	%
Logic utilization (ALMs needed / total ALMs on device)	285 / 56,480	< 1 %
ALMs needed [=A-B+C]	285	
[A] ALMs used in final placement [=a+b+c+d]	386 / 56,480	< 1 %
[a] ALMs used for LUT logic and registers	81	
[b] ALMs used for LUT logic	79	
[c] ALMs used for registers	226	
[d] ALMs used for memory (up to half of total ALMs)	0	
[B] Estimate of ALMs recoverable by dense packing	114 / 56,480	< 1 %
[C] Estimate of ALMs unavailable [=a+b+c+d]	13 / 56,480	< 1 %
[a] Due to location constrained logic	0	
[b] Due to LAB-wide signal conflicts	1	
[c] Due to LAB input limits	12	
[d] Due to virtual I/Os	0	
Difficulty packing design	Low	
Total LABs: partially or completely used	102 / 5,648	2 %
-- Logic LABs	102	
-- Memory LABs (up to half of total LABs)	0	
Combinational ALUT usage for logic	178	
-- 7 input functions	1	
-- 6 input functions	153	
-- 5 input functions	0	
-- 4 input functions	0	
-- <=3 input functions	24	
Combinational ALUT usage for route-throughs	260	
Dedicated logic registers	616	
-- By type:		
-- Primary logic registers	613 / 112,960	< 1 %
-- Secondary logic registers	3 / 112,960	< 1 %
-- By function:		
-- Design implementation registers	616	
-- Routing optimization registers	0	
Virtual pins	0	
I/O pins	345 / 522	66 %
-- Clock pins	10 / 15	67 %
-- Dedicated input pins	0 / 29	0 %
Global signals	0	
M10K blocks	0 / 686	0 %
Total MLAB memory bits	0	
Total block memory bits	0 / 7,024,640	0 %
Total block memory implementation bits	0 / 7,024,640	0 %
Total DSP Blocks	0 / 156	0 %
Fractional PLLs	0 / 7	0 %
Global clocks	0 / 16	0 %
Quadrant clocks	0 / 88	0 %
Horizontal periphery clocks	0 / 18	0 %
SERDES Transmitters	0 / 120	0 %
SERDES Receivers	0 / 120	0 %
JTAGs	0 / 1	0 %
ASMI blocks	0 / 1	0 %
CRC blocks	0 / 1	0 %
Remote update blocks	0 / 1	0 %
Oscillator blocks	0 / 1	0 %
Standard RX PCSs	0 / 9	0 %
HSSI PMA RX Deserializers	0 / 9	0 %
Standard TX PCSs	0 / 9	0 %
HSSI PMA TX Serializers	0 / 9	0 %
Channel PLLs	0 / 9	0 %
Impedance control blocks	0 / 3	0 %
Average interconnect usage (total/H/V)	0.8% / 0.6% / 1.4%	
Peak interconnect usage (total/H/V)	10.3% / 10.7% / 9.2%	
Maximum fan-out	617	
Highest non-global fan-out	617	
Total fan-out	4235	
Average fan-out	2.43	

	Multiplexer Inputs	Bus Width	Baseline Area	Area if Restructured	Saving if Restructured	Registered	Example Multiplexer Output
1	3:1	616 bits	1232 LEs	0 LEs	1232 LEs	Yes	|IDEX4_Stage| sCPU_IDEX_Stage:IDEXdoi|ex7
2	4:1	153 bits	306 LEs	306 LEs	0 LEs	No	|IDEX4_Stage| Mux4:EX_MemSignExtend_Mux| Mux0

	Resource	Usage
1	Estimate of Logic utilization (ALMs needed)	387
2		
3	Combinational ALUT usage for logic	177
1	-- 7 input functions	1
2	-- 6 input functions	153
3	-- 5 input functions	0
4	-- 4 input functions	0
5	-- <=3 input functions	23
4		
5	Dedicated logic registers	616
6		
7	I/O pins	345
8		
9	Total DSP Blocks	0
10		
11	Maximum fan-out node	reset~input
12	Maximum fan-out	617
13	Total fan-out	3975
14	Average fan-out	2.68

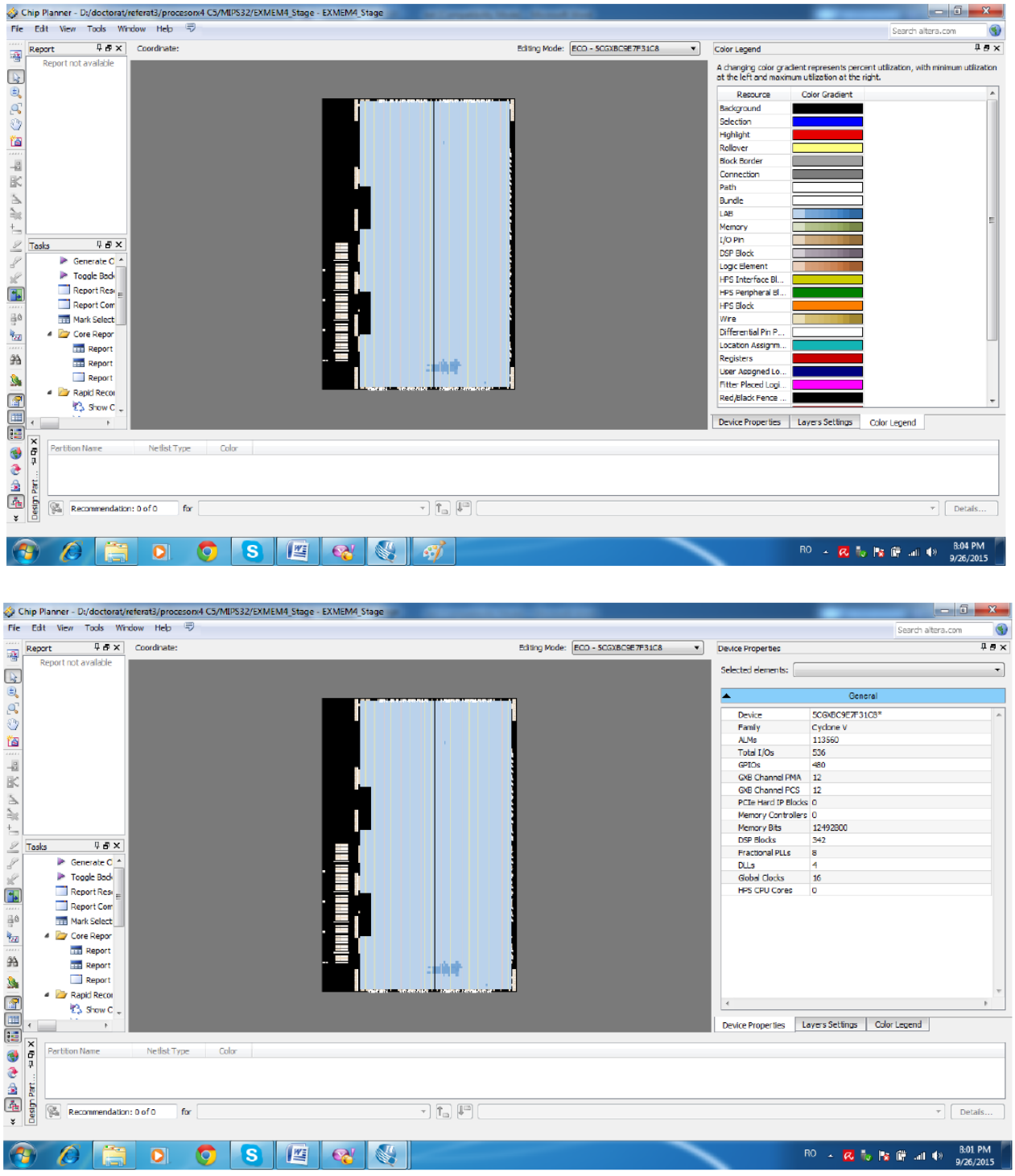


Figură 2-14 RTL IDEX4



Figură 2-15 Tehnology map IDEX4

2.2.3. EXMEM4



Figură 2-16 Chip Planner EXMEM4

```
module   EXMEM4_Stage(
        input       sCPU0,
        input       sCPU1,
        input       sCPU2,
        input       sCPU3,
        input       [ 1: 0]selMIPS,
        input       clock,
        input       reset,
        input       EX_Flush,
        input       EX_Stall,
        input       M_Stall,
        // Control Signals
        input       EX_Movn,
        input       EX_Movz,
        input       EX_BZero,
        input       EX_RegWrite,      // Future Co
        input       EX_MemtoReg,      // Future Co
        input       EX_ReverseEndian,
        input       EX_LLSC,
        input       EX_MemRead,
        input       EX_MemWrite,
        input       EX_MemByte,
        input       EX_MemHalf,
        input       EX_MemSignExtend,
        input       EX_Left,
        input       EX_Right,
        // Exception Control/Info
        input       EX_KernelMode,
        input       [ 31: 0]EX_RestartPC,
        input       EX_IsBDS,
        input       EX_Trap,
        input       EX_TrapCond,
        input       EX_M_CanErr,
        // Data Signals
        input       [ 31: 0]EX_ALU_Result,
        input       [ 31: 0]EX_ReadData2,
        input       [ 4: 0]EX_RtRd,
        // ------------------
        output      M_RegWrite,
        output      M_MemtoReg,
        output      M_ReverseEndian,
        output      M_LLSC,
        output      M_MemRead,
        output      M_MemWrite,
        output      M_MemByte,
        output      M_MemHalf,
        output      M_MemSignExtend,
        output      M_Left,
        output      M_Right,
        output      M_KernelMode,
        output      [ 31: 0]M_RestartPC,
        output      M_IsBDS,
        output      M_Trap,
        output      M_TrapCond,
        output      M_M_CanErr,
        output      [ 31: 0]M_ALU_Result,
        output      [ 31: 0]M_ReadData2,
        output      [ 4: 0]M_RtRd
        );

sCPU_EXMEM_Stageexmem0(
        .ex1(clock&sCPU0),
        .ex2(reset),
        .ex3(EX_Flush),
        .ex4(EX_Stall),
        .ex5(M_Stall),
        // Control Signals
        .ex6(EX_Movn),
        .ex7(EX_Movz),
        .ex8(EX_BZero),
        .ex9(EX_RegWrite),
        .ex10(EX_MemtoReg),
        .ex11(EX_ReverseEndian),
        .ex12(EX_LLSC),
        .ex13(EX_MemRead),
        .ex14(EX_MemWrite),
        .ex15(EX_MemByte),
        .ex16(EX_MemHalf),
        .ex17(EX_MemSignExtend),
        .ex18(EX_Left),
        .ex19(EX_Right),
        // Exception Control/Info
        .ex20(EX_KernelMode),
        .ex21(EX_RestartPC),
        .ex22(EX_IsBDS),
        .ex23(EX_Trap),
        .ex24(EX_TrapCond),
        .ex25(EX_M_CanErr),
        // Data Signals
        .ex26(EX_ALU_Result),
        .ex27(EX_ReadData2),
        .ex28(EX_RtRd),
        // ------------------
        .mem1(M_RegWrite0),
        .mem2(M_MemtoReg0),
        .mem3(M_ReverseEndian0),
        .mem4(M_LLSC0),
        .mem5(M_MemRead0),
        .mem6(M_MemWrite0),
        .mem7(M_MemByte0),
        .mem8(M_MemHalf0),
        .mem9(M_MemSignExtend0),
        .mem10(M_Left0),
        .mem11(M_Right0),
        .mem12(M_KernelMode0),
        .mem13(M_RestartPC0),
        .mem14(M_IsBDS0),
        .mem15(M_Trap0),
        .mem16(M_TrapCond0),
        .mem17(M_M_CanErr0),
        .mem18(M_ALU_Result0),
        .mem19(M_ReadData20),
        .mem20(M_RtRd0)
        );

        wire    M_RegWrite0;
        wire    M_MemtoReg0;
        wire    M_ReverseEndian0;
        wire    M_LLSC0;
        wire    M_MemRead0;
        wire    M_MemWrite0;
        wire    M_MemByte0;
        wire    M_MemHalf0;
        wire    M_MemSignExtend0;
        wire    M_Left0;
        wire    M_Right0;
        wire    M_KernelMode0;
        wire    [ 31: 0]M_RestartPC0;
        wire    M_IsBDS0;
        wire    M_Trap0;
        wire    M_TrapCond0;
        wire    M_M_CanErr0;
        wire    [ 31: 0]M_ALU_Result0
        wire    [ 31: 0]M_ReadData20;
        wire    [ 4: 0]M_RtRd0;


sCPU_EXMEM_Stageexmem1(
        .ex1(clock&sCPU1),
        .ex2(reset),
        .ex3(EX_Flush),
        .ex4(EX_Stall),
        .ex5(M_Stall),
        // Control Signals
        .ex6(EX_Movn),
        .ex7(EX_Movz),
        .ex8(EX_BZero),
        .ex9(EX_RegWrite),
        .ex10(EX_MemtoReg),
        .ex11(EX_ReverseEndian),
        .ex12(EX_LLSC),
        .ex13(EX_MemRead),
        .ex14(EX_MemWrite),
        .ex15(EX_MemByte),
        .ex16(EX_MemHalf),
        .ex17(EX_MemSignExtend),
        .ex18(EX_Left),
        .ex19(EX_Right),
        // Exception Control/Info
        .ex20(EX_KernelMode),
        .ex21(EX_RestartPC),
        .ex22(EX_IsBDS),
        .ex23(EX_Trap),
        .ex24(EX_TrapCond),
        .ex25(EX_M_CanErr),
        // Data Signals
        .ex26(EX_ALU_Result),
        .ex27(EX_ReadData2),
        .ex28(EX_RtRd),
        // ------------------
        .mem1(M_RegWrite1),
        .mem2(M_MemtoReg1),
        .mem3(M_ReverseEndian1),
        .mem4(M_LLSC1),
        .mem5(M_MemRead1),
        .mem6(M_MemWrite1),
        .mem7(M_MemByte1),
        .mem8(M_MemHalf1),
        .mem9(M_MemSignExtend1),
        .mem10(M_Left1),
        .mem11(M_Right1),
        .mem12(M_KernelMode1),
        .mem13(M_RestartPC1),
        .mem14(M_IsBDS1),
        .mem15(M_Trap1),
        .mem16(M_TrapCond1),
```

```
187         .mem17(M_M_CanErr1),
188         .mem18(M_ALU_Result1),
189         .mem19(M_ReadData21),
190         .mem20(M_RtRd1)
191         );
192
193
194         wire    M_RegWrite1;
195         wire    M_MemtoReg1;
196         wire    M_ReverseEndian1;
197         wire    M_LLSC1;
198         wire    M_MemRead1;
199         wire    M_MemWrite1;
200         wire    M_MemByte1;
201         wire    M_MemHalf1;
202         wire    M_MemSignExtend1;
203         wire    M_Left1;
204         wire    M_Right1;
205         wire    M_KernelMode1;
206         wire    [ 31: 0]M_RestartPC1;
207         wire    M_IsBDS1;
208         wire    M_Trap1;
209         wire    M_TrapCond1;
210         wire    M_M_CanErr1;
211         wire    [ 31: 0]M_ALU_Result1;
212         wire    [ 31: 0]M_ReadData21;
213         wire    [ 4: 0]M_RtRd1;
214
215 sCPU_EXMEM_Stageexmem2(
216         .ex1(clock&sCPU2),
217         .ex2(reset),
218         .ex3(EX_Flush),
219         .ex4(EX_Stall),
220         .ex5(M_Stall),
221         // Control Signals
222         .ex6(EX_Movn),
223         .ex7(EX_Movz),
224         .ex8(EX_BZero),
225         .ex9(EX_RegWrite),          //
226         .ex10(EX_MemtoReg),         //
227         .ex11(EX_ReverseEndian),
228         .ex12(EX_LLSC),
229         .ex13(EX_MemRead),
230         .ex14(EX_MemWrite),
231         .ex15(EX_MemByte),
232         .ex16(EX_MemHalf),
233         .ex17(EX_MemSignExtend),
234         .ex18(EX_Left),
235         .ex19(EX_Right),
236         // Exception Control/Info
237         .ex20(EX_KernelMode),
238         .ex21(EX_RestartPC),
239         .ex22(EX_IsBDS),
240         .ex23(EX_Trap),
241         .ex24(EX_TrapCond),
242         .ex25(EX_M_CanErr),
243         // Data Signals
244         .ex26(EX_ALU_Result),
245         .ex27(EX_ReadData2),
246         .ex28(EX_RtRd),
247         // ----------------
248         .mem1(M_RegWrite2),
249         .mem2(M_MemtoReg2),
250         .mem3(M_ReverseEndian2),
251         .mem4(M_LLSC2),
252         .mem5(M_MemRead2),
253         .mem6(M_MemWrite2),
254         .mem7(M_MemByte2),
255         .mem8(M_MemHalf2),
256         .mem9(M_MemSignExtend2),
257         .mem10(M_Left2),
258         .mem11(M_Right2),
259         .mem12(M_KernelMode2),
260         .mem13(M_RestartPC2),
261         .mem14(M_IsBDS2),
262         .mem15(M_Trap2),
263         .mem16(M_TrapCond2),
264         .mem17(M_M_CanErr2),
265         .mem18(M_ALU_Result2),
266         .mem19(M_ReadData22),
267         .mem20(M_RtRd2)
268         );
269
270
271         wire    M_RegWrite2;
272         wire    M_MemtoReg2;
273         wire    M_ReverseEndian2;
274         wire    M_LLSC2;
275         wire    M_MemRead2;
276         wire    M_MemWrite2;
277         wire    M_MemByte2;
278         wire    M_MemHalf2;
279         wire    M_MemSignExtend2;
280         wire    M_Left2;
281         wire    M_Right2;
282         wire    M_KernelMode2;
283         wire    [ 31: 0]M_RestartPC2;
284         wire    M_IsBDS2;
285         wire    M_Trap2;
286         wire    M_TrapCond2;
287         wire    M_M_CanErr2;
288         wire    [ 31: 0]M_ALU_Result2;
289         wire    [ 31: 0]M_ReadData22;
290         wire    [ 4: 0]M_RtRd2;
291
292 sCPU_EXMEM_Stageexmem3(
293         .ex1(clock&sCPU3),
294         .ex2(reset),
295         .ex3(EX_Flush),
296         .ex4(EX_Stall),
297         .ex5(M_Stall),
298         // Control Signals
299         .ex6(EX_Movn),
300         .ex7(EX_Movz),
301         .ex8(EX_BZero),
302         .ex9(EX_RegWrite),          //
303         .ex10(EX_MemtoReg),         //
304         .ex11(EX_ReverseEndian),
305         .ex12(EX_LLSC),
306         .ex13(EX_MemRead),
307         .ex14(EX_MemWrite),
308         .ex15(EX_MemByte),
309         .ex16(EX_MemHalf),
310         .ex17(EX_MemSignExtend),
311         .ex18(EX_Left),
312         .ex19(EX_Right),
313         // Exception Control/Info
314         .ex20(EX_KernelMode),
315         .ex21(EX_RestartPC),
316         .ex22(EX_IsBDS),
317         .ex23(EX_Trap),
318         .ex24(EX_TrapCond),
319         .ex25(EX_M_CanErr),
320         // Data Signals
321         .ex26(EX_ALU_Result),
322         .ex27(EX_ReadData2),
323         .ex28(EX_RtRd),
324         // ----------------
325         .mem1(M_RegWrite3),
326         .mem2(M_MemtoReg3),
327         .mem3(M_ReverseEndian3),
328         .mem4(M_LLSC3),
329         .mem5(M_MemRead3),
330         .mem6(M_MemWrite3),
331         .mem7(M_MemByte3),
332         .mem8(M_MemHalf3),
333         .mem9(M_MemSignExtend3),
334         .mem10(M_Left3),
335         .mem11(M_Right3),
336         .mem12(M_KernelMode3),
337         .mem13(M_RestartPC3),
338         .mem14(M_IsBDS3),
339         .mem15(M_Trap3),
340         .mem16(M_TrapCond3),
341         .mem17(M_M_CanErr3),
342         .mem18(M_ALU_Result3),
343         .mem19(M_ReadData23),
344         .mem20(M_RtRd3)
345         );
346
347
348
349         wire    M_RegWrite3;
350         wire    M_MemtoReg3;
351         wire    M_ReverseEndian3;
352         wire    M_LLSC3;
353         wire    M_MemRead3;
354         wire    M_MemWrite3;
355         wire    M_MemByte3;
356         wire    M_MemHalf3;
357         wire    M_MemSignExtend3;
358         wire    M_Left3;
359         wire    M_Right3;
360         wire    M_KernelMode3;
361         wire    [ 31: 0]M_RestartPC3;
362         wire    M_IsBDS3;
363         wire    M_Trap3;
364         wire    M_TrapCond3;
365         wire    M_M_CanErr3;
366         wire    [ 31: 0]M_ALU_Result3;
367         wire    [ 31: 0]M_ReadData23;
368         wire    [ 4: 0]M_RtRd3;
369
370
371 Mux4#(.WIDTH(    1))M_RegWrite_Mux(
372         .sel(selMIPS),
```

```
373             .in0(M_RegWrite0),
374             .in1(M_RegWrite1),
375             .in2(M_RegWrite2),
376             .in3(M_RegWrite3),
377             .out(M_RegWrite)
378             );
379
380     Mux4#(.WIDTH(    1))M_MemtoReg_Mux(
381             .sel(selMIPS),
382             .in0(M_MemtoReg0),
383             .in1(M_MemtoReg1),
384             .in2(M_MemtoReg2),
385             .in3(M_MemtoReg3),
386             .out(M_MemtoReg)
387             );
388
389     Mux4#(.WIDTH(    1))M_ReverseEndian_Mux(
390             .sel(selMIPS),
391             .in0(M_ReverseEndian0),
392             .in1(M_ReverseEndian1),
393             .in2(M_ReverseEndian2),
394             .in3(M_ReverseEndian3),
395             .out(M_ReverseEndian)
396             );
397
398     Mux4#(.WIDTH(    1))M_LLSC_Mux(
399             .sel(selMIPS),
400             .in0(M_LLSC0),
401             .in1(M_LLSC1),
402             .in2(M_LLSC2),
403             .in3(M_LLSC3),
404             .out(M_LLSC)
405             );
406
407     Mux4#(.WIDTH(    1))M_MemRead_Mux(
408             .sel(selMIPS),
409             .in0(M_MemRead0),
410             .in1(M_MemRead1),
411             .in2(M_MemRead2),
412             .in3(M_MemRead3),
413             .out(M_MemRead)
414             );
415
416
417     Mux4#(.WIDTH(    1))M_MemWrite_Mux(
418             .sel(selMIPS),
419             .in0(M_MemWrite0),
420             .in1(M_MemWrite1),
421             .in2(M_MemWrite2),
422             .in3(M_MemWrite3),
423             .out(M_MemWrite)
424             );
425
426     Mux4#(.WIDTH(    1))M_MemByte_Mux(
427             .sel(selMIPS),
428             .in0(M_MemByte0),
429             .in1(M_MemByte1),
430             .in2(M_MemByte2),
431             .in3(M_MemByte3),
432             .out(M_MemByte)
433             );
434
435     Mux4#(.WIDTH(    1))M_MemHalf_Mux(
436             .sel(selMIPS),
437             .in0(M_MemHalf0),
438             .in1(M_MemHalf1),
439             .in2(M_MemHalf2),
440             .in3(M_MemHalf3),
441             .out(M_MemHalf)
442             );
443
444     Mux4#(.WIDTH(    1))M_MemSignExtend_Mux(
445             .sel(selMIPS),
446             .in0(M_MemSignExtend0),
447             .in1(M_MemSignExtend1),
448             .in2(M_MemSignExtend2),
449             .in3(M_MemSignExtend3),
450             .out(M_MemSignExtend)
451             );
452
453
454     Mux4#(.WIDTH(    1))M_Left_Mux(
455             .sel(selMIPS),
456             .in0(M_Left0),
457             .in1(M_Left1),
458             .in2(M_Left2),
459             .in3(M_Left3),
460             .out(M_Left)
461             );
462
463     Mux4#(.WIDTH(    1))M_Right_Mux(
464             .sel(selMIPS),
465             .in0(M_Right0),
466             .in1(M_Right1),
467             .in2(M_Right2),
468             .in3(M_Right3),
469             .out(M_Right)
470             );
471     Mux4#(.WIDTH(    1))M_KernelMode_Mux(
472             .sel(selMIPS),
473             .in0(M_KernelMode0),
474             .in1(M_KernelMode1),
475             .in2(M_KernelMode2),
476             .in3(M_KernelMode3),
477             .out(M_KernelMode)
478             );
479
480     Mux4#(.WIDTH(    32))M_RestartPC_Mux(
481             .sel(selMIPS),
482             .in0(M_RestartPC0),
483             .in1(M_RestartPC1),
484             .in2(M_RestartPC2),
485             .in3(M_RestartPC3),
486             .out(M_RestartPC)
487             );
488
489
490     Mux4#(.WIDTH(    1))M_IsBDS_Mux(
491             .sel(selMIPS),
492             .in0(M_IsBDS0),
493             .in1(M_IsBDS1),
494             .in2(M_IsBDS2),
495             .in3(M_IsBDS3),
496             .out(M_IsBDS)
497             );
498
499
500     Mux4#(.WIDTH(    1))M_Trap_Mux(
501             .sel(selMIPS),
502             .in0(M_Trap0),
503             .in1(M_Trap1),
504             .in2(M_Trap2),
505             .in3(M_Trap3),
506             .out(M_Trap)
507             );
508
509
510     Mux4#(.WIDTH(    1))M_TrapCond_Mux(
511             .sel(selMIPS),
512             .in0(M_TrapCond0),
513             .in1(M_TrapCond1),
514             .in2(M_TrapCond2),
515             .in3(M_TrapCond3),
516             .out(M_TrapCond)
517             );
518
519     Mux4#(.WIDTH(    1))M_M_CanErr_Mux(
520             .sel(selMIPS),
521             .in0(M_M_CanErr0),
522             .in1(M_M_CanErr1),
523             .in2(M_M_CanErr2),
524             .in3(M_M_CanErr3),
525             .out(M_M_CanErr)
526             );
527
528
529
530     Mux4#(.WIDTH(    32))M_ALU_Result_Mux(
531             .sel(selMIPS),
532             .in0(M_ALU_Result0),
533             .in1(M_ALU_Result1),
534             .in2(M_ALU_Result2),
535             .in3(M_ALU_Result3),
536             .out(M_ALU_Result)
537             );
538
539
540     Mux4#(.WIDTH(    32))M_ReadData2_Mux(
541             .sel(selMIPS),
542             .in0(M_ReadData20),
543             .in1(M_ReadData21),
544             .in2(M_ReadData22),
545             .in3(M_ReadData23),
546             .out(M_ReadData2)
547             );
548
549     Mux4#(.WIDTH(    5))M_RtRd_Mux(
550             .sel(selMIPS),
551             .in0(M_RtRd0),
552             .in1(M_RtRd1),
553             .in2(M_RtRd2),
554             .in3(M_RtRd3),
555             .out(M_RtRd)
556             );
557
558     endmodule
```

Figură 2-17 Codul verilog pentru EXMEM4

EXMEM4_Stage-Flow Summary

Flow Summary report for EXMEM4_Stage
Tue Sep 22 21:26:14 2015
Quartus II 64-Bit Version 15.0.0 Build 145 04/22/2015 SJ Web Edition

Flow Summary	
Flow Status	Successful - Sat Sep 19 18:42:46 2015
Quartus II 64-Bit Version	15.0.0 Build 145 04/22/2015 SJ Web Edition
Revision Name	EXMEM4_Stage
Top-level Entity Name	EXMEM4_Stage
Family	Cyclone V
Device	5CGXBC9E7F31C8
Timing Models	Final
Logic utilization (in ALMs)	220 / 113,560 (< 1 %)
Total registers	468
Total pins	248 / 536 (46 %)
Total virtual pins	0
Total block memory bits	0 / 12,492,800 (0 %)
Total DSP Blocks	0 / 342 (0 %)
Total HSSI RX PCSs	0 / 12 (0 %)
Total HSSI PMA RX Deserializers	0 / 12 (0 %)
Total HSSI TX PCSs	0 / 12 (0 %)
Total HSSI PMA TX Serializers	0 / 12 (0 %)
Total PLLs	0 / 20 (0 %)
Total DLLs	0 / 4 (0 %)

Multiplexer Restructuring Statistics (Restructuring Performed) report for EXMEM4_Stage
Tue Sep 22 21:30:27 2015
Quartus II 64-Bit Version 15.0.0 Build 145 04/22/2015 SJ Web Edition

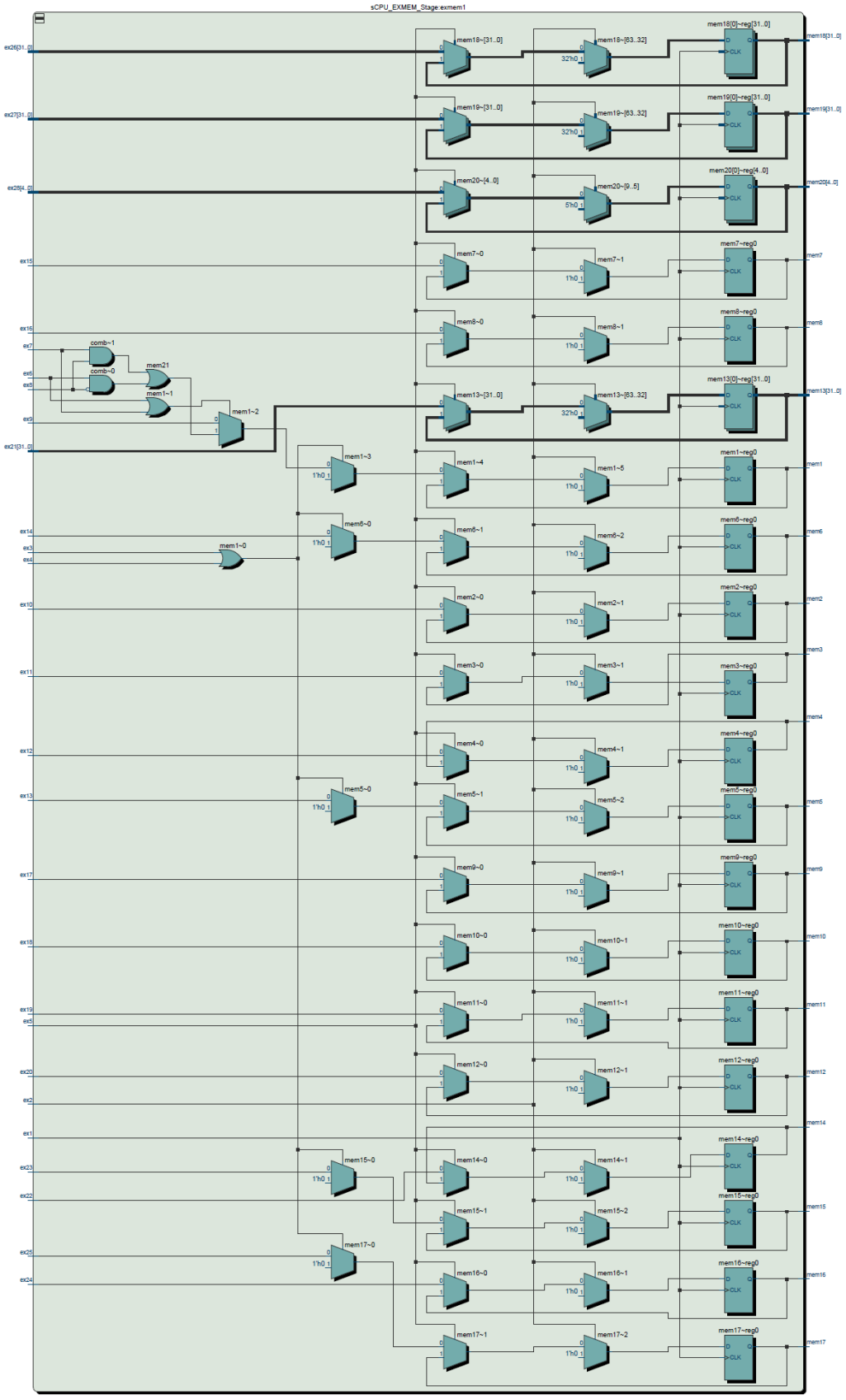
Multiplexer Restructuring Statistics (Restructuring Performed)

Multiplexer Inputs	Bus Width	Baseline Area	Area if Restructured	Saving if Restructured	Registered	Example Multiplexer Output
3:1	468 bits	936 LEs	0 LEs	936 LEs	Yes	|EXMEM4_Stage|sCPU_EXMEM_Stage:exmem1|mem8
4:1	117 bits	234 LEs	234 LEs	0 LEs	No	|EXMEM4_Stage|Mux4:M_MemWrite_Mux|Mux0

EXMEM4_Stage-Post-Synthesis Netlist Statistics for Top Partition

Post-Synthesis Netlist Statistics for Top Partition report for EXMEM4_Stage
Tue Sep 22 21:54:46 2015
Quartus II 64-Bit Version 15.0.0 Build 145 04/22/2015 SJ Web Edition

Post-Synthesis Netlist Statistics for Top Partition	
Type	Count
arriav_ff	468
ENA SCLR	468
arriav_lcell_comb	127
normal	127
2 data inputs	5
3 data inputs	4
6 data inputs	118
boundary_port	248
Max LUT depth	1.00
Average LUT depth	0.85

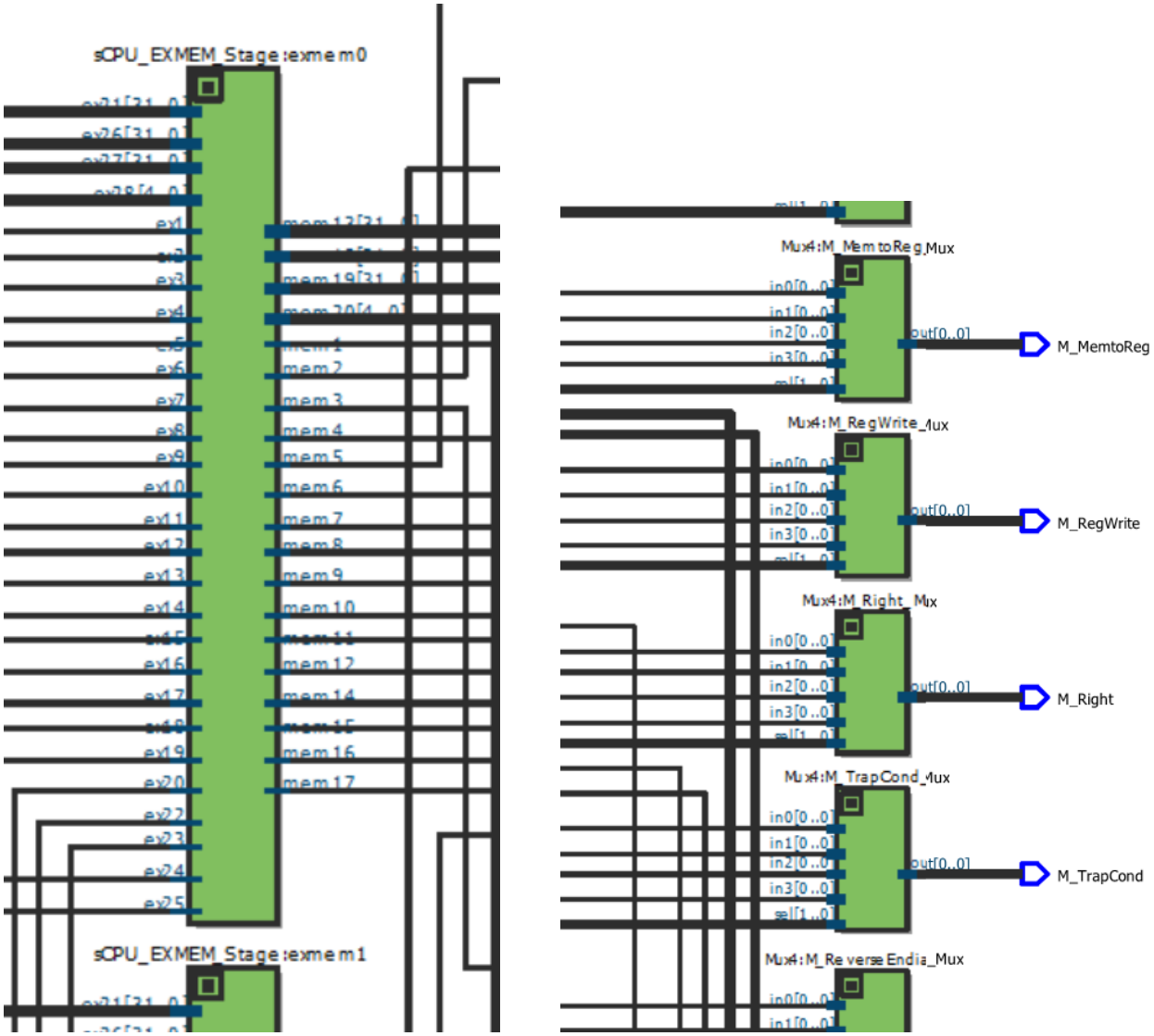


Figură 2-18 Logic exmem1

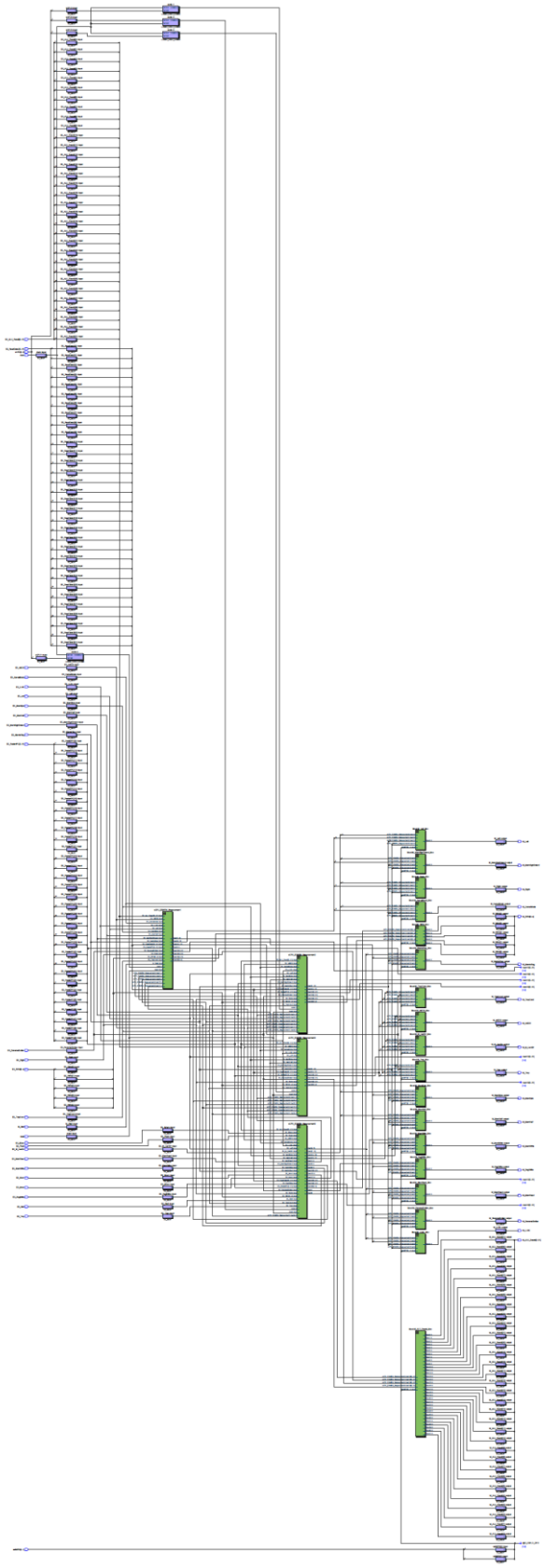
	Multiplexer Inputs	Bus Width	Baseline Area	Area if Restructured	Saving if Restructured	Registered	Example Multiplexer Output
1	3:1	468 bits	936 LEs	0 LEs	936 LEs	Yes	|EXMEM4_Stage| sCPU_EXMEM_Stage:exmem1| mem8
2	4:1	117 bits	234 LEs	234 LEs	0 LEs	No	|EXMEM4_Stage| Mux4:M_MemWrite_Mux|Mux0

	Resource	Usage
1	Estimate of Logic utilization (ALMs needed)	294
2		
3	Combinational ALUT usage for logic	127
1	-- 7 input functions	0
2	-- 6 input functions	118
3	-- 5 input functions	0
4	-- 4 input functions	0
5	-- <=3 input functions	9
4		
5	Dedicated logic registers	468
6		
7	I/O pins	248
8		
9	Total DSP Blocks	0
10		
11	Maximum fan-out node	reset~input
12	Maximum fan-out	469
13	Total fan-out	2967
14	Average fan-out	2.72

	Property	Setup	Hold
1	Illegal Clocks	0	0
2	Unconstrained Clocks	0	0
3	Unconstrained Input Ports	126	126
4	Unconstrained Input Port Paths	1690	1690
5	Unconstrained Output Ports	117	117
6	Unconstrained Output Port Paths	702	702

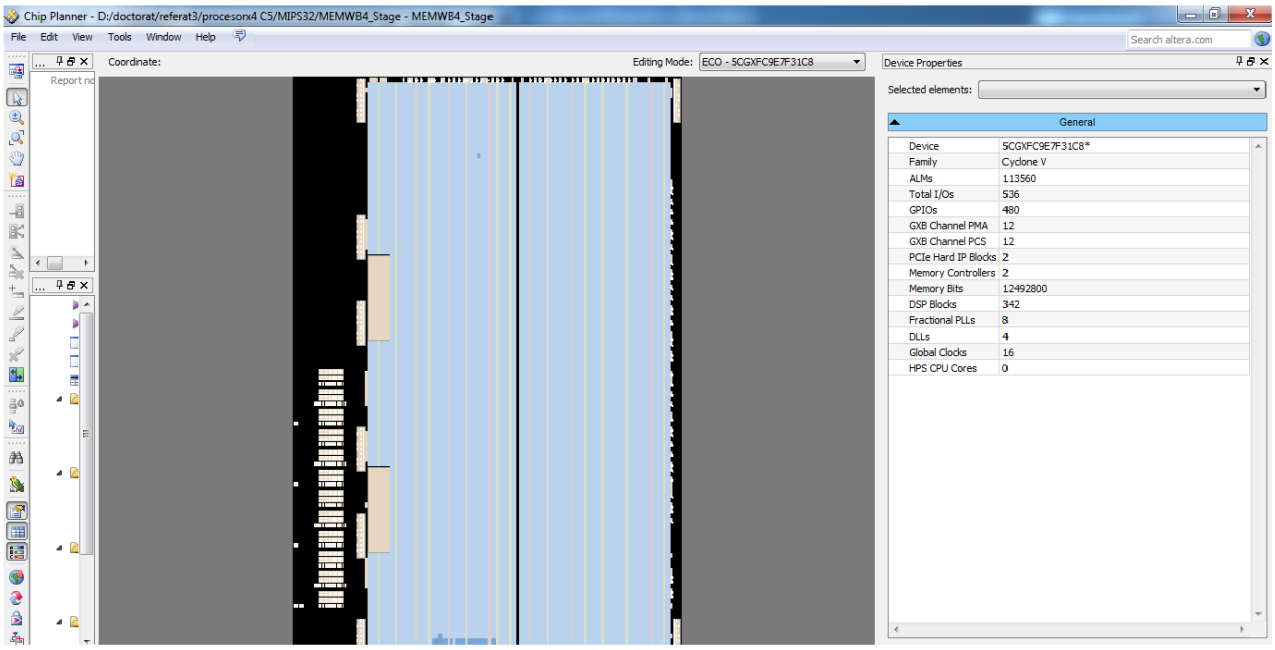


Figură 2-19 Parţial RTL EXMEM4

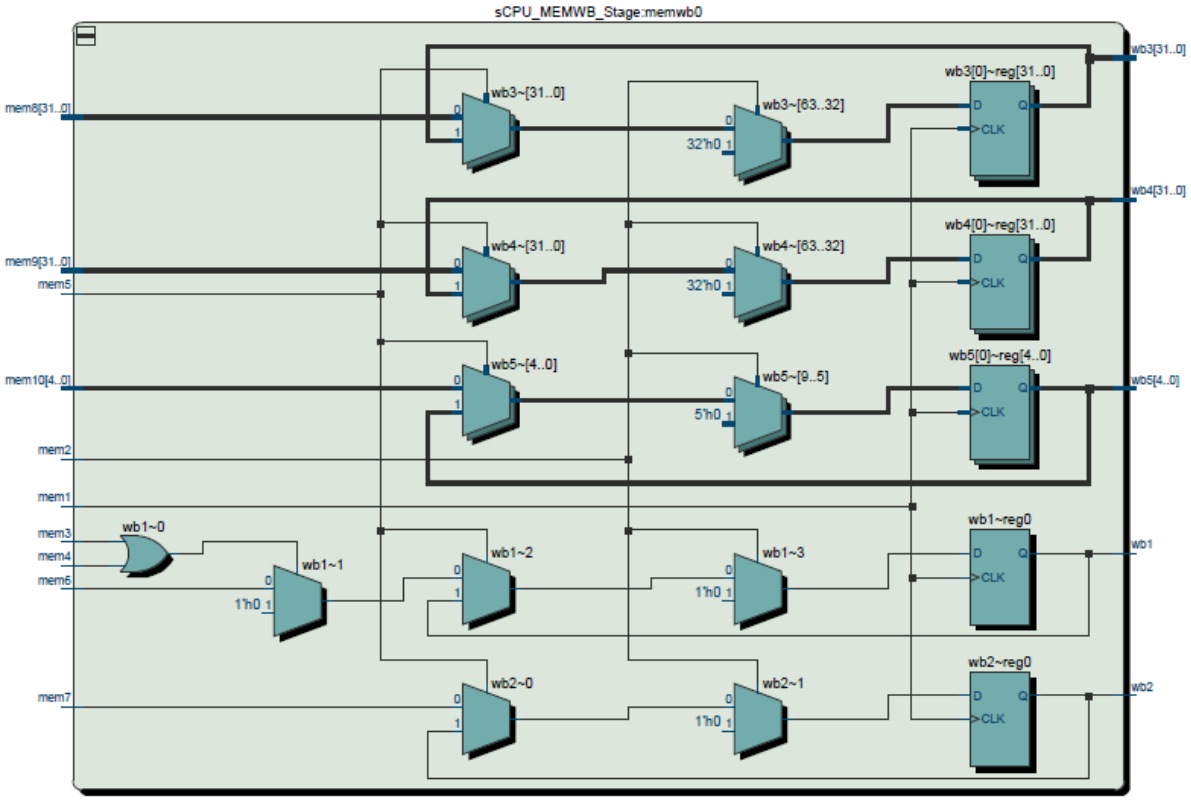


Figură 2-20 Parţial Tehnology map EXMEM4

2.2.4. MEMWB4



Figură 2-21 Chip planner MEMWB4



Figură 2-22 Logic memwb0

```
module  MEMWB4_Stage(

        input       sCPU0,
        input       sCPU1,
        input       sCPU2,
        input       sCPU3,
        input       [ 1: 0]selMIPS,
        input       clock,
        input       reset,
        input       M_Flush,
        input       M_Stall,
        input       WB_Stall,
        // Control Signals
        input       M_RegWrite,
        input       M_MemtoReg,
        // Data Signals
        input       [ 31: 0]M_ReadData,
        input       [ 31: 0]M_ALU_Result,
        input       [ 4: 0]M_RtRd,
        // ----------------
        output      WB_RegWrite,
        output      WB_MemtoReg,
        output      [ 31: 0]WB_ReadData,
        output      [ 31: 0]WB_ALU_Result,
        output      [ 4: 0]WB_RtRd
        );
sCPU_MEMWB_Stagememwb0(
        .mem1(clock&sCPU0),
        .mem2(reset),
        .mem3(M_Flush),
        .mem4(M_Stall),
        .mem5(WB_Stall),
        // Control Signals
        .mem6(M_RegWrite),
        .mem7(M_MemtoReg),
        // Data Signals
        .mem8(M_ReadData),
        .mem9(M_ALU_Result),
        .mem10(M_RtRd),
        // ----------------
        .wb1(WB_RegWrite0),
        .wb2(WB_MemtoReg0),
        .wb3(WB_ReadData0),
        .wb4(WB_ALU_Result0),
        .wb5(WB_RtRd0)
        );

        wire    WB_RegWrite0;
        wire    WB_MemtoReg0;
        wire    [ 31: 0]WB_ReadData0;
        wire    [ 31: 0]WB_ALU_Result0;
        wire    [ 4: 0]WB_RtRd0;

sCPU_MEMWB_Stagememwb1(
        .mem1(clock&sCPU1),
        .mem2(reset),
        .mem3(M_Flush),
        .mem4(M_Stall),
        .mem5(WB_Stall),
        // Control Signals
        .mem6(M_RegWrite),
        .mem7(M_MemtoReg),
        // Data Signals
        .mem8(M_ReadData),
        .mem9(M_ALU_Result),
        .mem10(M_RtRd),
        // ----------------
        .wb1(WB_RegWrite1),
        .wb2(WB_MemtoReg1),
        .wb3(WB_ReadData1),
        .wb4(WB_ALU_Result1),
        .wb5(WB_RtRd1)
        );
        wire    WB_RegWrite1;
        wire    WB_MemtoReg1;
        wire    [ 31: 0]WB_ReadData1;
        wire    [ 31: 0]WB_ALU_Result1;
        wire    [ 4: 0]WB_RtRd1;

sCPU_MEMWB_Stagememwb2(
        .mem1(clock&sCPU2),
        .mem2(reset),
        .mem3(M_Flush),
        .mem4(M_Stall),
        .mem5(WB_Stall),
        // Control Signals
        .mem6(M_RegWrite),
        .mem7(M_MemtoReg),
        // Data Signals
        .mem8(M_ReadData),
        .mem9(M_ALU_Result),
        .mem10(M_RtRd),
        // ----------------
        .wb1(WB_RegWrite2),
        .wb2(WB_MemtoReg2),
        .wb3(WB_ReadData2),
        .wb4(WB_ALU_Result2),
        .wb5(WB_RtRd2)
        );

        wire    WB_RegWrite2;
        wire    WB_MemtoReg2;
        wire    [ 31: 0]WB_ReadData2;
        wire    [ 31: 0]WB_ALU_Result2;
        wire    [ 4: 0]WB_RtRd2;

sCPU_MEMWB_Stagememwb3(
        .mem1(clock&sCPU3),
        .mem2(reset),
        .mem3(M_Flush),
        .mem4(M_Stall),
        .mem5(WB_Stall),
        // Control Signals
        .mem6(M_RegWrite),
        .mem7(M_MemtoReg),
        // Data Signals
        .mem8(M_ReadData),
        .mem9(M_ALU_Result),
        .mem10(M_RtRd),
        // ----------------
        .wb1(WB_RegWrite3),
        .wb2(WB_MemtoReg3),
        .wb3(WB_ReadData3),
        .wb4(WB_ALU_Result3),
        .wb5(WB_RtRd3)
        );

        wire    WB_RegWrite3;
        wire    WB_MemtoReg3;
        wire    [ 31: 0]WB_ReadData3;
        wire    [ 31: 0]WB_ALU_Result3;
        wire    [ 4: 0]WB_RtRd3;

Mux4#(.WIDTH(   1))WB_RegWrite_Mux(
        .sel(selMIPS),
        .in0(WB_RegWrite0),
        .in1(WB_RegWrite1),
        .in2(WB_RegWrite2),
        .in3(WB_RegWrite3),
        .out(WB_RegWrite)
        );

Mux4#(.WIDTH(   1))WB_MemtoReg_Mux(
        .sel(selMIPS),
        .in0(WB_MemtoReg0),
        .in1(WB_MemtoReg1),
        .in2(WB_MemtoReg2),
        .in3(WB_MemtoReg3),
        .out(WB_MemtoReg)
        );

Mux4#(.WIDTH(   32))WB_ReadData_Mux(
        .sel(selMIPS),
        .in0(WB_ReadData0),
        .in1(WB_ReadData1),
        .in2(WB_ReadData2),
        .in3(WB_ReadData3),
        .out(WB_ReadData)
        );

Mux4#(.WIDTH(   32))WB_ALU_Result_Mux(
        .sel(selMIPS),
        .in0(WB_ALU_Result0),
        .in1(WB_ALU_Result1),
        .in2(WB_ALU_Result2),
        .in3(WB_ALU_Result3),
        .out(WB_ALU_Result)
        );

Mux4#(.WIDTH(   5))WB_RtRd_Mux(
        .sel(selMIPS),
        .in0(WB_RtRd0),
        .in1(WB_RtRd1),
        .in2(WB_RtRd2),
        .in3(WB_RtRd3),
        .out(WB_RtRd)
        );


endmodule
```

Figură 2-23 Codul verilog pentru MEMWB4

```
Multiplexer Restructuring Statistics (Restructuring Performed) report for MEMWB4_Stage
Tue Sep 22 22:17:33 2015
Quartus II 64-Bit Version 15.0.0 Build 145 04/22/2015 SJ Web Edition

+-----------------------------------------------------------------------------------------------------------------------------------------------------------+
; Multiplexer Restructuring Statistics (Restructuring Performed)                                                                                            ;
+--------------------+-----------+---------------+----------------------+------------------------+------------+-----------------------------------------+
; Multiplexer Inputs ; Bus Width ; Baseline Area ; Area if Restructured ; Saving if Restructured ; Registered ; Example Multiplexer Output              ;
+--------------------+-----------+---------------+----------------------+------------------------+------------+-----------------------------------------+
; 3:1                ; 284 bits  ; 568 LEs       ; 0 LEs                ; 568 LEs                ; Yes        ; |MEMWB4_Stage|sCPU_MEMWB_Stage:memwb1|wb3[2] ;
; 4:1                ; 71 bits   ; 142 LEs       ; 142 LEs              ; 0 LEs                  ; No         ; |MEMWB4_Stage|Mux4:WB_ReadData_Mux|Mux31     ;
+--------------------+-----------+---------------+----------------------+------------------------+------------+-----------------------------------------+
```

MEMWB4_Stage-Flow Summary

Flow Summary report for MEMWB4_Stage
Tue Sep 22 22:16:41 2015
Quartus II 64-Bit Version 15.0.0 Build 145 04/22/2015 SJ Web Edition

Flow Summary	
Flow Status	Successful - Sat Sep 19 11:53:43 2015
Quartus II 64-Bit Version	15.0.0 Build 145 04/22/2015 SJ Web Edition
Revision Name	MEMWB4_Stage
Top-level Entity Name	MEMWB4_Stage
Family	Cyclone V
Device	5CGXFC9E7F31C8
Timing Models	Final
Logic utilization (in ALMs)	132 / 113,560 (< 1 %)
Total registers	284
Total pins	153 / 536 (29 %)
Total virtual pins	0
Total block memory bits	0 / 12,492,800 (0 %)
Total DSP Blocks	0 / 342 (0 %)
Total HSSI RX PCSs	0 / 12 (0 %)
Total HSSI PMA RX Deserializers	0 / 12 (0 %)
Total HSSI TX PCSs	0 / 12 (0 %)
Total HSSI PMA TX Serializers	0 / 12 (0 %)
Total PLLs	0 / 20 (0 %)
Total DLLs	0 / 4 (0 %)

MEMWB4_Stage-Post-Synthesis Netlist Statistics for Top Partition

Post-Synthesis Netlist Statistics for Top Partition report for MEMWB4_Stage
Tue Sep 22 22:17:48 2015
Quartus II 64-Bit Version 15.0.0 Build 145 04/22/2015 SJ Web Edition

Post-Synthesis Netlist Statistics for Top Partition	
Type	Count
arriav_ff	284
ENA SCLR	284
arriav_lcell_comb	77
normal	77
2 data inputs	5
3 data inputs	1
6 data inputs	71
boundary_port	153
Max LUT depth	1.00
Average LUT depth	0.84

MEMWB4_Stage-Routing Usage Summary

Routing Usage Summary report for MEMWB4_Stage
Tue Sep 22 22:19:15 2015
Quartus II 64-Bit Version 15.0.0 Build 145 04/22/2015 SJ Web Edition

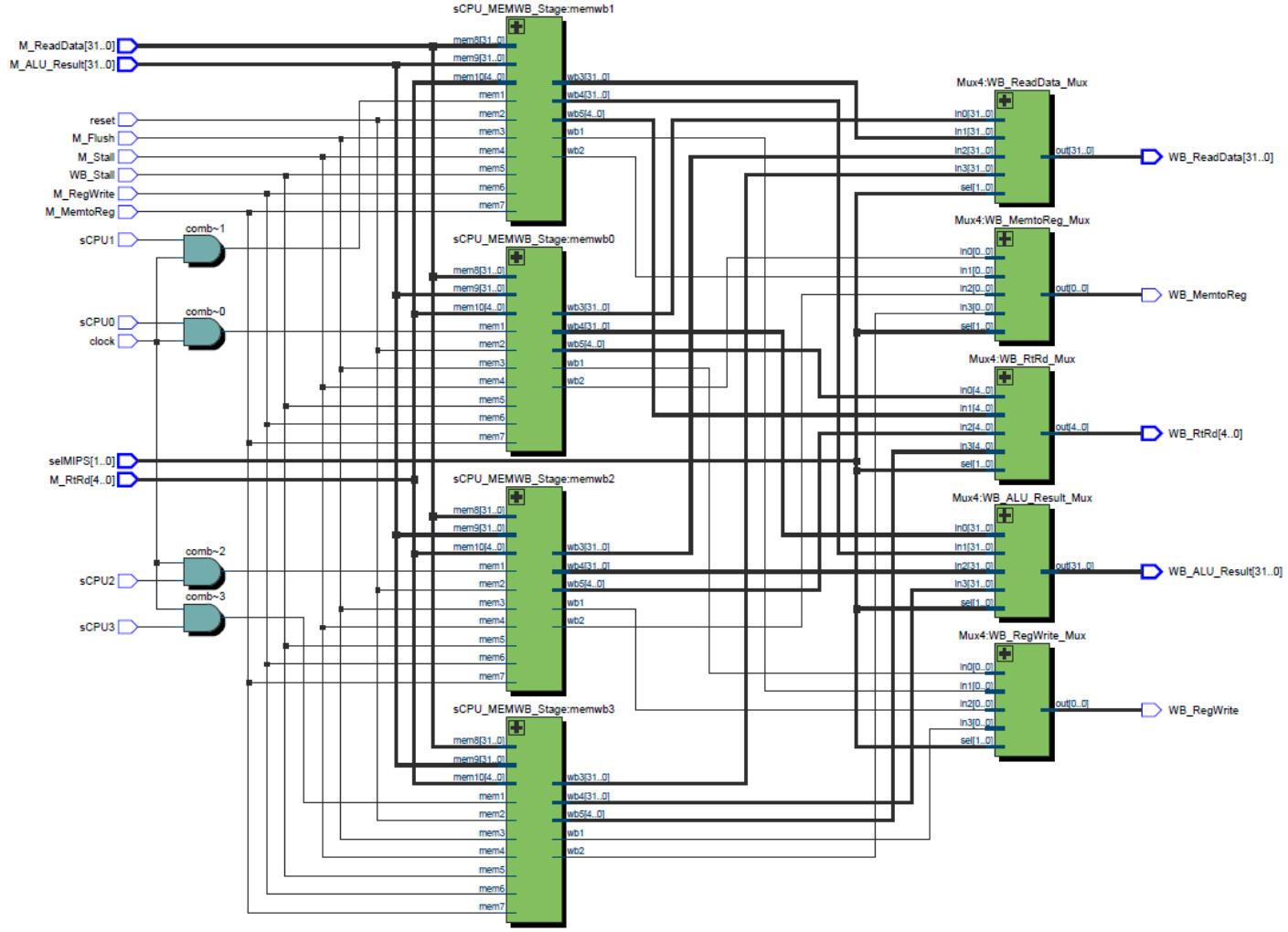
Routing Usage Summary	
Routing Resource Type	Usage
Block interconnects	764 / 721,028 (< 1 %)
C12 interconnects	21 / 30,780 (< 1 %)
C2 interconnects	235 / 303,248 (< 1 %)
C4 interconnects	169 / 140,620 (< 1 %)
DQS bus muxes	0 / 35 (0 %)
DQS-18 I/O buses	0 / 35 (0 %)
DQS-9 I/O buses	0 / 35 (0 %)
Direct links	22 / 721,028 (< 1 %)
Global clocks	0 / 16 (0 %)
Horizontal periphery clocks	0 / 96 (0 %)
Local interconnects	6 / 227,120 (< 1 %)
Quadrant clocks	0 / 88 (0 %)
R14 interconnects	143 / 30,168 (< 1 %)
R14/C12 interconnect drivers	143 / 53,952 (< 1 %)
R3 interconnects	375 / 331,920 (< 1 %)
R6 interconnects	766 / 622,512 (< 1 %)
Spine clocks	0 / 480 (0 %)
Wire stub REs	0 / 40,996 (0 %)

MEMWB4_Stage-Resource Usage Summary

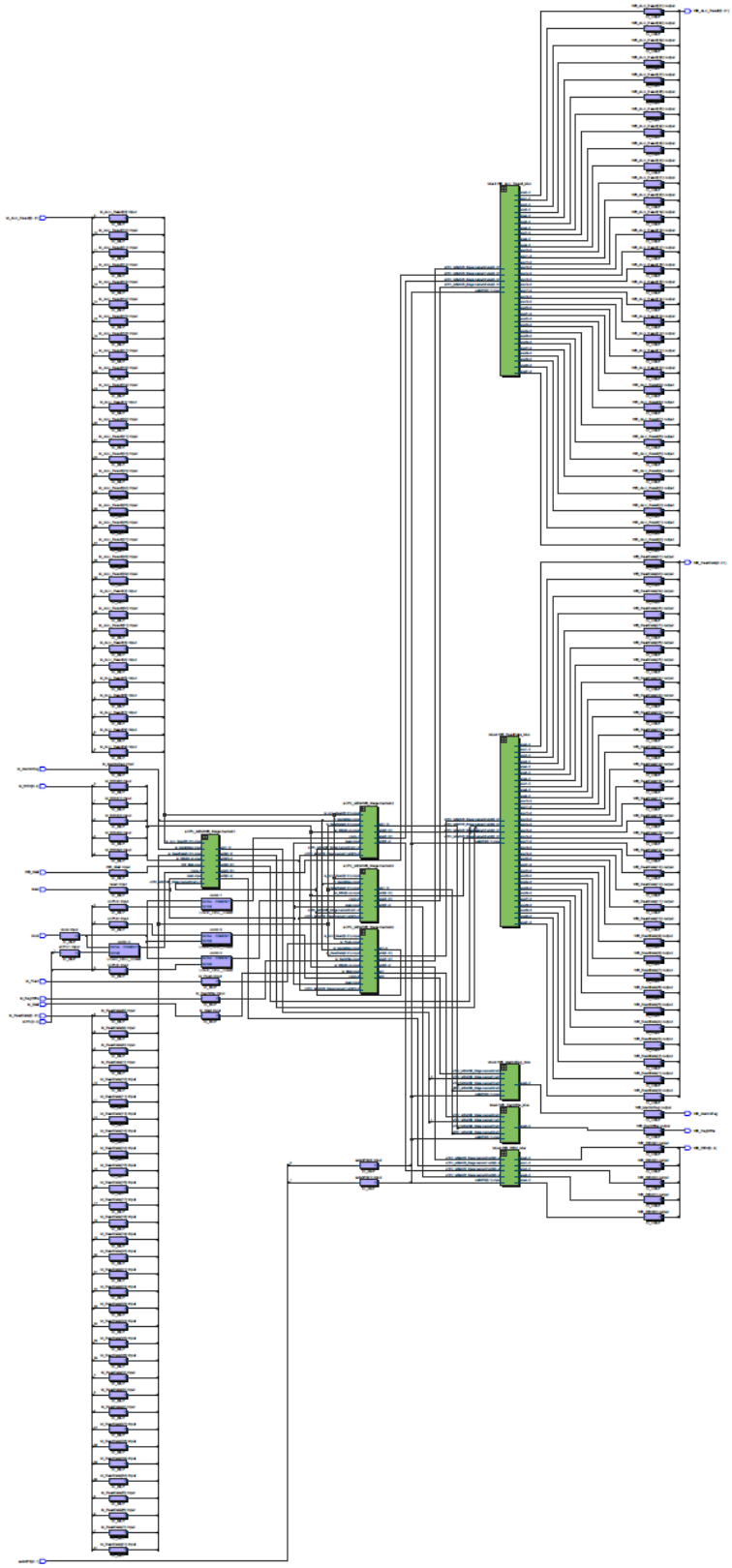
Resource Usage Summary report for MEMWB4_Stage
Tue Sep 22 22:18:24 2015
Quartus II 64-Bit Version 15.0.0 Build 145 04/22/2015 SJ Web Edition

Fitter Resource Usage Summary

Resource	Usage	%
Logic utilization (ALMs needed / total ALMs on device)	132 / 113,560	< 1 %
ALMs needed [=A-B+C]	132	
[A] ALMs used in final placement [=a+b+c+d]	182 / 113,560	< 1 %
[a] ALMs used for LUT logic and registers	34	
[b] ALMs used for LUT logic	39	
[c] ALMs used for registers	109	
[d] ALMs used for memory (up to half of total ALMs)	0	
[B] Estimate of ALMs recoverable by dense packing	55 / 113,560	< 1 %
[C] Estimate of ALMs unavailable [=a+b+c+d]	5 / 113,560	< 1 %
[a] Due to location constrained logic	0	
[b] Due to LAB-wide signal conflicts	0	
[c] Due to LAB input limits	5	
[d] Due to virtual I/Os	0	
Difficulty packing design	Low	
Total LABs: partially or completely used	50 / 11,356	< 1 %
-- Logic LABs	50	
-- Memory LABs (up to half of total LABs)	0	
Combinational ALUT usage for logic	78	
-- 7 input functions	0	
-- 6 input functions	71	
-- 5 input functions	0	
-- 4 input functions	0	
-- <=3 input functions	7	
Combinational ALUT usage for route-throughs	138	
Dedicated logic registers	284	
-- By type:		
-- Primary logic registers	284 / 227,120	< 1 %
-- Secondary logic registers	0 / 227,120	0 %
-- By function:		
-- Design implementation registers	284	
-- Routing optimization registers	0	
Virtual pins	0	
I/O pins	153 / 536	29 %
-- Clock pins	4 / 16	25 %
-- Dedicated input pins	0 / 35	0 %
Global signals	0	
M10K blocks	0 / 1,220	0 %
Total MLAB memory bits	0	
Total block memory bits	0 / 12,492,800	0 %
Total block memory implementation bits	0 / 12,492,800	0 %
Total DSP Blocks	0 / 342	0 %
Fractional PLLs	0 / 8	0 %
Global clocks	0 / 16	0 %
Quadrant clocks	0 / 88	0 %
Horizontal periphery clocks	0 / 24	0 %
SERDES Transmitters	0 / 140	0 %
SERDES Receivers	0 / 140	0 %
JTAGs	0 / 1	0 %
ASMI blocks	0 / 1	0 %
CRC blocks	0 / 1	0 %
Remote update blocks	0 / 1	0 %
Oscillator blocks	0 / 1	0 %
Hard IPs	0 / 2	0 %
Standard RX PCSs	0 / 12	0 %
HSSI PMA RX Deserializers	0 / 12	0 %
Standard TX PCSs	0 / 12	0 %
HSSI PMA TX Serializers	0 / 12	0 %
Channel PLLs	0 / 12	0 %
Impedance control blocks	0 / 3	0 %
Hard Memory Controllers	0 / 2	0 %
Average interconnect usage (total/H/V)	0.1% / 0.2% / 0.1%	
Peak interconnect usage (total/H/V)	4.6% / 5.2% / 2.7%	
Maximum fan-out	285	
Highest non-global fan-out	285	
Total fan-out	1937	
Average fan-out	2.40	



Figură 2-24 RTL MEMWB4



Figură 2-25 Tehnology map MEMWB4

2.2.5. PC4

```
1   module   PC4(
2         input    [ 1: 0]selPC,
3         input    en_MIPS,
4         input    clock,
5         input    reset,
6         input    enable,
7         input    [ 31: 0]IF_PCIn,
8
9         output    sCPU0,
10        output    sCPU1,
11        output    sCPU2,
12        output    sCPU3,
13        output    [ 31: 0]IF_PCOut
14
15        );
16
17  wire  [ 31: 0]Q0,Q1,Q2,Q3;
18
19
20        /*** Program Counter (MIPS spec is 0xBFC00000 starting address) ***/
21        Register#(.WIDTH(        32))PC0(
22              .clock(clock),
23              .reset(reset),
24              //.enable  (~IF_Stall),   // XXX verify. HERE. Was 1 but on stall latches PC+4, ad
    nauseum.
25              .enable((enable&sCPU0)),
26              .D(IF_PCIn),
27              .Q(Q0)
28        );
29
30
31        /*** Program Counter (MIPS spec is 0xBFC00000 starting address) ***/
32        Register#(.WIDTH(        32))PC1(
33              .clock(clock),
34              .reset(reset),
35              //.enable  (~IF_Stall),   // XXX verify. HERE. Was 1 but on stall latches PC+4, ad
    nauseum.
36              .enable((enable&sCPU1)),
37              .D(IF_PCIn),
38              .Q(Q1)
39        );
40
41        /*** Program Counter (MIPS spec is 0xBFC00000 starting address) ***/
42        Register#(.WIDTH(        32))PC2(
43              .clock(clock),
44              .reset(reset),
45              //.enable  (~IF_Stall),   // XXX verify. HERE. Was 1 but on stall latches PC+4, ad
    nauseum.
46              .enable((enable&sCPU2)),
47              .D(IF_PCIn),
48              .Q(Q2)
49        );
50
51        /*** Program Counter (MIPS spec is 0xBFC00000 starting address) ***/
52        Register#(.WIDTH(        32))PC3(
53              .clock(clock),
54              .reset(reset),
55              //.enable  (~IF_Stall),   // XXX verify. HERE. Was 1 but on stall latches PC+4, ad
    nauseum.
56              .enable((enable&sCPU3)),
57              .D(IF_PCIn),
58              .Q(Q3)
59        );
60
61
62
63        /*** PC Source Non-Exception Mux ***/
64        Mux4#(.WIDTH(    32 ))PCsCPUi_Mux(
65              .sel(selPC),
66              .in0(Q0),                          //(IF_PCout0),
67              .in1(Q1),                         // (IF_PCout1),
68              .in2(Q2),                          // (IF_PCout2),
69              .in3(Q3),                          //(IF_PCout3),
70              .out(IF_PCOut)
71        );
72
73  endmodule
74
```

PC4-Flow Summary

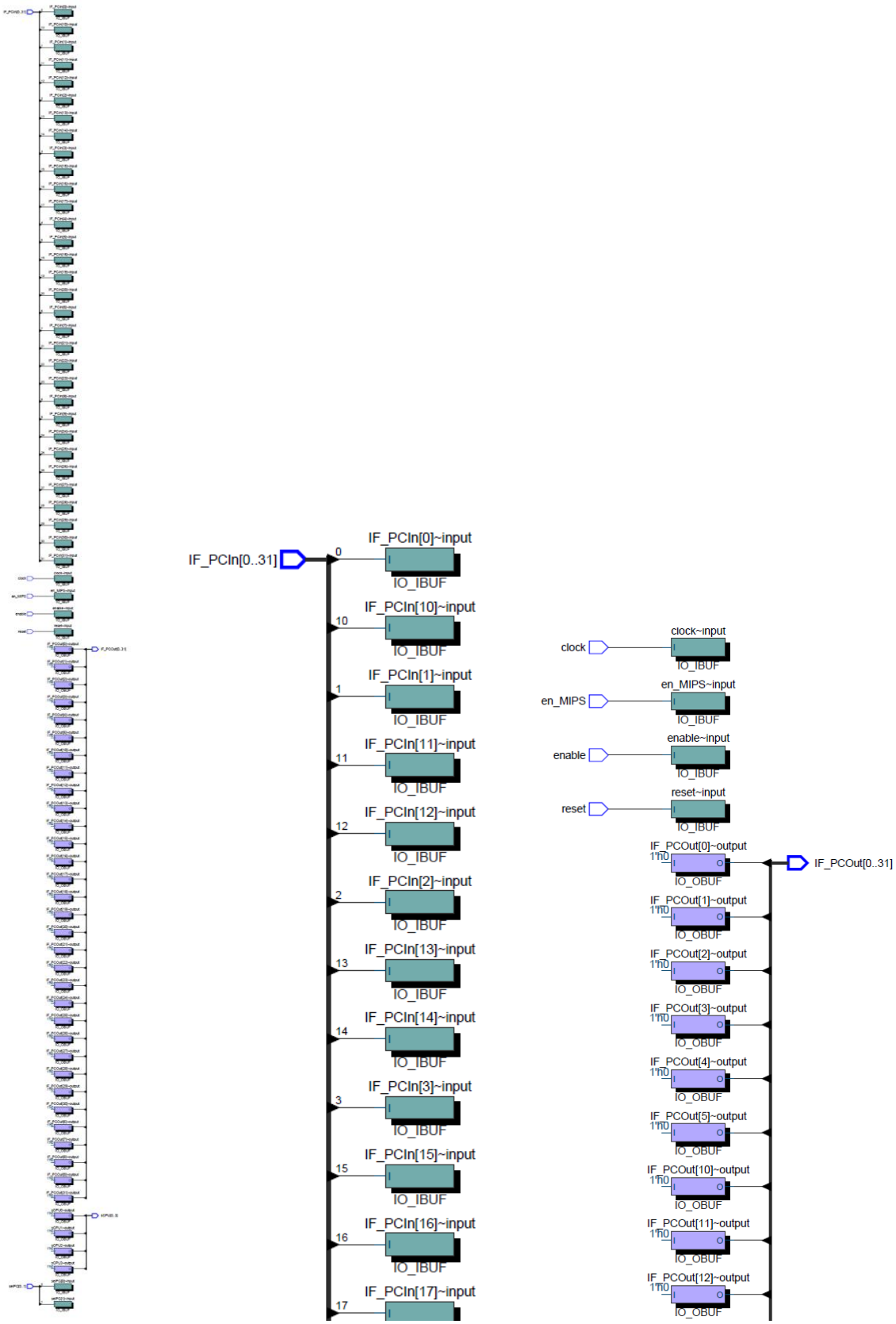
Flow Summary report for PC4
Tue Sep 22 22:23:56 2015
Quartus II 64-Bit Version 15.0.0 Build 145 04/22/2015 SJ Web Edition

Flow Summary	
Flow Status	Successful - Sat Sep 19 17:50:15 2015
Quartus II 64-Bit Version	15.0.0 Build 145 04/22/2015 SJ Web Edition
Revision Name	PC4
Top-level Entity Name	PC4
Family	Cyclone V
Device	5CGXFC9E7F31C8
Timing Models	Final
Logic utilization (in ALMs)	1 / 113,560 (< 1 %)
Total registers	0
Total pins	74 / 536 (14 %)
Total virtual pins	0
Total block memory bits	0 / 12,492,800 (0 %)
Total DSP Blocks	0 / 342 (0 %)
Total HSSI RX PCSs	0 / 12 (0 %)
Total HSSI PMA RX Deserializers	0 / 12 (0 %)
Total HSSI TX PCSs	0 / 12 (0 %)
Total HSSI PMA TX Serializers	0 / 12 (0 %)
Total PLLs	0 / 20 (0 %)
Total DLLs	0 / 4 (0 %)

PC4-Post-Synthesis Netlist Statistics for Top Partition

Post-Synthesis Netlist Statistics for Top Partition report for PC4
Tue Sep 22 22:24:43 2015
Quartus II 64-Bit Version 15.0.0 Build 145 04/22/2015 SJ Web Edition

Post-Synthesis Netlist Statistics for Top Partition	
Type	Count
arriav_lcell_comb	1
normal	1
0 data inputs	1
boundary_port	74
Max LUT depth	0.00
Average LUT depth	0.00



Figură 2-26 Tehnology map PC4

2.2.6. Register file4

```
1   module   RegisterFile4(
2
3       input    sCPU0,
4       input    sCPU1,
5       input    sCPU2,
6       input    sCPU3,
7       input    [ 1: 0]selMIPS,
8       input    clock,
9       input    reset,
10      input    [ 4: 0]ReadReg1,
11      input    [ 4: 0]ReadReg2,
12      input    [ 4: 0]WriteReg,
13      input    [ 31: 0]WriteData,
14      input    RegWrite,
15
16      output    [ 31: 0]ID_ReadData1_RF,
17      output    [ 31: 0]ID_ReadData2_RF
18
19  );
20
21  wire   [ 31: 0]ID_ReadData1_RF0;
22  wire   [ 31: 0]ID_ReadData2_RF0;
23  wire   [ 31: 0]ID_ReadData1_RF1;
24  wire   [ 31: 0]ID_ReadData2_RF1;
25  wire   [ 31: 0]ID_ReadData1_RF2;
26  wire   [ 31: 0]ID_ReadData2_RF2;
27  wire   [ 31: 0]ID_ReadData1_RF3;
28  wire   [ 31: 0]ID_ReadData2_RF3;
29
30
31      /*** Register File ***/
32          RegisterFileRegisterFile_sCPU0(
33              .clock(clock&sCPU0),
34              .reset(reset),
35              .ReadReg1(ReadReg1),
36              .ReadReg2(ReadReg2),
37              .WriteReg(WriteReg),
38              .WriteData(WriteData),
39              .RegWrite(RegWrite),
40              .ReadData1(ID_ReadData1_RF0),
41              .ReadData2(ID_ReadData2_RF0)
42          );
43
44          /*** Register File ***/
45          RegisterFileRegisterFile_sCPU1(
46              .clock(clock&sCPU1),
47              .reset(reset),
48              .ReadReg1(ReadReg1),
49              .ReadReg2(ReadReg2),
50              .WriteReg(WriteReg),
51              .WriteData(WriteData),
52              .RegWrite(RegWrite),
53              .ReadData1(ID_ReadData1_RF1),
54              .ReadData2(ID_ReadData2_RF1)
55          );
56
57          /*** Register File ***/
58          RegisterFileRegisterFile_sCPU2(
59              .clock(clock&sCPU2),
60              .reset(reset),
61              .ReadReg1(ReadReg1),
62              .ReadReg2(ReadReg2),
63              .WriteReg(WriteReg),
64              .WriteData(WriteData),
65              .RegWrite(RegWrite),
66              .ReadData1(ID_ReadData1_RF2),
67              .ReadData2(ID_ReadData2_RF2)
68          );
69
70          /*** Register File ***/
71          RegisterFileRegisterFile_sCPU3(
72              .clock(clock&sCPU3),
73              .reset(reset),
74              .ReadReg1(ReadReg1),
75              .ReadReg2(ReadReg2),
76              .WriteReg(WriteReg),
77              .WriteData(WriteData),
78              .RegWrite(RegWrite),
79              .ReadData1(ID_ReadData1_RF3),
80              .ReadData2(ID_ReadData2_RF3)
81          );
82
83          /*** ID Rs Forwarding/Link Mux ***/
84          Mux4#(.WIDTH(    32 ))RegFile_Data1(
85              .sel(selMIPS),
86              .in0(ID_ReadData1_RF0),
87              .in1(ID_ReadData1_RF1),
88              .in2(ID_ReadData1_RF2),
89              .in3(ID_ReadData1_RF3),
90              .out(ID_ReadData1_RF)
91              );
92
93            /*** ID Rs Forwarding/Link Mux ***/
94          Mux4#(.WIDTH(    32 ))RegFile_Data2(
95              .sel(selMIPS),
96              .in0(ID_ReadData2_RF0),
97              .in1(ID_ReadData2_RF1),
98              .in2(ID_ReadData2_RF2),
99              .in3(ID_ReadData2_RF3),
100             .out(ID_ReadData2_RF)
101             );
102
103 endmodule
```

Figură 2-27 Codul verilog pentru Register file4

	Source Clock(s)	Destination Clock(s)	Delay Added in ns
1	I/O	clock	421.9
2	clock,I/O,clock	clock	184.0
3	clock	clock	180.0

RegisterFile4-Flow Summary

```
Flow Summary report for RegisterFile4
Tue Sep 22 22:33:39 2015
Quartus II 64-Bit Version 15.0.0 Build 145 04/22/2015 SJ Web Edition
```

Flow Summary	
Flow Status	Successful - Sat Sep 19 15:28:12 2015
Quartus II 64-Bit Version	15.0.0 Build 145 04/22/2015 SJ Web Edition
Revision Name	RegisterFile4
Top-level Entity Name	RegisterFile4
Family	Cyclone V
Device	5CGXFC9E7F31C8
Timing Models	Final
Logic utilization (in ALMs)	4,247 / 113,560 (4 %)
Total registers	3968
Total pins	120 / 536 (22 %)
Total virtual pins	0
Total block memory bits	0 / 12,492,800 (0 %)
Total DSP Blocks	0 / 342 (0 %)
Total HSSI RX PCSs	0 / 12 (0 %)
Total HSSI PMA RX Deserializers	0 / 12 (0 %)
Total HSSI TX PCSs	0 / 12 (0 %)
Total HSSI PMA TX Serializers	0 / 12 (0 %)
Total PLLs	0 / 20 (0 %)
Total DLLs	0 / 4 (0 %)

RegisterFile4-Post-Synthesis Netlist Statistics for Top Partition

```
Post-Synthesis Netlist Statistics for Top Partition report for RegisterFile4
Tue Sep 22 22:34:57 2015
Quartus II 64-Bit Version 15.0.0 Build 145 04/22/2015 SJ Web Edition
```

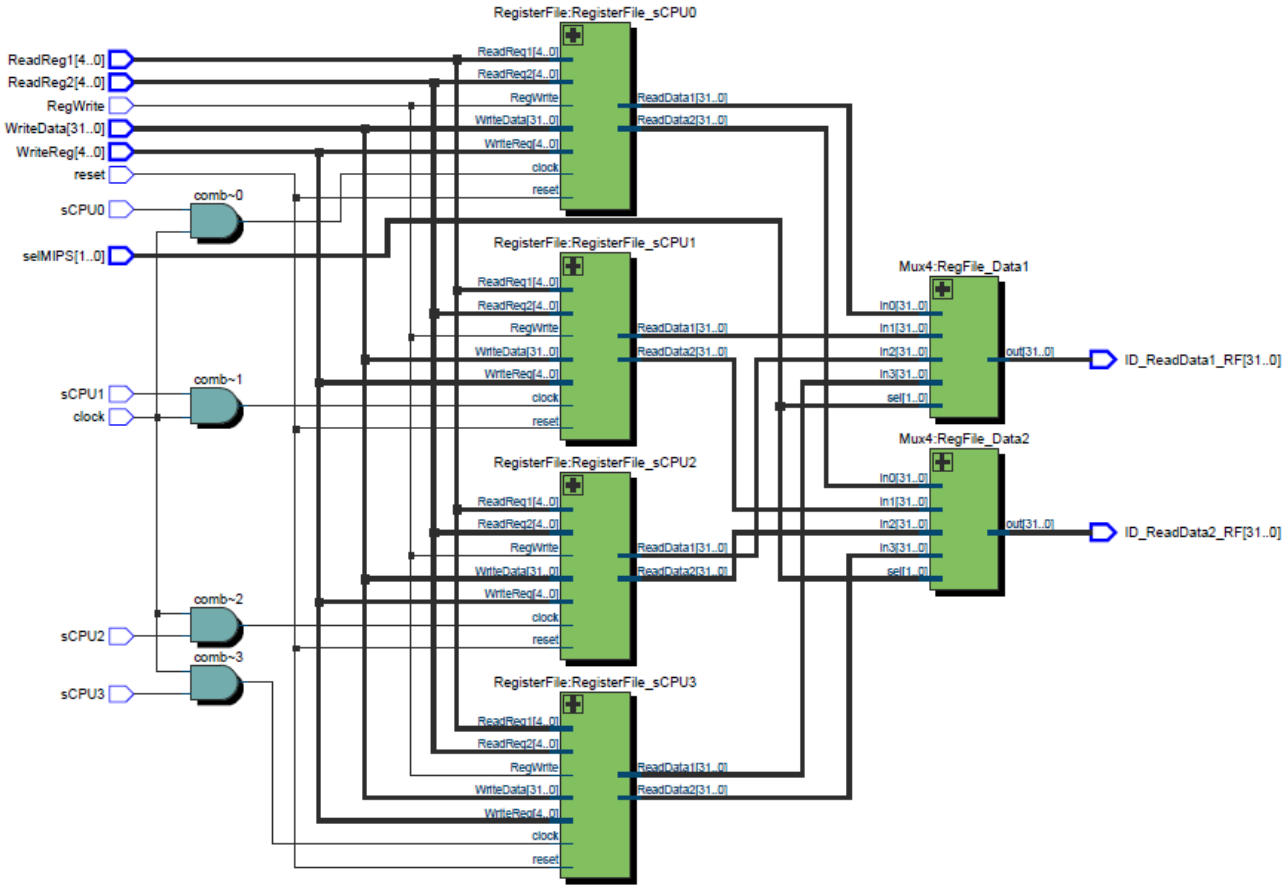
Post-Synthesis Netlist Statistics for Top Partition	
Type	Count
arriav_ff	3968
ENA SCLR	3968
arriav_lcell_comb	4115
extend	64
7 data inputs	64
normal	4051
2 data inputs	42
3 data inputs	6
4 data inputs	2
5 data inputs	258
6 data inputs	3743
boundary_port	120
Max LUT depth	4.00
Average LUT depth	3.56

RegisterFile4-Routing Usage Summary

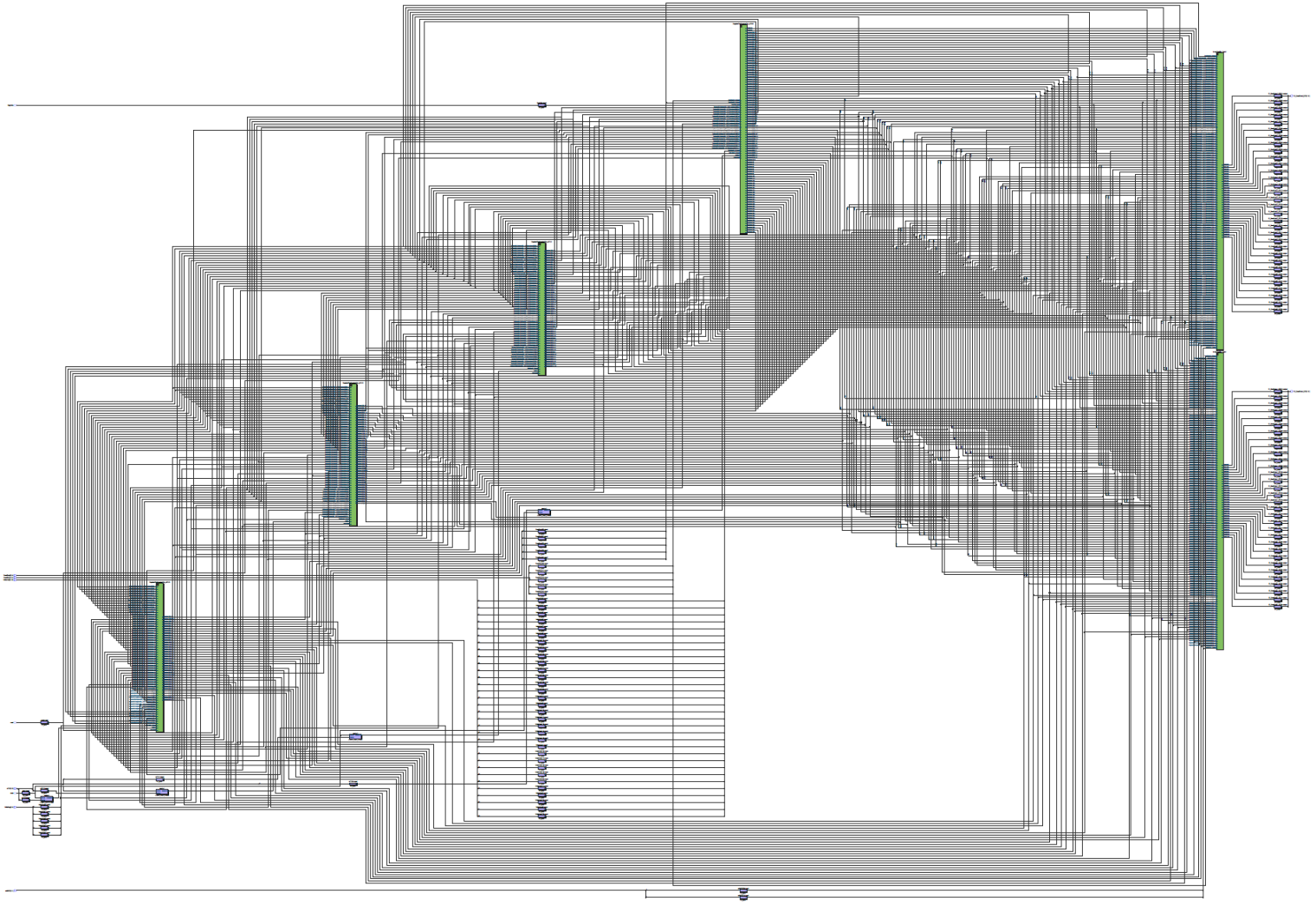
Routing Usage Summary report for RegisterFile4
Tue Sep 22 22:36:35 2015
Quartus II 64-Bit Version 15.0.0 Build 145 04/22/2015 SJ Web Edition

; Routing Usage Summary

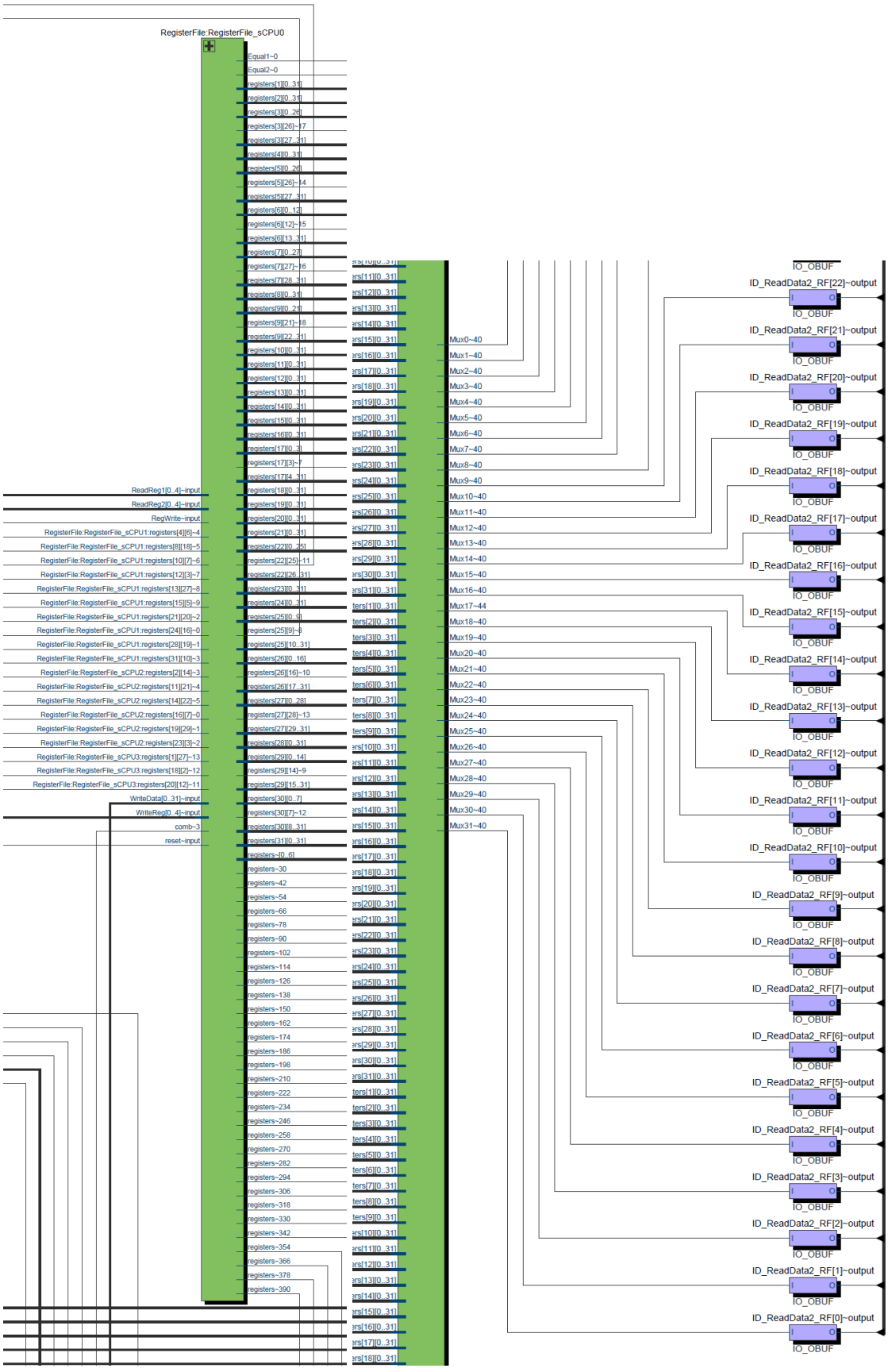
Routing Resource Type	Usage
Block interconnects	19,964 / 721,028 (3 %)
C12 interconnects	312 / 30,780 (1 %)
C2 interconnects	6,354 / 303,248 (2 %)
C4 interconnects	3,578 / 140,620 (3 %)
DQS bus muxes	0 / 35 (0 %)
DQS-18 I/O buses	0 / 35 (0 %)
DQS-9 I/O buses	0 / 35 (0 %)
Direct links	666 / 721,028 (< 1 %)
Global clocks	0 / 16 (0 %)
Horizontal periphery clocks	0 / 96 (0 %)
Local interconnects	3,293 / 227,120 (1 %)
Quadrant clocks	0 / 88 (0 %)
R14 interconnects	733 / 30,168 (2 %)
R14/C12 interconnect drivers	801 / 53,952 (1 %)
R3 interconnects	8,450 / 331,920 (3 %)
R6 interconnects	13,601 / 622,512 (2 %)
Spine clocks	0 / 480 (0 %)
Wire stub REs	0 / 40,996 (0 %)



Figură 2-28 RTL Register file4

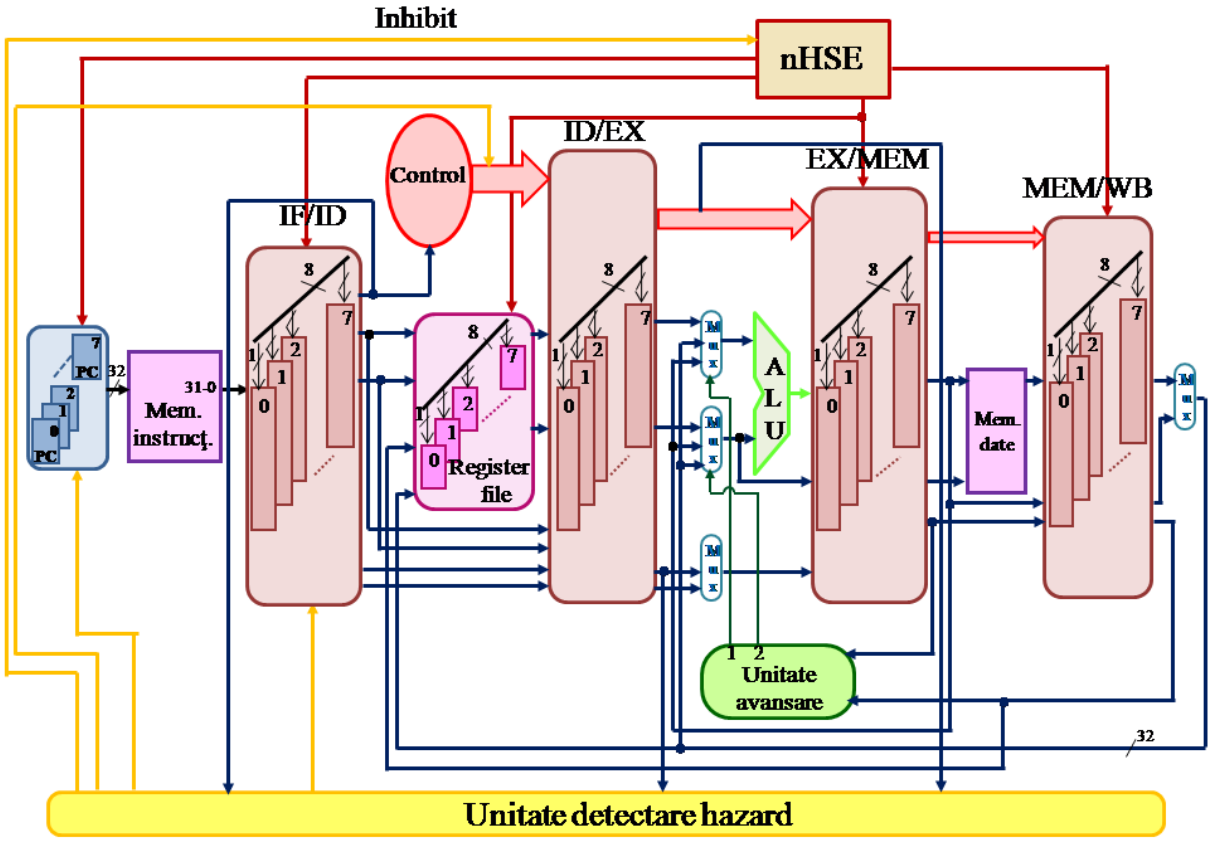


Figură 2-29 Tehnology map Register file4



Figură 2-30 Parţial Tehnology map Register file4

2.3. 8MPRA



Figură 2-31 *8MPRA*

PowerPlay Power Analyzer Status	Successful - Sat Sep 26 22:37:13 2015
Quartus II 64-Bit Version	15.0.0 Build 145 04/22/2015 SJ Web Edition
Revision Name	Processor8
Top-level Entity Name	Processor8
Family	Cyclone V
Device	5CGXFC7D7F31C8
Power Models	Final
Total Thermal Power Dissipation	365.20 mW
Core Dynamic Thermal Power Dissipation	0.00 mW
Core Static Thermal Power Dissipation	348.96 mW
I/O Thermal Power Dissipation	16.24 mW
Power Estimation Confidence	Low: user provided insufficient toggle rate data

	Source Clock(s)	Destination Clock(s)	Delay Added in ns
1	clock	clock	3.8

	Source Register	Destination Register	Delay Added in ns
1	Register:IODataIn|Q[27]	Register:IODataReg|Q[27]	0.154
2	Register:IODataIn|Q[23]	Register:IODataReg|Q[23]	0.154
3	Register:IODataIn|Q[22]	Register:IODataReg|Q[22]	0.151
4	Register:IODataIn|Q[18]	Register:IODataReg|Q[18]	0.151
5	Register:IODataIn|Q[26]	Register:IODataReg|Q[26]	0.151
6	Register:IODataIn|Q[15]	Register:IODataReg|Q[15]	0.149
7	Register:IODataIn|Q[8]	Register:IODataReg|Q[8]	0.149
8	Register:IODataIn|Q[29]	Register:IODataReg|Q[29]	0.148
9	Register:IODataIn|Q[5]	Register:IODataReg|Q[5]	0.147
10	Register:IODataIn|Q[28]	Register:IODataReg|Q[28]	0.146
11	Register:IODataIn|Q[11]	Register:IODataReg|Q[11]	0.145
12	Register:IODataIn|Q[13]	Register:IODataReg|Q[13]	0.144
13	Register:IODataIn|Q[12]	Register:IODataReg|Q[12]	0.144
14	Register:IODataIn|Q[6]	Register:IODataReg|Q[6]	0.144
15	Register:IODataIn|Q[2]	Register:IODataReg|Q[2]	0.144
16	Register:IODataIn|Q[0]	Register:IODataReg|Q[0]	0.142
17	Register:IODataIn|Q[4]	Register:IODataReg|Q[4]	0.141
18	Register:IODataIn|Q[17]	Register:IODataReg|Q[17]	0.124
19	Register:IODataIn|Q[10]	Register:IODataReg|Q[10]	0.124
20	Register:IODataIn|Q[1]	Register:IODataReg|Q[1]	0.119
21	Register:IODataIn|Q[25]	Register:IODataReg|Q[25]	0.084
22	Register:IODataIn|Q[7]	Register:IODataReg|Q[7]	0.082
23	Register:IODataIn|Q[31]	Register:IODataReg|Q[31]	0.082
24	Register:IODataIn|Q[14]	Register:IODataReg|Q[14]	0.082
25	Register:IODataIn|Q[3]	Register:IODataReg|Q[3]	0.082
26	Register:IODataIn|Q[30]	Register:IODataReg|Q[30]	0.079
27	Register:IODataIn|Q[21]	Register:IODataReg|Q[21]	0.079
28	Register:IODataIn|Q[20]	Register:IODataReg|Q[20]	0.079
29	Register:IODataIn|Q[19]	Register:IODataReg|Q[19]	0.079
30	Register:IODataIn|Q[9]	Register:IODataReg|Q[9]	0.079
31	Register:IODataIn|Q[16]	Register:IODataReg|Q[16]	0.077
32	Register:IODataIn|Q[24]	Register:IODataReg|Q[24]	0.074

Processor8-Flow Summary

```
Flow Summary report for Processor8
Wed Sep 23 14:46:36 2015
Quartus II 64-Bit Version 15.0.0 Build 145 04/22/2015 SJ Web Edition
```

Flow Summary	
Flow Status	Successful - Tue Sep 22 01:12:45 2015
Quartus II 64-Bit Version	15.0.0 Build 145 04/22/2015 SJ Web Edition
Revision Name	Processor8
Top-level Entity Name	Processor8
Family	Cyclone V
Device	5CGXFC7D7F31C8
Timing Models	Final
Logic utilization (in ALMs)	17 / 56,480 (< 1 %)
Total registers	64
Total pins	302 / 522 (58 %)
Total virtual pins	0
Total block memory bits	0 / 7,024,640 (0 %)
Total DSP Blocks	0 / 156 (0 %)
Total HSSI RX PCSs	0 / 9 (0 %)
Total HSSI PMA RX Deserializers	0 / 9 (0 %)
Total HSSI TX PCSs	0 / 9 (0 %)
Total HSSI PMA TX Serializers	0 / 9 (0 %)
Total PLLs	0 / 16 (0 %)
Total DLLs	0 / 4 (0 %)

Processor8-Routing Usage Summary

Routing Usage Summary report for Processor8
Wed Sep 23 14:48:42 2015
Quartus II 64-Bit Version 15.0.0 Build 145 04/22/2015 SJ Web Edition

Routing Usage Summary	
Routing Resource Type	Usage
Block interconnects	103 / 374,484 (< 1 %)
C12 interconnects	1 / 16,664 (< 1 %)
C2 interconnects	71 / 155,012 (< 1 %)
C4 interconnects	31 / 72,600 (< 1 %)
DQS bus muxes	0 / 30 (0 %)
DQS-18 I/O buses	0 / 30 (0 %)
DQS-9 I/O buses	0 / 30 (0 %)
Direct links	0 / 374,484 (0 %)
Global clocks	1 / 16 (6 %)
Horizontal periphery clocks	0 / 72 (0 %)
Local interconnects	31 / 112,960 (< 1 %)
Quadrant clocks	0 / 88 (0 %)
R14 interconnects	8 / 15,868 (< 1 %)
R14/C12 interconnect drivers	8 / 27,256 (< 1 %)
R3 interconnects	84 / 169,296 (< 1 %)
R6 interconnects	31 / 330,800 (< 1 %)
Spine clocks	3 / 480 (< 1 %)
Wire stub REs	0 / 20,834 (0 %)

2.3.1. IFID8

IFID8_Stage-Flow Summary

Flow Summary report for IFID8_Stage
Wed Sep 23 14:12:35 2015
Quartus II 64-Bit Version 15.0.0 Build 145 04/22/2015 SJ Web Edition

Flow Summary	
Flow Status	Successful - Mon Sep 21 23:22:47 2015
Quartus II 64-Bit Version	15.0.0 Build 145 04/22/2015 SJ Web Edition
Revision Name	IFID8_Stage
Top-level Entity Name	IFID8_Stage
Family	Cyclone V
Device	5CGXFC7D7F31C8
Timing Models	Final
Logic utilization (in ALMs)	1 / 56,480 (< 1 %)
Total registers	0
Total pins	210 / 522 (40 %)
Total virtual pins	0
Total block memory bits	0 / 7,024,640 (0 %)
Total DSP Blocks	0 / 156 (0 %)
Total HSSI RX PCSs	0 / 9 (0 %)
Total HSSI PMA RX Deserializers	0 / 9 (0 %)
Total HSSI TX PCSs	0 / 9 (0 %)
Total HSSI PMA TX Serializers	0 / 9 (0 %)
Total PLLs	0 / 16 (0 %)
Total DLLs	0 / 4 (0 %)

```
1
2   module  IFID8_Stage(
3
4           input       sCPU0,
5           input       sCPU1,
6           input       sCPU2,
7           input       sCPU3,
8           input       sCPU4,
9           input       sCPU5,
10          input       sCPU6,
11          input       sCPU7,
12          input       [ 1: 0]selMIPS,
13          input       clock,
14          input       reset,
15          input       IF_Flush,
16          input       IF_Stall,
17          input       ID_Stall,
18          // Control Signals
19          input       [ 31: 0]IF_Instruction,
20          // Data Signals
21          input       [ 31: 0]IF_PCAdd4,
22          input       [ 31: 0]IF_PC,
23
24          input       IF_IsBDS,
25          // ------------------
26          output      [ 31: 0]ID_Instruction,
27          output      [ 31: 0]ID_PCAdd4,
28          output      [ 31: 0]ID_RestartPC,
29          output      ID_IsBDS,
30          output      ID_IsFlushed
31          );
32
33  sCPU_IFID_StageIFID0(
34          .if1(clock&sCPU0),
35          .if2(reset),
36          .if3(IF_Flush),
37          .if4(IF_Stall),
38          .if5(ID_Stall),
39          // Control Signals
40          .if6(IF_Instruction),
41          // Data Signals
42          .if7(IF_PCAdd4),
43          .if8(IF_PC),
44          .if9(IF_IsBDS),
45          // ------------------
46          .id1(ID_Instruction0),
47          .id2(ID_PCAdd40),
48          .id3(ID_RestartPC0),
49          .id4(ID_IsBDS0),
50          .id5(ID_IsFlushed0)
51          );
52
53          wire    [ 31: 0]ID_Instruction0;
54          wire    [ 31: 0]ID_PCAdd40;
55          wire    [ 31: 0]ID_RestartPC0;
56          wire    ID_IsBDS0;
57          wire    ID_IsFlushed0;
58
59  sCPU_IFID_StageIFID1(
60          .if1(clock&sCPU1),
61          .if2(reset),
62          .if3(IF_Flush),
63          .if4(IF_Stall),
64          .if5(ID_Stall),
65          // Control Signals
66          .if6(IF_Instruction),
67          // Data Signals
68          .if7(IF_PCAdd4),
69          .if8(IF_PC),
70          .if9(IF_IsBDS),
71          // ------------------
72          .id1(ID_Instruction1),
73          .id2(ID_PCAdd41),
74          .id3(ID_RestartPC1),
75          .id4(ID_IsBDS1),
76          .id5(ID_IsFlushed1)
77          );
78
79          wire    [ 31: 0]ID_Instruction1;
80          wire    [ 31: 0]ID_PCAdd41;
81          wire    [ 31: 0]ID_RestartPC1;
82          wire    ID_IsBDS1;
83          wire    ID_IsFlushed1;
84
85          sCPU_IFID_StageIFID2(
86          .if1(clock&sCPU2),
87          .if2(reset),
88          .if3(IF_Flush),
89          .if4(IF_Stall),
90          .if5(ID_Stall),
91          // Control Signals
92          .if6(IF_Instruction),
93          // Data Signals
94          .if7(IF_PCAdd4),
95          .if8(IF_PC),
96          .if9(IF_IsBDS),
97          // ------------------
98          .id1(ID_Instruction2),
99          .id2(ID_PCAdd42),
100         .id3(ID_RestartPC2),
101         .id4(ID_IsBDS2),
102         .id5(ID_IsFlushed2)
103         );
104
105         wire    [ 31: 0]ID_Instruction2;
106         wire    [ 31: 0]ID_PCAdd42;
107         wire    [ 31: 0]ID_RestartPC2;
108         wire    ID_IsBDS2;
109         wire    ID_IsFlushed2;
110
111 sCPU_IFID_StageIFID3(
112         .if1(clock&sCPU3),
113         .if2(reset),
114         .if3(IF_Flush),
115         .if4(IF_Stall),
116         .if5(ID_Stall),
117         // Control Signals
118         .if6(IF_Instruction),
119         // Data Signals
120         .if7(IF_PCAdd4),
121         .if8(IF_PC),
122         .if9(IF_IsBDS),
123         // ------------------
124         .id1(ID_Instruction3),
125         .id2(ID_PCAdd43),
126         .id3(ID_RestartPC3),
127         .id4(ID_IsBDS3),
128         .id5(ID_IsFlushed3)
129         );
130
131         wire    [ 31: 0]ID_Instruction3;
132         wire    [ 31: 0]ID_PCAdd43;
133         wire    [ 31: 0]ID_RestartPC3;
134         wire    ID_IsBDS3;
135         wire    ID_IsFlushed3;
136
137         sCPU_IFID_StageIFID4(
138         .if1(clock&sCPU4),
139         .if2(reset),
140         .if3(IF_Flush),
141         .if4(IF_Stall),
142         .if5(ID_Stall),
143         // Control Signals
144         .if6(IF_Instruction),
145         // Data Signals
146         .if7(IF_PCAdd4),
147         .if8(IF_PC),
148         .if9(IF_IsBDS),
149         // ------------------
150         .id1(ID_Instruction4),
151         .id2(ID_PCAdd44),
152         .id3(ID_RestartPC4),
153         .id4(ID_IsBDS4),
154         .id5(ID_IsFlushed4)
155         );
156
157         wire    [ 31: 0]ID_Instruction4;
158         wire    [ 31: 0]ID_PCAdd44;
159         wire    [ 31: 0]ID_RestartPC4;
160         wire    ID_IsBDS4;
161         wire    ID_IsFlushed4;
162
163         sCPU_IFID_StageIFID5(
164         .if1(clock&sCPU5),
165         .if2(reset),
166         .if3(IF_Flush),
167         .if4(IF_Stall),
168         .if5(ID_Stall),
169         // Control Signals
170         .if6(IF_Instruction),
171         // Data Signals
172         .if7(IF_PCAdd4),
173         .if8(IF_PC),
174         .if9(IF_IsBDS),
175         // ------------------
176         .id1(ID_Instruction5),
177         .id2(ID_PCAdd45),
178         .id3(ID_RestartPC5),
179         .id4(ID_IsBDS5),
180         .id5(ID_IsFlushed5)
181         );
182
183         wire    [ 31: 0]ID_Instruction5;
184         wire    [ 31: 0]ID_PCAdd45;
185         wire    [ 31: 0]ID_RestartPC5;
186         wire    ID_IsBDS5;
```

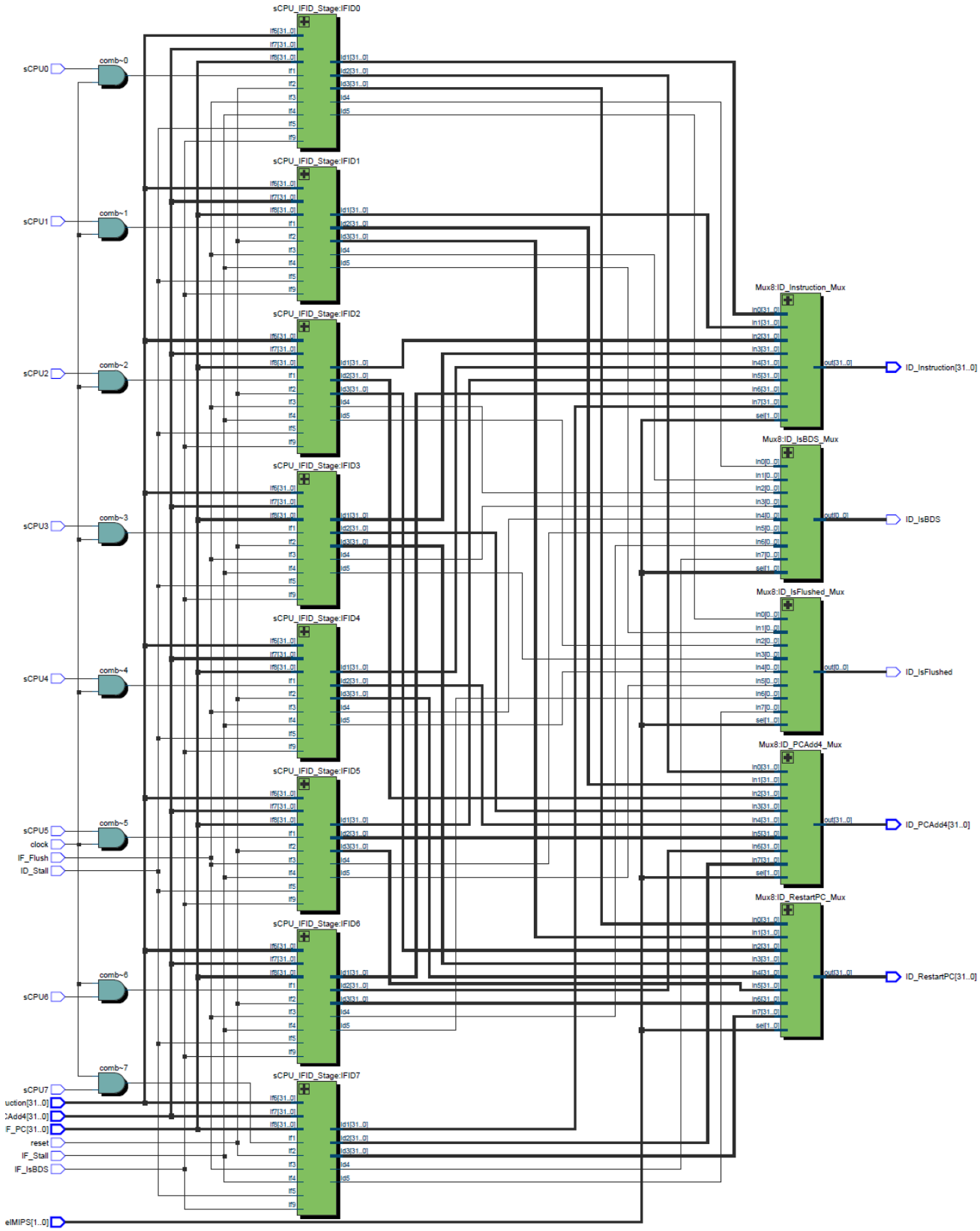
```
187         wire    ID_IsFlushed5;
188
189         sCPU_IFID_StageIFID6(
190         .if1(clock&sCPU6),
191         .if2(reset),
192         .if3(IF_Flush),
193         .if4(IF_Stall),
194         .if5(ID_Stall),
195         // Control Signals
196         .if6(IF_Instruction),
197         // Data Signals
198         .if7(IF_PCAdd4),
199         .if8(IF_PC),
200         .if9(IF_IsBDS),
201         // ----------------
202         .id1(ID_Instruction6),
203         .id2(ID_PCAdd46),
204         .id3(ID_RestartPC6),
205         .id4(ID_IsBDS6),
206         .id5(ID_IsFlushed6)
207         );
208
209         wire    [ 31: 0]ID_Instruction6;
210         wire  [ 31: 0]ID_PCAdd46;
211         wire  [ 31: 0]ID_RestartPC6;
212         wire    ID_IsBDS6;
213         wire    ID_IsFlushed6;
214
215         sCPU_IFID_StageIFID7(
216         .if1(clock&sCPU7),
217         .if2(reset),
218         .if3(IF_Flush),
219         .if4(IF_Stall),
220         .if5(ID_Stall),
221         // Control Signals
222         .if6(IF_Instruction),
223         // Data Signals
224         .if7(IF_PCAdd4),
225         .if8(IF_PC),
226         .if9(IF_IsBDS),
227         // ----------------
228         .id1(ID_Instruction7),
229         .id2(ID_PCAdd47),
230         .id3(ID_RestartPC7),
231         .id4(ID_IsBDS7),
232         .id5(ID_IsFlushed7)
233         );
234
235         wire    [ 31: 0]ID_Instruction7;
236         wire  [ 31: 0]ID_PCAdd47;
237         wire  [ 31: 0]ID_RestartPC7;
238         wire    ID_IsBDS7;
239         wire    ID_IsFlushed7;
240
241
242     Mux8#(.WIDTH(    32))ID_Instruction_Mux(
243                 .sel(selMIPS),
244                 .in0(ID_Instruction0),
245                 .in1(ID_Instruction1),
246                 .in2(ID_Instruction2),
247                 .in3(ID_Instruction3),
248                 .in4(ID_Instruction4),
249                 .in5(ID_Instruction5),
250                 .in6(ID_Instruction6),
251                 .in7(ID_Instruction7),
252                 .out(ID_Instruction)
253                 );
254
255     Mux8#(.WIDTH(    32))ID_PCAdd4_Mux(
256                 .sel(selMIPS),
257                 .in0(ID_PCAdd40),
258                 .in1(ID_PCAdd41),
259                 .in2(ID_PCAdd42),
260                 .in3(ID_PCAdd43),
261                 .in4(ID_PCAdd44),
262                 .in5(ID_PCAdd45),
263                 .in6(ID_PCAdd46),
264                 .in7(ID_PCAdd47),
265                 .out(ID_PCAdd4)
266                 );
267
268     Mux8#(.WIDTH(    32))ID_RestartPC_Mux(
269                 .sel(selMIPS),
270                 .in0(ID_RestartPC0),
271                 .in1(ID_RestartPC1),
272                 .in2(ID_RestartPC2),
273                 .in3(ID_RestartPC3),
274                 .in4(ID_RestartPC4),
275                 .in5(ID_RestartPC5),
276                 .in6(ID_RestartPC6),
277                 .in7(ID_RestartPC7),
278                 .out(ID_RestartPC)
279                 );
280
281     Mux8#(.WIDTH(    1))ID_IsBDS_Mux(
282                 .sel(selMIPS),
283                 .in0(ID_IsBDS0),
284                 .in1(ID_IsBDS1),
285                 .in2(ID_IsBDS2),
286                 .in3(ID_IsBDS3),
287                 .in4(ID_IsBDS4),
288                 .in5(ID_IsBDS5),
289                 .in6(ID_IsBDS6),
290                 .in7(ID_IsBDS7),
291                 .out(ID_IsBDS)
292                 );
293
294     Mux8#(.WIDTH(    1))ID_IsFlushed_Mux(
295                 .sel(selMIPS),
296                 .in0(ID_IsFlushed0),
297                 .in1(ID_IsFlushed1),
298                 .in2(ID_IsFlushed2),
299                 .in3(ID_IsFlushed3),
300                 .in4(ID_IsFlushed4),
301                 .in5(ID_IsFlushed5),
302                 .in6(ID_IsFlushed6),
303                 .in7(ID_IsFlushed7),
304                 .out(ID_IsFlushed)
305                 );
306
307     endmodule
308
```

Figură 2-32 Codul verilog pentru IFID8

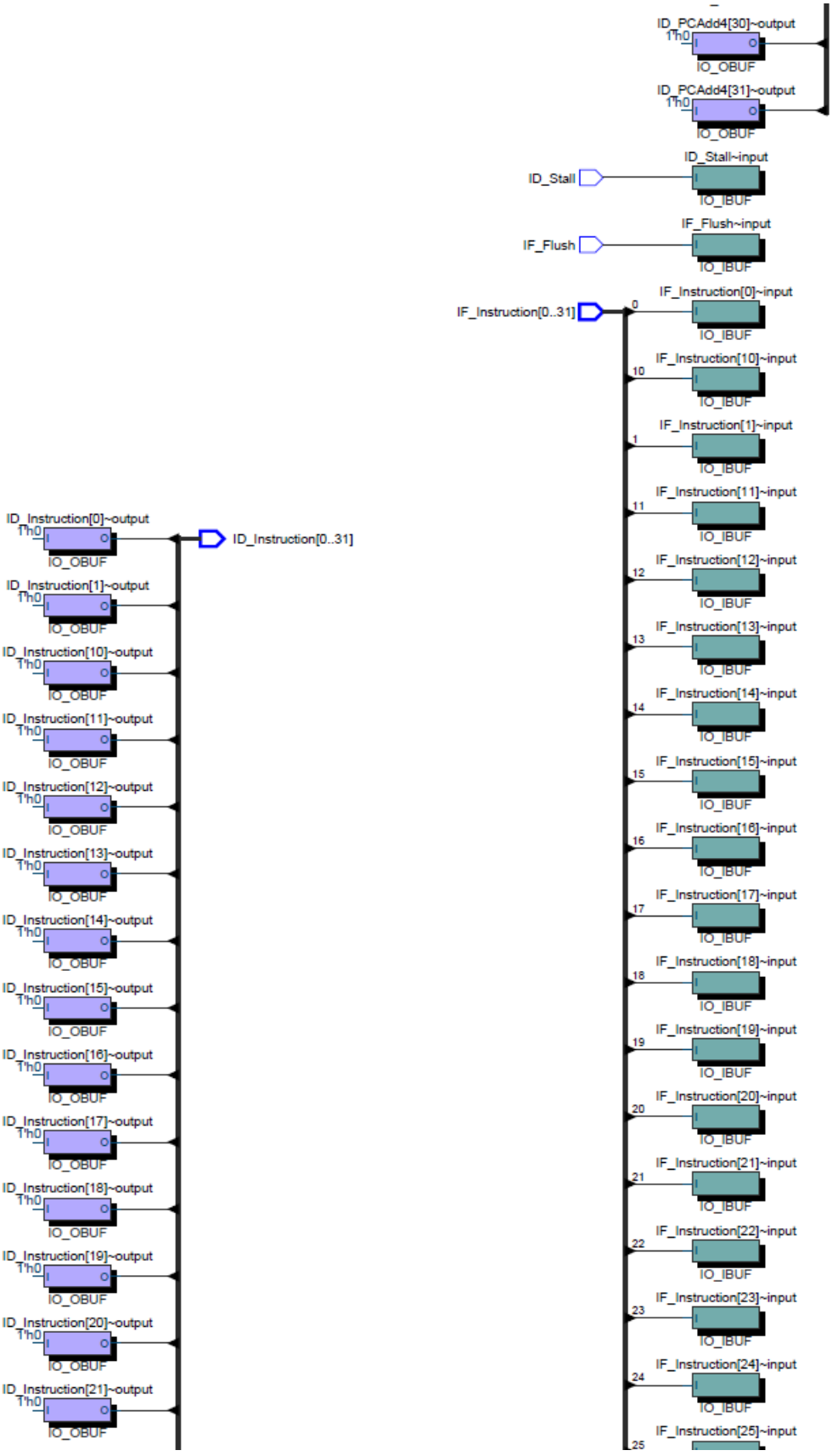
IFID8_Stage-Partition Statistics

Partition Statistics report for IFID8_Stage
Wed Sep 23 14:13:41 2015
Quartus II 64-Bit Version 15.0.0 Build 145 04/22/2015 SJ Web Edition

Fitter Partition Statistics		
Statistic	Top	hard_block:auto_generated_inst
Logic utilization (ALMs needed / total ALMs on device)	1 / 56480 (< 1 %)	0 / 56480 (0 %)
ALMs needed [=A-B+C]	1	0
[A] ALMs used in final placement [=a+b+c+d]	1 / 56480 (< 1 %)	0 / 56480 (0 %)
[a] ALMs used for LUT logic and registers	0	0
[b] ALMs used for LUT logic	1	0
[c] ALMs used for registers	0	0
[d] ALMs used for memory (up to half of total ALMs)	0	0
[B] Estimate of ALMs recoverable by dense packing	0 / 56480 (0 %)	0 / 56480 (0 %)
[C] Estimate of ALMs unavailable [=a+b+c+d]	0 / 56480 (0 %)	0 / 56480 (0 %)
[a] Due to location constrained logic	0	0
[b] Due to LAB-wide signal conflicts	0	0
[c] Due to LAB input limits	0	0
[d] Due to virtual I/Os	0	0
Difficulty packing design	Low	Low
Total LABs: partially or completely used	1 / 5648 (< 1 %)	0 / 5648 (0 %)
-- Logic LABs	1	0
-- Memory LABs (up to half of total LABs)	0	0
Combinational ALUT usage for logic	1	0
-- 7 input functions	0	0
-- 6 input functions	0	0
-- 5 input functions	0	0
-- 4 input functions	0	0
-- <=3 input functions	1	0
Combinational ALUT usage for route-throughs	0	0
Memory ALUT usage	0	0
-- 64-address deep	0	0
-- 32-address deep	0	0
Dedicated logic registers	0	0
-- By type:		
-- Primary logic registers	0 / 112960 (0 %)	0 / 112960 (0 %)
-- Secondary logic registers	0 / 112960 (0 %)	0 / 112960 (0 %)
-- By function:		
-- Design implementation registers	0	0
-- Routing optimization registers	0	0
Virtual pins	0	0
I/O pins	210	0
I/O registers	0	0
Total block memory bits	0	0
Total block memory implementation bits	0	0
Connections		
-- Input Connections	0	0
-- Registered Input Connections	0	0
-- Output Connections	0	0
-- Registered Output Connections	0	0
Internal Connections		
-- Total Connections	210	0
-- Registered Connections	0	0
External Connections		
-- Top	0	0
-- hard_block:auto_generated_inst	0	0
Partition Interface		
-- Input Ports	112	0
-- Output Ports	98	0
-- Bidir Ports	0	0
Registered Ports		
-- Registered Input Ports	0	0
-- Registered Output Ports	0	0
Port Connectivity		
-- Input Ports driven by GND	0	0
-- Output Ports driven by GND	0	0
-- Input Ports driven by VCC	0	0
-- Output Ports driven by VCC	0	0
-- Input Ports with no Source	0	0
-- Output Ports with no Source	0	0
-- Input Ports with no Fanout	0	0
-- Output Ports with no Fanout	0	0

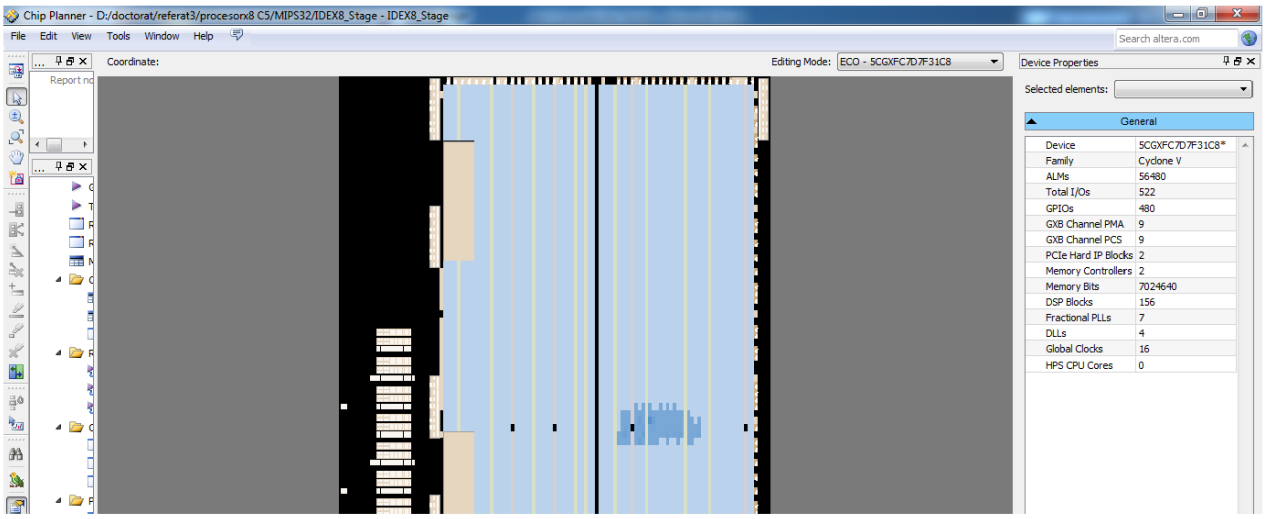


Figură 2-33 RTL IFID8



Figură 2-34 Parţial Tehnology map IFID8

2.3.2. IDEX8



Figură 2-35 Chip Planner IDEX8

IDEX8_Stage-Flow Summary

Flow Summary report for IDEX8_Stage
Wed Sep 23 14:03:12 2015
Quartus II 64-Bit Version 15.0.0 Build 145 04/22/2015 SJ Web Edition

Flow Summary	
Flow Status	Successful - Mon Sep 21 23:18:22 2015
Quartus II 64-Bit Version	15.0.0 Build 145 04/22/2015 SJ Web Edition
Revision Name	IDEX8_Stage
Top-level Entity Name	IDEX8_Stage
Family	Cyclone V
Device	5CGXFC7D7F31C8
Timing Models	Final
Logic utilization (in ALMs)	286 / 56,480 (< 1 %)
Total registers	616
Total pins	349 / 522 (67 %)
Total virtual pins	0
Total block memory bits	0 / 7,024,640 (0 %)
Total DSP Blocks	0 / 156 (0 %)
Total HSSI RX PCSs	0 / 9 (0 %)
Total HSSI PMA RX Deserializers	0 / 9 (0 %)
Total HSSI TX PCSs	0 / 9 (0 %)
Total HSSI PMA TX Serializers	0 / 9 (0 %)
Total PLLs	0 / 16 (0 %)
Total DLLs	0 / 4 (0 %)

Multiplexer Restructuring Statistics (Restructuring Performed) report for IDEX8_Stage
Wed Sep 23 14:03:50 2015
Quartus II 64-Bit Version 15.0.0 Build 145 04/22/2015 SJ Web Edition

Multiplexer Restructuring Statistics (Restructuring Performed)

Multiplexer Inputs	Bus Width	Baseline Area	Area if Restructured	Saving if Restructured	Registered	Example Multiplexer Output
3:1	616 bits	1232 LEs	0 LEs	1232 LEs	Yes	|IDEX8_Stage|sCPU_IDEX_Stage:IDEXdoi|ex7
4:1	153 bits	306 LEs	306 LEs	0 LEs	No	|IDEX8_Stage|Mux8:EX_MemSignExtend_Mux|Mux0

```
1   module  IDEX8_Stage(
2           input       sCPU0,
3           input       sCPU1,
4           input       sCPU2,
5           input       sCPU3,
6           input       sCPU4,
7           input       sCPU5,
8           input       sCPU6,
9           input       sCPU7,
10          input       [ 1: 0]selMIPS,
11          input       clock,
12          input       reset,
13          input       ID_Flush,
14          input       ID_Stall,
15          input       EX_Stall,
16          // Control Signals
17          input       ID_Link,
18          input       ID_RegDst,
19          input       ID_ALUSrcImm,
20          input       [ 4: 0]ID_ALUOp,
21          input       ID_Movn,
22          input       ID_Movz,
23          input       ID_LLSC,
24          input       ID_MemRead,
25          input       ID_MemWrite,
26          input       ID_MemByte,
27          input       ID_MemHalf,
28          input       ID_MemSignExtend,
29          input       ID_Left,
30          input       ID_Right,
31          input       ID_RegWrite,
32          input       ID_MemtoReg,
33          input       ID_ReverseEndian,
34          // Hazard & Forwarding
35          input       [ 4: 0]ID_Rs,
36          input       [ 4: 0]ID_Rt,
37          input       ID_WantRsByEX,
38          input       ID_NeedRsByEX,
39          input       ID_WantRtByEX,
40          input       ID_NeedRtByEX,
41          // Exception Control/Info
42          input       ID_KernelMode,
43          input       [ 31: 0]ID_RestartPC,
44          input       ID_IsBDS,
45          input       ID_Trap,
46          input       ID_TrapCond,
47          input       ID_EX_CanErr,
48          input       ID_M_CanErr,
49          // Data Signals
50          input       [ 31: 0]ID_ReadData1,
51          input       [ 31: 0]ID_ReadData2,
52          input       [ 16: 0]ID_SignExtImm,
53          // ---------------
54          output      EX_Link,
55          output      [ 1: 0]EX_LinkRegDst,
56          output      EX_ALUSrcImm,
57          output      [ 4: 0]EX_ALUOp,
58          output      EX_Movn,
59          output      EX_Movz,
60          output      EX_LLSC,
61          output      EX_MemRead,
62          output      EX_MemWrite,
63          output      EX_MemByte,
64          output      EX_MemHalf,
65          output      EX_MemSignExtend,
66          output      EX_Left,
67          output      EX_Right,
68          output      EX_RegWrite,
69          output      EX_MemtoReg,
70          output      EX_ReverseEndian,
71          output      [ 4: 0]EX_Rs,
72          output      [ 4: 0]EX_Rt,
73          output      EX_WantRsByEX,
74          output      EX_NeedRsByEX,
75          output      EX_WantRtByEX,
76          output      EX_NeedRtByEX,
77          output      EX_KernelMode,
78          output      [ 31: 0]EX_RestartPC,
79          output      EX_IsBDS,
80          output      EX_Trap,
81          output      EX_TrapCond,
82          output      EX_EX_CanErr,
83          output      EX_M_CanErr,
84          output      [ 31: 0]EX_ReadData1,
85          output      [ 31: 0]EX_ReadData2,
86          output      [ 31: 0]EX_SignExtImm,
87          output      [ 4: 0]EX_Rd,
88          output      [ 4: 0]EX_Shamt
89          );
90
91
92
93  sCPU_IDEX_StageIDEXzero(
94          .id1(clock&sCPU0),
95          .id2(reset),
96          .id3(ID_Flush),
97          .id4(ID_Stall),
98          .id5(EX_Stall),
99          // Control Signals
100         .id6(ID_Link),
101         .id7(ID_RegDst),
102         .id8(ID_ALUSrcImm),
103         .id9(ID_ALUOp),
104         .id10(ID_Movn),
105         .id11(ID_Movz),
106         .id12(ID_LLSC),
107         .id13(ID_MemRead),
108         .id14(ID_MemWrite),
109         .id15(ID_MemByte),
110         .id16(ID_MemHalf),
111         .id17(ID_MemSignExtend),
112         .id18(ID_Left),
113         .id19(ID_Right),
114         .id20(ID_RegWrite),
115         .id21(ID_MemtoReg),
116         .id22(ID_ReverseEndian),
117         // Hazard & Forwarding
118         .id23(ID_Rs),
119         .id24(ID_Rt),
120         .id25(ID_WantRsByEX),
121         .id26(ID_NeedRsByEX),
122         .id27(ID_WantRtByEX),
123         .id28(ID_NeedRtByEX),
124         // Exception Control/Info
```

Figură 2-36 Parţial codul verilog pentru IDEX8

IDEX8_Stage-Routing Usage Summary

Routing Usage Summary report for IDEX8_Stage
Wed Sep 23 14:05:23 2015
Quartus II 64-Bit Version 15.0.0 Build 145 04/22/2015 SJ Web Edition

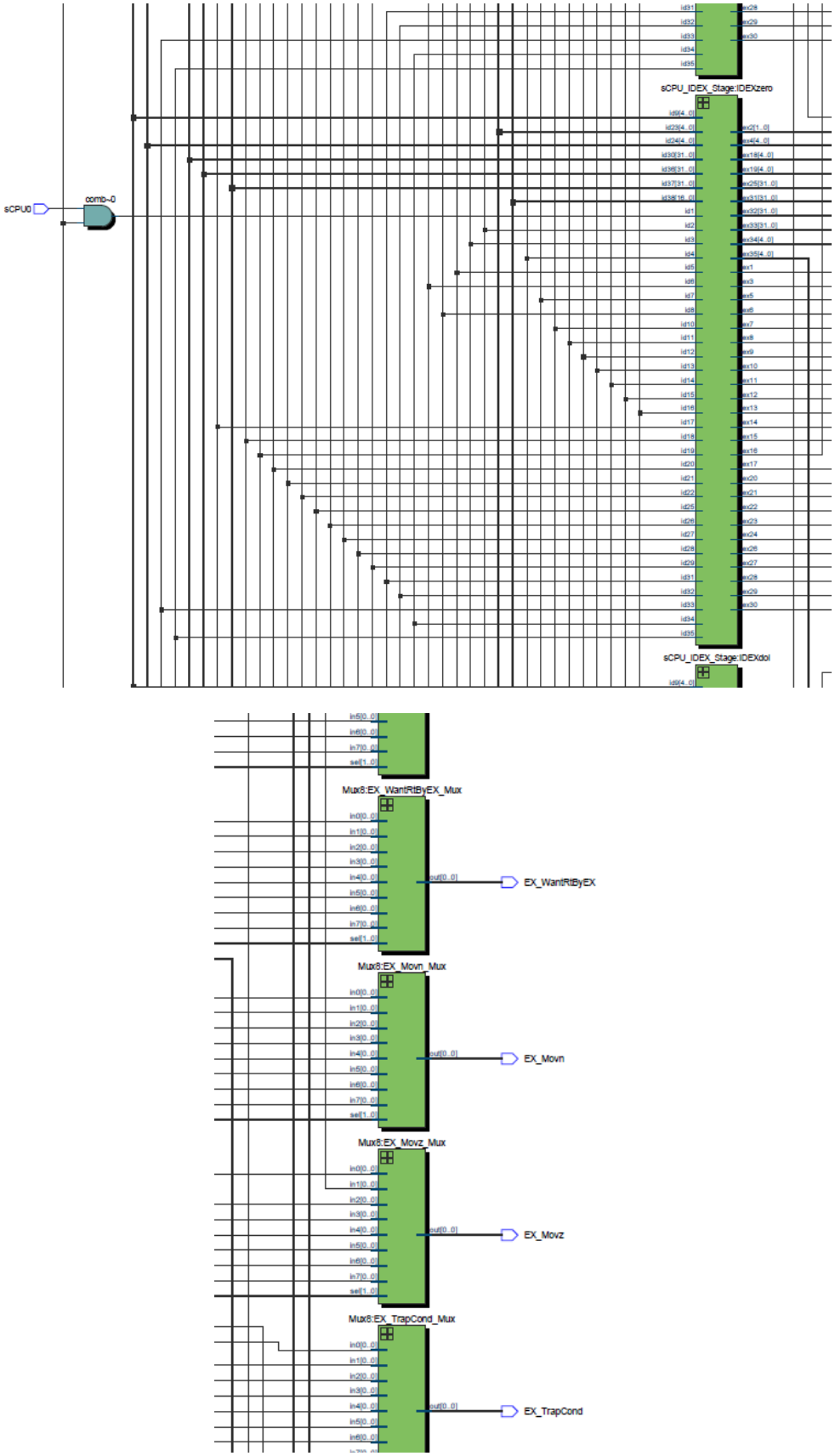
Routing Usage Summary	
Routing Resource Type	Usage
Block interconnects	1,671 / 374,484 (< 1 %)
C12 interconnects	453 / 16,664 (3 %)
C2 interconnects	639 / 155,012 (< 1 %)
C4 interconnects	1,246 / 72,600 (2 %)
DQS bus muxes	0 / 30 (0 %)
DQS-18 I/O buses	0 / 30 (0 %)
DQS-9 I/O buses	0 / 30 (0 %)
Direct links	43 / 374,484 (< 1 %)
Global clocks	0 / 16 (0 %)
Horizontal periphery clocks	0 / 72 (0 %)
Local interconnects	10 / 112,960 (< 1 %)
Quadrant clocks	0 / 88 (0 %)
R14 interconnects	378 / 15,868 (2 %)
R14/C12 interconnect drivers	694 / 27,256 (3 %)
R3 interconnects	851 / 169,296 (< 1 %)
R6 interconnects	1,337 / 330,800 (< 1 %)
Spine clocks	0 / 480 (0 %)
Wire stub REs	0 / 20,834 (0 %)

IDEX8_Stage-Post-Synthesis Netlist Statistics for Top Partition

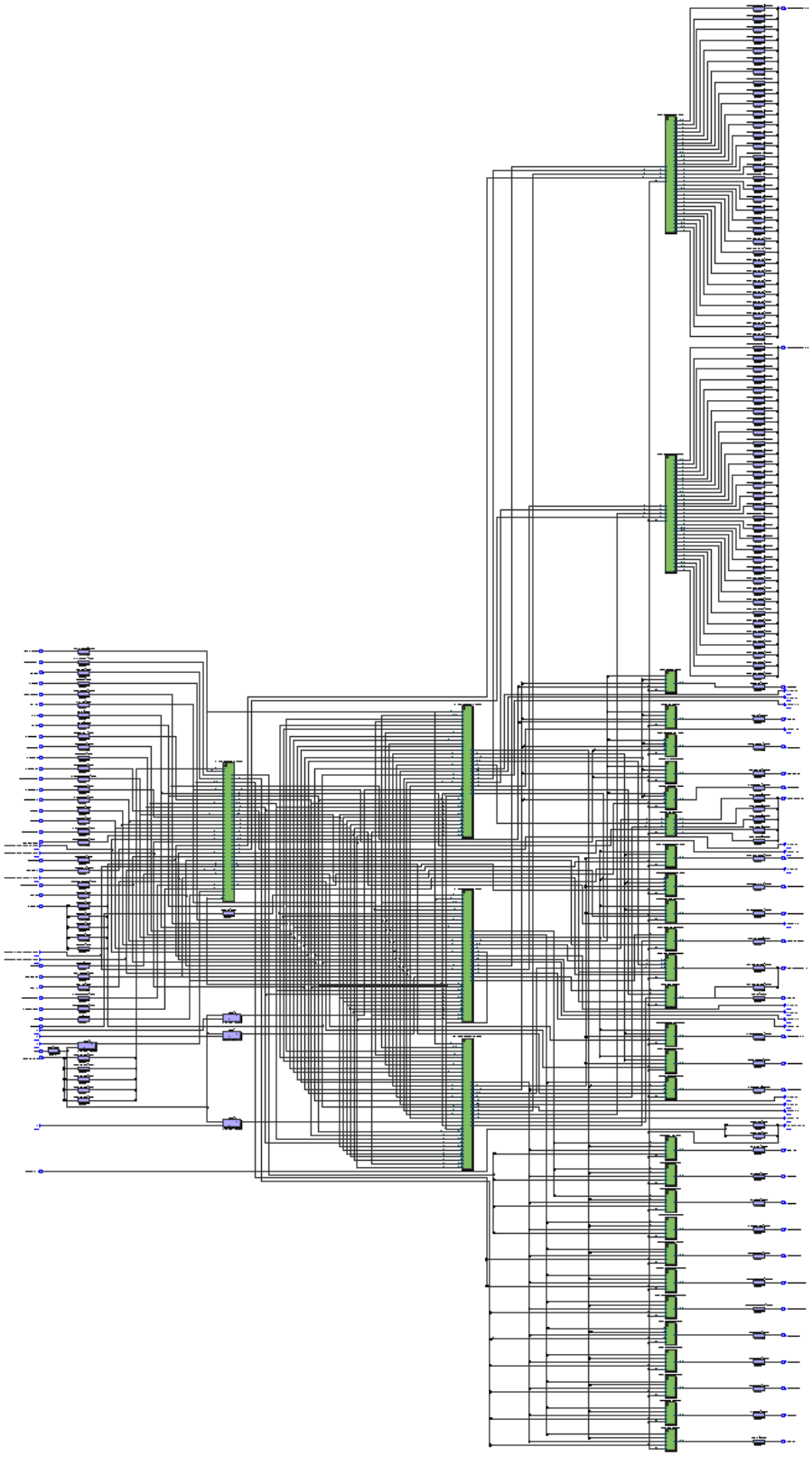
Post-Synthesis Netlist Statistics for Top Partition report for IDEX8_Stage
Wed Sep 23 14:04:04 2015
Quartus II 64-Bit Version 15.0.0 Build 145 04/22/2015 SJ Web Edition

Post-Synthesis Netlist Statistics for Top Partition	
Type	Count
arriav_ff	616
ENA SCLR	616
arriav_lcell_comb	177
extend	1
7 data inputs	1
normal	176
2 data inputs	8
3 data inputs	15
6 data inputs	153
boundary_port	349
Max LUT depth	2.00
Average LUT depth	0.87

PowerPlay Power Analyzer Status	Successful - Sat Sep 26 21:50:05 2015
Quartus II 64-Bit Version	15.0.0 Build 145 04/22/2015 SJ Web Edition
Revision Name	IDEX8_Stage
Top-level Entity Name	IDEX8_Stage
Family	Cyclone V
Device	5CGXFC7D7F31C8
Power Models	Final
Total Thermal Power Dissipation	368.56 mW
Core Dynamic Thermal Power Dissipation	0.00 mW
Core Static Thermal Power Dissipation	348.97 mW
I/O Thermal Power Dissipation	19.59 mW
Power Estimation Confidence	Low: user provided insufficient toggle rate data

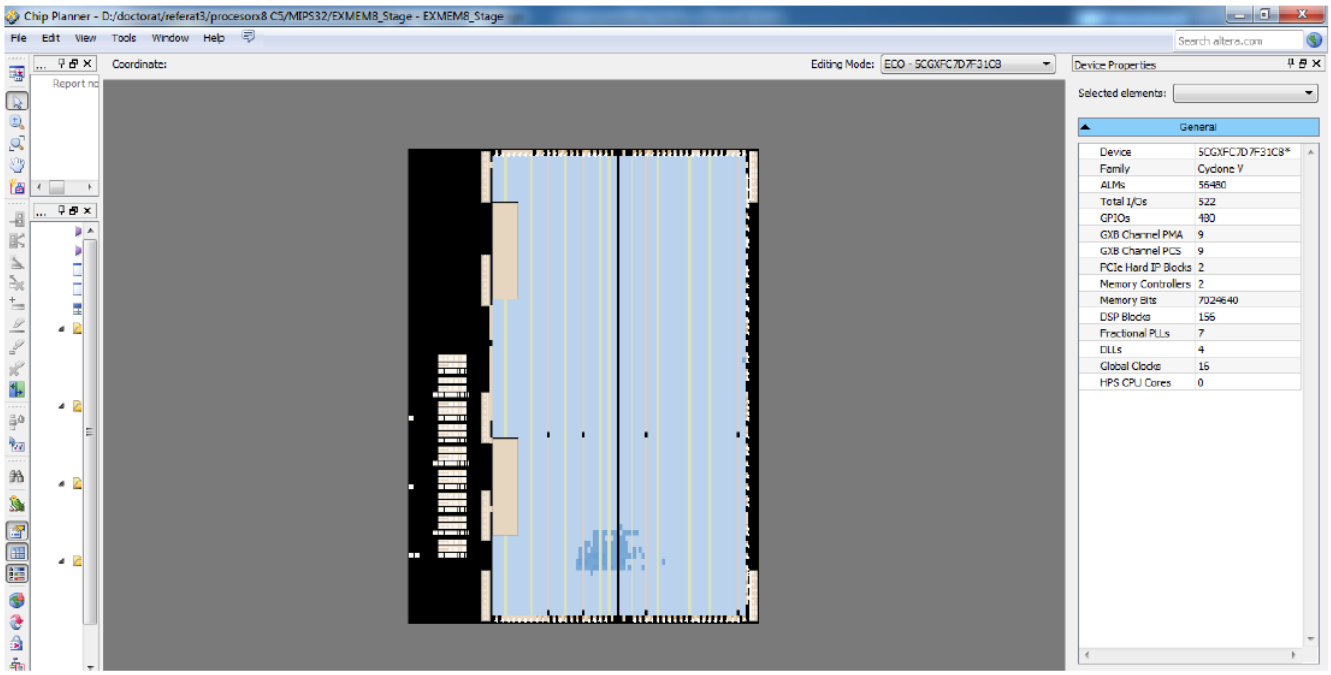


Figură 2-37 Parţial RTL IDEX8



Figură 2-38 Parțial Tehnology map IDEX8

2.3.3. EXMEM8



Figură 2-39 Chip planner EXMEM8

EXMEM8_Stage-Flow Summary

Flow Summary report for EXMEM8_Stage
Wed Sep 23 13:54:32 2015
Quartus II 64-Bit Version 15.0.0 Build 145 04/22/2015 SJ Web Edition

Flow Summary	
Flow Status	Successful - Mon Sep 21 23:12:32 2015
Quartus II 64-Bit Version	15.0.0 Build 145 04/22/2015 SJ Web Edition
Revision Name	EXMEM8_Stage
Top-level Entity Name	EXMEM8_Stage
Family	Cyclone V
Device	5CGXFC7D7F31C8
Timing Models	Final
Logic utilization (in ALMs)	216 / 56,480 (< 1 %)
Total registers	468
Total pins	252 / 522 (48 %)
Total virtual pins	0
Total block memory bits	0 / 7,024,640 (0 %)
Total DSP Blocks	0 / 156 (0 %)
Total HSSI RX PCSs	0 / 9 (0 %)
Total HSSI PMA RX Deserializers	0 / 9 (0 %)
Total HSSI TX PCSs	0 / 9 (0 %)
Total HSSI PMA TX Serializers	0 / 9 (0 %)
Total PLLs	0 / 16 (0 %)
Total DLLs	0 / 4 (0 %)

Multiplexer Restructuring Statistics (Restructuring Performed) report for EXMEM8_Stage
Wed Sep 23 13:55:12 2015
Quartus II 64-Bit Version 15.0.0 Build 145 04/22/2015 SJ Web Edition

Multiplexer Restructuring Statistics (Restructuring Performed)

Multiplexer Inputs	Bus Width	Baseline Area	Area if Restructured	Saving if Restructured	Registered	Example Multiplexer Output
3:1	468 bits	936 LEs	0 LEs	936 LEs	Yes	|EXMEM8_Stage|sCPU_EXMEM_Stage:exmem1|mem8
4:1	117 bits	234 LEs	234 LEs	0 LEs	No	|EXMEM8_Stage|Mux8:M_MemWrite_Mux|Mux0

```
1   module  EXMEM8_Stage(
2           input       sCPU0,
3           input       sCPU1,
4           input       sCPU2,
5           input       sCPU3,
6           input       sCPU4,
7           input       sCPU5,
8           input       sCPU6,
9           input       sCPU7,
10          input       [ 1: 0]selMIPS,
11          input       clock,
12          input       reset,
13          input       EX_Flush,
14          input       EX_Stall,
15          input       M_Stall,
16          // Control Signals
17          input       EX_Movn,
18          input       EX_Movz,
19          input       EX_BZero,
20          input       EX_RegWrite,      // Future Control to WB
21          input       EX_MemtoReg,      // Future Control to WB
22          input       EX_ReverseEndian,
23          input       EX_LLSC,
24          input       EX_MemRead,
25          input       EX_MemWrite,
26          input       EX_MemByte,
27          input       EX_MemHalf,
28          input       EX_MemSignExtend,
29          input       EX_Left,
30          input       EX_Right,
31          // Exception Control/Info
32          input       EX_KernelMode,
33          input       [ 31: 0]EX_RestartPC,
34          input       EX_IsBDS,
35          input       EX_Trap,
36          input       EX_TrapCond,
37          input       EX_M_CanErr,
38          // Data Signals
39          input       [ 31: 0]EX_ALU_Result,
40          input       [ 31: 0]EX_ReadData2,
41          input       [ 4: 0]EX_RtRd,
42          // ----------------
43          output      M_RegWrite,
44          output      M_MemtoReg,
45          output      M_ReverseEndian,
46          output      M_LLSC,
47          output      M_MemRead,
48          output      M_MemWrite,
49          output      M_MemByte,
50          output      M_MemHalf,
51          output      M_MemSignExtend,
52          output      M_Left,
53          output      M_Right,
54          output      M_KernelMode,
55          output      [ 31: 0]M_RestartPC,
56          output      M_IsBDS,
57          output      M_Trap,
58          output      M_TrapCond,
59          output      M_M_CanErr,
60          output      [ 31: 0]M_ALU_Result,
61          output      [ 31: 0]M_ReadData2,
62          output      [ 4: 0]M_RtRd
63          );
64
65  sCPU_EXMEM_Stageexmem0(
66          .ex1(clock&sCPU0),
67          .ex2(reset),
68          .ex3(EX_Flush),
69          .ex4(EX_Stall),
70          .ex5(M_Stall),
71          // Control Signals
72          .ex6(EX_Movn),
73          .ex7(EX_Movz),
74          .ex8(EX_BZero),
75          .ex9(EX_RegWrite),          // Future Control to WB
76          .ex10(EX_MemtoReg),         // Future Control to WB
77          .ex11(EX_ReverseEndian),
78          .ex12(EX_LLSC),
79          .ex13(EX_MemRead),
80          .ex14(EX_MemWrite),
81          .ex15(EX_MemByte),
82          .ex16(EX_MemHalf),
83          .ex17(EX_MemSignExtend),
84          .ex18(EX_Left),
85          .ex19(EX_Right),
86          // Exception Control/Info
87          .ex20(EX_KernelMode),
88          .ex21(EX_RestartPC),
89          .ex22(EX_IsBDS),
90          .ex23(EX_Trap),
91          .ex24(EX_TrapCond),
92          .ex25(EX_M_CanErr),
93          // Data Signals
94          .ex26(EX_ALU_Result),
95          .ex27(EX_ReadData2),
96          .ex28(EX_RtRd),
97          // ----------------
98          .mem1(M_RegWrite0),
99          .mem2(M_MemtoReg0),
100         .mem3(M_ReverseEndian0),
101         .mem4(M_LLSC0),
102         .mem5(M_MemRead0),
103         .mem6(M_MemWrite0),
104         .mem7(M_MemByte0),
105         .mem8(M_MemHalf0),
106         .mem9(M_MemSignExtend0),
107         .mem10(M_Left0),
108         .mem11(M_Right0),
109         .mem12(M_KernelMode0),
110         .mem13(M_RestartPC0),
111         .mem14(M_IsBDS0),
112         .mem15(M_Trap0),
113         .mem16(M_TrapCond0),
114         .mem17(M_M_CanErr0),
115         .mem18(M_ALU_Result0),
116         .mem19(M_ReadData20),
117         .mem20(M_RtRd0)
118         );
119
120         wire    M_RegWrite0;
121         wire    M_MemtoReg0;
122         wire    M_ReverseEndian0;
123         wire    M_LLSC0;
124         wire    M_MemRead0;
```

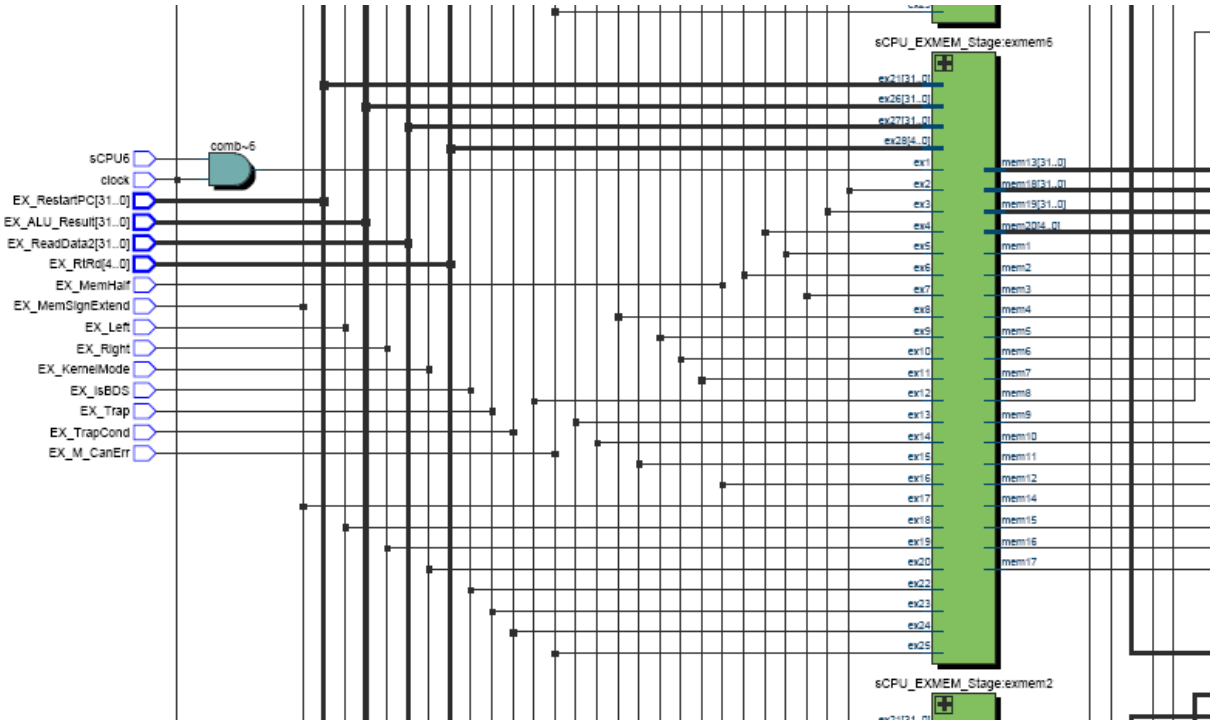
Figură 2-40 Parţial codul verilog pentru EXMEM8

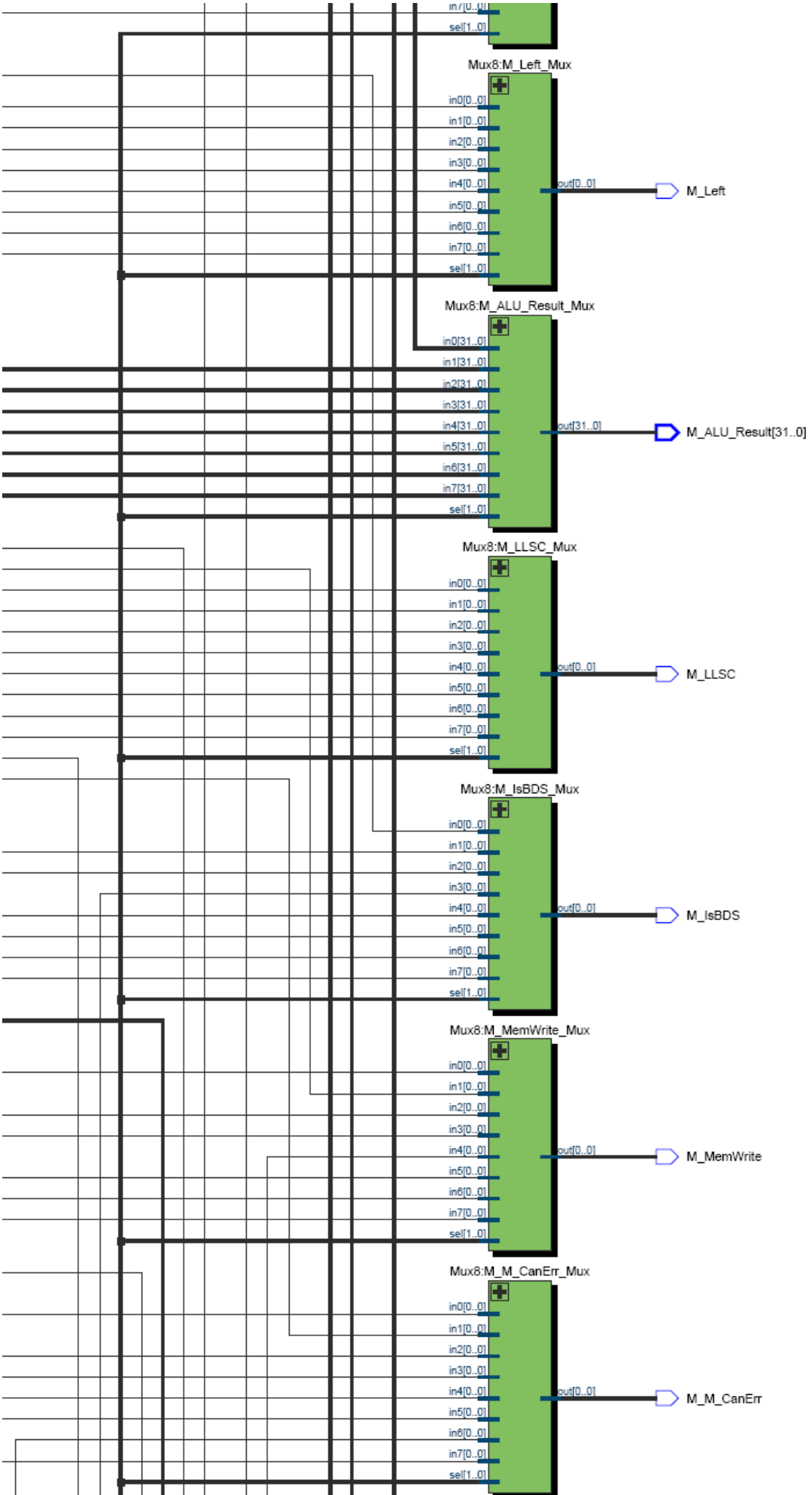
EXMEM8_Stage-Routing Usage Summary

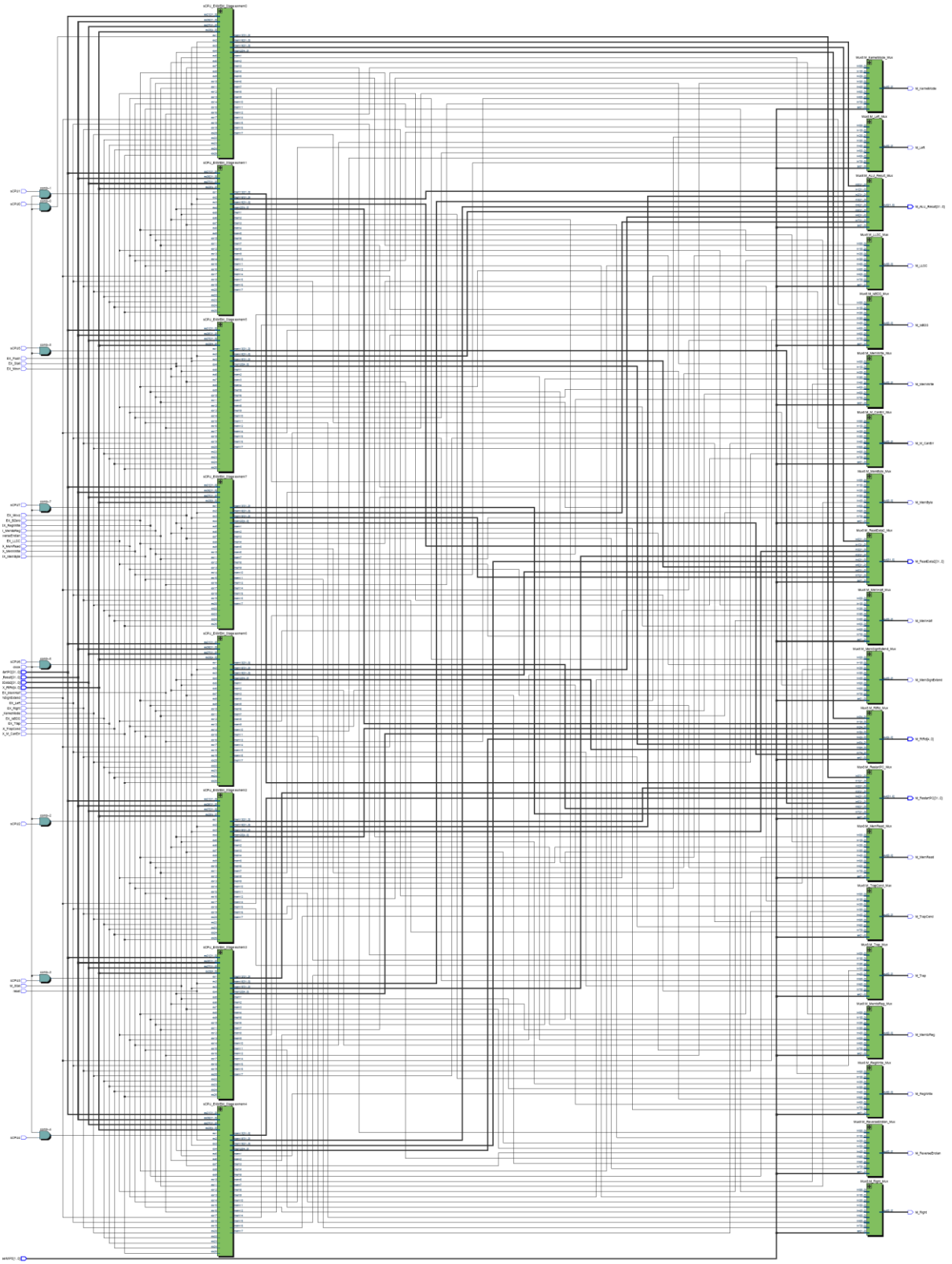
Routing Usage Summary report for EXMEM8_Stage
Wed Sep 23 13:56:49 2015
Quartus II 64-Bit Version 15.0.0 Build 145 04/22/2015 SJ Web Edition

Routing Usage Summary	
Routing Resource Type	Usage
Block interconnects	1,276 / 374,484 (< 1 %)
C12 interconnects	416 / 16,664 (2 %)
C2 interconnects	498 / 155,012 (< 1 %)
C4 interconnects	541 / 72,600 (< 1 %)
DQS bus muxes	0 / 30 (0 %)
DQS-18 I/O buses	0 / 30 (0 %)
DQS-9 I/O buses	0 / 30 (0 %)
Direct links	23 / 374,484 (< 1 %)
Global clocks	0 / 16 (0 %)
Horizontal periphery clocks	0 / 72 (0 %)
Local interconnects	8 / 112,960 (< 1 %)
Quadrant clocks	0 / 88 (0 %)
R14 interconnects	414 / 15,868 (3 %)
R14/C12 interconnect drivers	713 / 27,256 (3 %)
R3 interconnects	736 / 169,296 (< 1 %)
R6 interconnects	1,052 / 330,800 (< 1 %)
Spine clocks	0 / 480 (0 %)
Wire stub REs	0 / 20,834 (0 %)

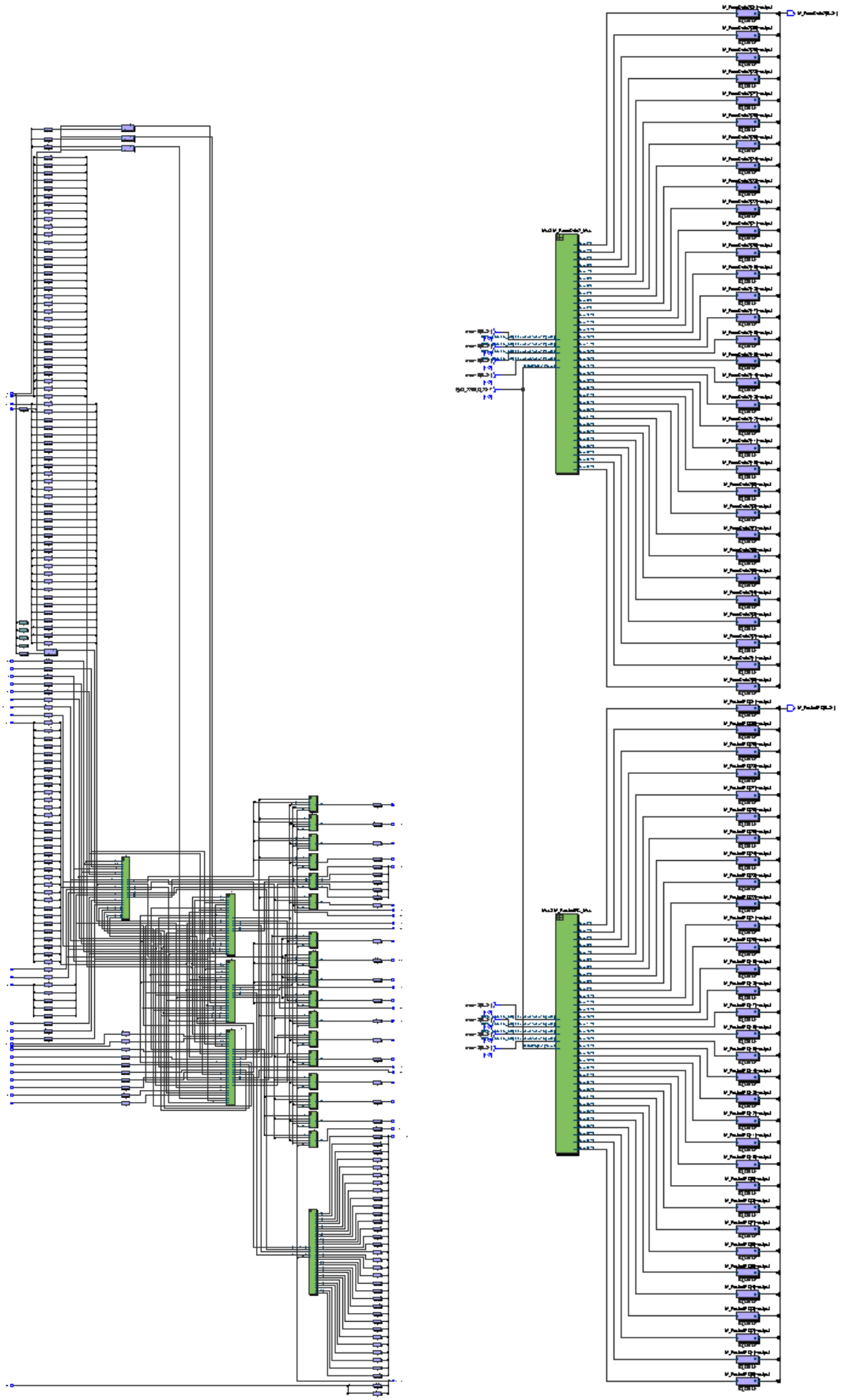
PowerPlay Power Analyzer Status	Successful - Sat Sep 26 21:43:24 2015
Quartus II 64-Bit Version	15.0.0 Build 145 04/22/2015 SJ Web Edition
Revision Name	EXMEM8_Stage
Top-level Entity Name	EXMEM8_Stage
Family	Cyclone V
Device	5CGXFC7D7F31C8
Power Models	Final
Total Thermal Power Dissipation	365.17 mW
Core Dynamic Thermal Power Dissipation	0.00 mW
Core Static Thermal Power Dissipation	348.96 mW
I/O Thermal Power Dissipation	16.22 mW
Power Estimation Confidence	Low: user provided insufficient toggle rate data

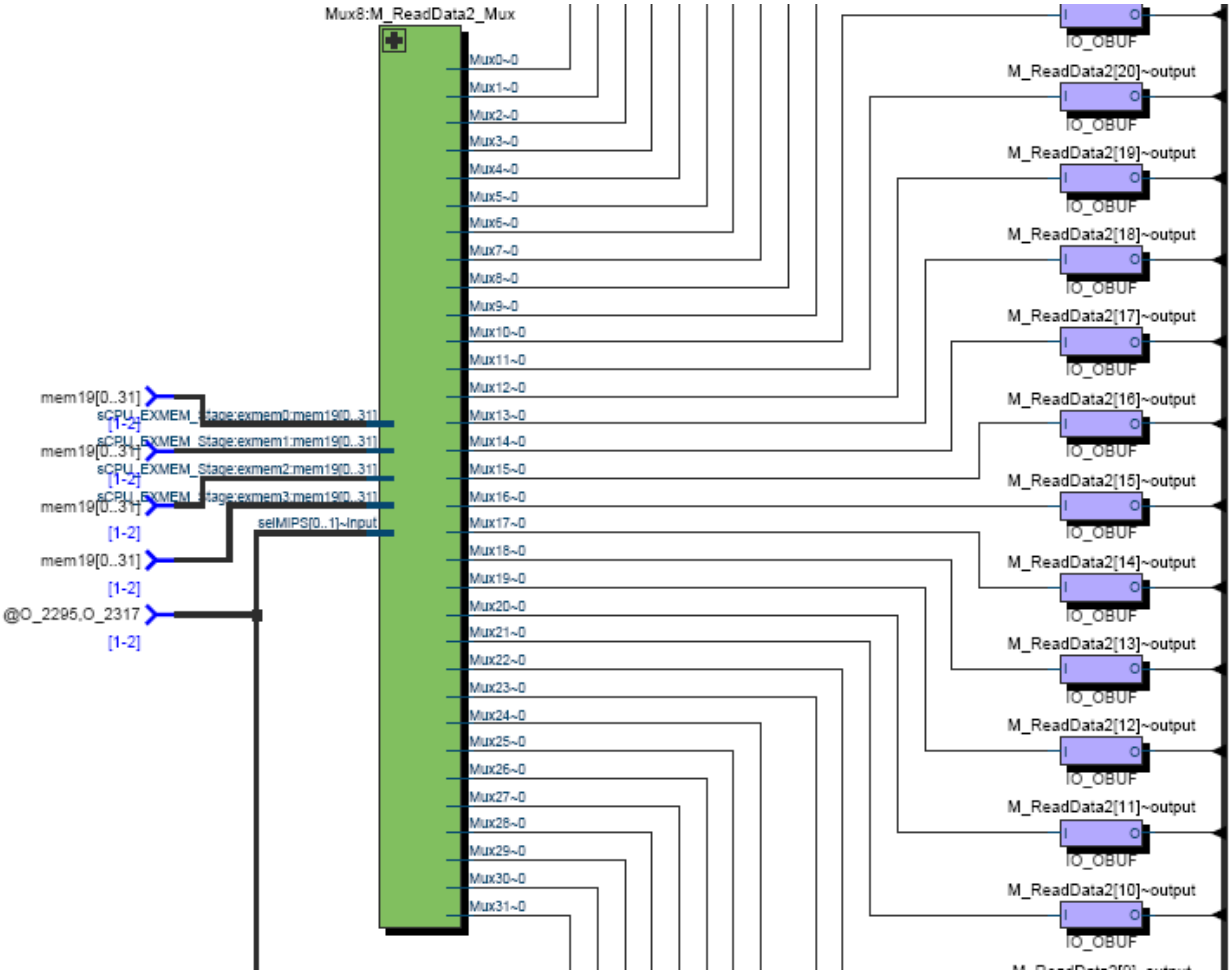



sel[1..0]
Mux8:M_Left_Mux
in0[0..0]
in1[0..0]
in2[0..0]
in3[0..0]
in4[0..0]
in5[0..0]
in6[0..0]
in7[0..0]
sel[1..0]
out[0..0]
M_Left
Mux8:M_ALU_Result_Mux
in0[31..0]
in1[31..0]
in2[31..0]
in3[31..0]
in4[31..0]
in5[31..0]
in6[31..0]
in7[31..0]
sel[1..0]
out[31..0]
M_ALU_Result[31..0]
Mux8:M_LLSC_Mux
in0[0..0]
in1[0..0]
in2[0..0]
in3[0..0]
in4[0..0]
in5[0..0]
in6[0..0]
in7[0..0]
sel[1..0]
out[0..0]
M_LLSC
Mux8:M_IsBDS_Mux
in0[0..0]
in1[0..0]
in2[0..0]
in3[0..0]
in4[0..0]
in5[0..0]
in6[0..0]
in7[0..0]
sel[1..0]
out[0..0]
M_IsBDS
Mux8:M_MemWrite_Mux
in0[0..0]
in1[0..0]
in2[0..0]
in3[0..0]
in4[0..0]
in5[0..0]
in6[0..0]
in7[0..0]
sel[1..0]
out[0..0]
M_MemWrite
Mux8:M_M_CanErr_Mux
in0[0..0]
in1[0..0]
in2[0..0]
in3[0..0]
in4[0..0]
in5[0..0]
in6[0..0]
in7[0..0]
sel[1..0]
out[0..0]
M_M_CanErr



Figură 2-41 RTL EXMEM8





Figură 2-42 Tehnology map EXMEM8

2.3.4. MEMWB8

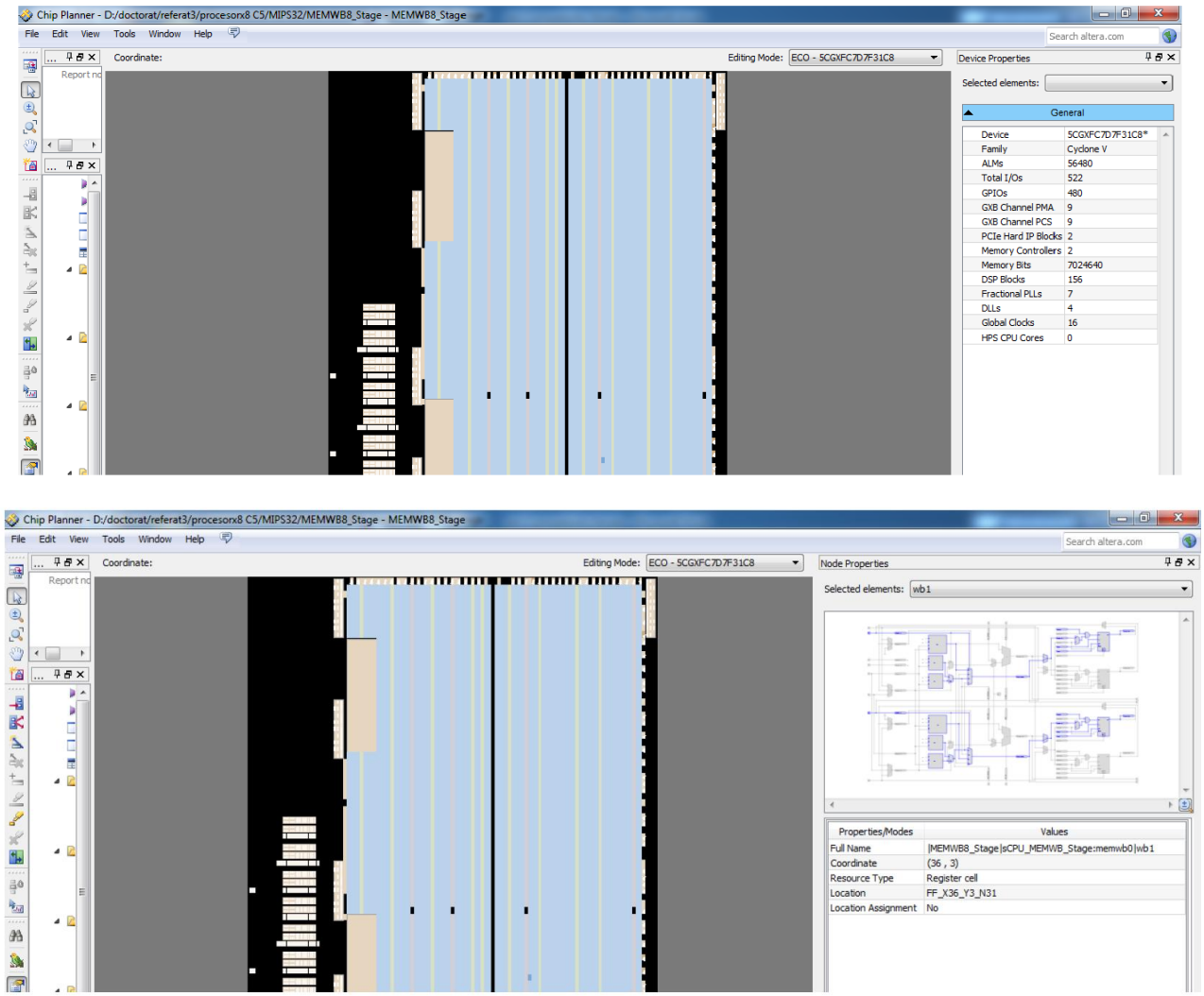
PowerPlay Power Analyzer Status	Successful - Sat Sep 26 21:59:35 2015
Quartus II 64-Bit Version	15.0.0 Build 145 04/22/2015 SJ Web Edition
Revision Name	MEMWB8_Stage
Top-level Entity Name	MEMWB8_Stage
Family	Cyclone V
Device	5CGXFC7D7F31C8
Power Models	Final
Total Thermal Power Dissipation	360.70 mW
Core Dynamic Thermal Power Dissipation	0.00 mW
Core Static Thermal Power Dissipation	348.94 mW
I/O Thermal Power Dissipation	11.77 mW
Power Estimation Confidence	Low: user provided insufficient toggle rate data

```
Multiplexer Restructuring Statistics (Restructuring Performed) report for MEMWB8_Stage
Wed Sep 23 14:20:19 2015
Quartus II 64-Bit Version 15.0.0 Build 145 04/22/2015 SJ Web Edition

+-----------------------------------------------------------------------------------------------------------------------------------------------------------+
; Multiplexer Restructuring Statistics (Restructuring Performed)                                                                                            ;
+--------------------+-----------+--------------+---------------------+----------------------+------------+-----------------------------------------------+
; Multiplexer Inputs ; Bus Width ; Baseline Area ; Area if Restructured ; Saving if Restructured ; Registered ; Example Multiplexer Output                    ;
+--------------------+-----------+--------------+---------------------+----------------------+------------+-----------------------------------------------+
; 3:1                ; 284 bits  ; 568 LEs       ; 0 LEs               ; 568 LEs              ; Yes        ; |MEMWB8_Stage|sCPU_MEMWB_Stage:memwb1|wb3[2] ;
; 4:1                ; 71 bits   ; 142 LEs       ; 142 LEs             ; 0 LEs                ; No         ; |MEMWB8_Stage|Mux8:WB_ReadData_Mux|Mux31     ;
+--------------------+-----------+--------------+---------------------+----------------------+------------+-----------------------------------------------+
```

```
module   MEMWB8_Stage(

        input       sCPU0,
        input       sCPU1,
        input       sCPU2,
        input       sCPU3,
        input       sCPU4,
        input       sCPU5,
        input       sCPU6,
        input       sCPU7,
        input       [ 1: 0]selMIPS,
        input       clock,
        input       reset,
        input       M_Flush,
        input       M_Stall,
        input       WB_Stall,
        // Control Signals
        input       M_RegWrite,
        input       M_MemtoReg,
        // Data Signals
        input       [ 31: 0]M_ReadData,
        input       [ 31: 0]M_ALU_Result,
        input       [ 4: 0]M_RtRd,
        // ---------------
        output      WB_RegWrite,
        output      WB_MemtoReg,
        output      [ 31: 0]WB_ReadData,
        output      [ 31: 0]WB_ALU_Result,
        output      [ 4: 0]WB_RtRd
        );
sCPU_MEMWB_Stagememwb0(
        .mem1(clock&sCPU0),
        .mem2(reset),
        .mem3(M_Flush),
        .mem4(M_Stall),
        .mem5(WB_Stall),
        // Control Signals
        .mem6(M_RegWrite),
        .mem7(M_MemtoReg),
        // Data Signals
        .mem8(M_ReadData),
        .mem9(M_ALU_Result),
        .mem10(M_RtRd),
        // ---------------
        .wb1(WB_RegWrite0),
        .wb2(WB_MemtoReg0),
        .wb3(WB_ReadData0),
        .wb4(WB_ALU_Result0),
        .wb5(WB_RtRd0)
        );

        wire    WB_RegWrite0;
        wire    WB_MemtoReg0;
        wire    [ 31: 0]WB_ReadData0;
        wire    [ 31: 0]WB_ALU_Result0;
        wire    [ 4: 0]WB_RtRd0;

sCPU_MEMWB_Stagememwb1(
        .mem1(clock&sCPU1),
        .mem2(reset),
        .mem3(M_Flush),
        .mem4(M_Stall),
        .mem5(WB_Stall),
        // Control Signals
        .mem6(M_RegWrite),
        .mem7(M_MemtoReg),
        // Data Signals
        .mem8(M_ReadData),
        .mem9(M_ALU_Result),
        .mem10(M_RtRd),
        // ---------------
        .wb1(WB_RegWrite1),
        .wb2(WB_MemtoReg1),
        .wb3(WB_ReadData1),
        .wb4(WB_ALU_Result1),
        .wb5(WB_RtRd1)
        );
        wire    WB_RegWrite1;
        wire    WB_MemtoReg1;
        wire    [ 31: 0]WB_ReadData1;
        wire    [ 31: 0]WB_ALU_Result1;
        wire    [ 4: 0]WB_RtRd1;

sCPU_MEMWB_Stagememwb2(
        .mem1(clock&sCPU2),
        .mem2(reset),
        .mem3(M_Flush),
        .mem4(M_Stall),
        .mem5(WB_Stall),
        // Control Signals
        .mem6(M_RegWrite),
        .mem7(M_MemtoReg),
        // Data Signals
        .mem8(M_ReadData),
        .mem9(M_ALU_Result),
        .mem10(M_RtRd),
        // ---------------
        .wb1(WB_RegWrite2),
        .wb2(WB_MemtoReg2),
        .wb3(WB_ReadData2),
        .wb4(WB_ALU_Result2),
        .wb5(WB_RtRd2)
        );

        wire    WB_RegWrite2;
        wire    WB_MemtoReg2;
        wire    [ 31: 0]WB_ReadData2;
        wire    [ 31: 0]WB_ALU_Result2;
        wire    [ 4: 0]WB_RtRd2;

sCPU_MEMWB_Stagememwb3(
        .mem1(clock&sCPU3),
        .mem2(reset),
        .mem3(M_Flush),
        .mem4(M_Stall),
        .mem5(WB_Stall),
        // Control Signals
        .mem6(M_RegWrite),
        .mem7(M_MemtoReg),
        // Data Signals
        .mem8(M_ReadData),
        .mem9(M_ALU_Result),
        .mem10(M_RtRd),
        // ---------------
        .wb1(WB_RegWrite3),
        .wb2(WB_MemtoReg3),
        .wb3(WB_ReadData3),
        .wb4(WB_ALU_Result3),
        .wb5(WB_RtRd3)
        );

        wire    WB_RegWrite3;
        wire    WB_MemtoReg3;
        wire    [ 31: 0]WB_ReadData3;
        wire    [ 31: 0]WB_ALU_Result3;
        wire    [ 4: 0]WB_RtRd3;

sCPU_MEMWB_Stagememwb4(
        .mem1(clock&sCPU4),
        .mem2(reset),
        .mem3(M_Flush),
        .mem4(M_Stall),
        .mem5(WB_Stall),
        // Control Signals
        .mem6(M_RegWrite),
        .mem7(M_MemtoReg),
        // Data Signals
        .mem8(M_ReadData),
        .mem9(M_ALU_Result),
        .mem10(M_RtRd),
        // ---------------
        .wb1(WB_RegWrite4),
        .wb2(WB_MemtoReg4),
        .wb3(WB_ReadData4),
        .wb4(WB_ALU_Result4),
        .wb5(WB_RtRd4)
        );

        wire    WB_RegWrite4;
        wire    WB_MemtoReg4;
        wire    [ 31: 0]WB_ReadData4;
        wire    [ 31: 0]WB_ALU_Result4;
        wire    [ 4: 0]WB_RtRd4;

sCPU_MEMWB_Stagememwb5(
        .mem1(clock&sCPU5),
        .mem2(reset),
        .mem3(M_Flush),
        .mem4(M_Stall),
        .mem5(WB_Stall),
        // Control Signals
        .mem6(M_RegWrite),
        .mem7(M_MemtoReg),
        // Data Signals
        .mem8(M_ReadData),
        .mem9(M_ALU_Result),
        .mem10(M_RtRd),
        // ---------------
        .wb1(WB_RegWrite5),
        .wb2(WB_MemtoReg5),
        .wb3(WB_ReadData5),
        .wb4(WB_ALU_Result5),
        .wb5(WB_RtRd5)
        );

        wire    WB_RegWrite5;
```

```
187         wire    WB_MemtoReg5;
188         wire    [ 31: 0]WB_ReadData5;
189         wire    [ 31: 0]WB_ALU_Result5;
190         wire    [ 4: 0]WB_RtRd5;
191
192     sCPU_MEMWB_Stagememwb6(
193         .mem1(clock&sCPU6),
194         .mem2(reset),
195         .mem3(M_Flush),
196         .mem4(M_Stall),
197         .mem5(WB_Stall),
198         // Control Signals
199         .mem6(M_RegWrite),
200         .mem7(M_MemtoReg),
201         // Data Signals
202         .mem8(M_ReadData),
203         .mem9(M_ALU_Result),
204         .mem10(M_RtRd),
205         // ---------------
206         .wb1(WB_RegWrite6),
207         .wb2(WB_MemtoReg6),
208         .wb3(WB_ReadData6),
209         .wb4(WB_ALU_Result6),
210         .wb5(WB_RtRd6)
211         );
212
213         wire    WB_RegWrite6;
214         wire    WB_MemtoReg6;
215         wire    [ 31: 0]WB_ReadData6;
216         wire    [ 31: 0]WB_ALU_Result6;
217         wire    [ 4: 0]WB_RtRd6;
218
219     sCPU_MEMWB_Stagememwb7(
220         .mem1(clock&sCPU7),
221         .mem2(reset),
222         .mem3(M_Flush),
223         .mem4(M_Stall),
224         .mem5(WB_Stall),
225         // Control Signals
226         .mem6(M_RegWrite),
227         .mem7(M_MemtoReg),
228         // Data Signals
229         .mem8(M_ReadData),
230         .mem9(M_ALU_Result),
231         .mem10(M_RtRd),
232         // ---------------
233         .wb1(WB_RegWrite7),
234         .wb2(WB_MemtoReg7),
235         .wb3(WB_ReadData7),
236         .wb4(WB_ALU_Result7),
237         .wb5(WB_RtRd7)
238         );
239
240         wire    WB_RegWrite7;
241         wire    WB_MemtoReg7;
242         wire    [ 31: 0]WB_ReadData7;
243         wire    [ 31: 0]WB_ALU_Result7;
244         wire    [ 4: 0]WB_RtRd7;
245
246
247     Mux8#(.WIDTH(    1))WB_RegWrite_Mux(
248                 .sel(selMIPS),
249                 .in0(WB_RegWrite0),
250                 .in1(WB_RegWrite1),
251                 .in2(WB_RegWrite2),
252                 .in3(WB_RegWrite3),
253                 .in4(WB_RegWrite4),
254                 .in5(WB_RegWrite5),
255                 .in6(WB_RegWrite6),
256                 .in7(WB_RegWrite7),
257                 .out(WB_RegWrite)
258                 );
259
260     Mux8#(.WIDTH(    1))WB_MemtoReg_Mux(
261                 .sel(selMIPS),
262                 .in0(WB_MemtoReg0),
263                 .in1(WB_MemtoReg1),
264                 .in2(WB_MemtoReg2),
265                 .in3(WB_MemtoReg3),
266                 .in4(WB_MemtoReg4),
267                 .in5(WB_MemtoReg5),
268                 .in6(WB_MemtoReg6),
269                 .in7(WB_MemtoReg7),
270                 .out(WB_MemtoReg)
271                 );
272
273     Mux8#(.WIDTH(    32))WB_ReadData_Mux(
274                 .sel(selMIPS),
275                 .in0(WB_ReadData0),
276                 .in1(WB_ReadData1),
277                 .in2(WB_ReadData2),
278                 .in3(WB_ReadData3),
279                 .in4(WB_ReadData4),
280                 .in5(WB_ReadData5),
281                 .in6(WB_ReadData6),
282                 .in7(WB_ReadData7),
283                 .out(WB_ReadData)
284                 );
285
286     Mux8#(.WIDTH(    32))WB_ALU_Result_Mux(
287                 .sel(selMIPS),
288                 .in0(WB_ALU_Result0),
289                 .in1(WB_ALU_Result1),
290                 .in2(WB_ALU_Result2),
291                 .in3(WB_ALU_Result3),
292                 .in4(WB_ALU_Result4),
293                 .in5(WB_ALU_Result5),
294                 .in6(WB_ALU_Result6),
295                 .in7(WB_ALU_Result7),
296                 .out(WB_ALU_Result)
297                 );
298
299     Mux8#(.WIDTH(    5))WB_RtRd_Mux(
300                 .sel(selMIPS),
301                 .in0(WB_RtRd0),
302                 .in1(WB_RtRd1),
303                 .in2(WB_RtRd2),
304                 .in3(WB_RtRd3),
305                 .in4(WB_RtRd4),
306                 .in5(WB_RtRd5),
307                 .in6(WB_RtRd6),
308                 .in7(WB_RtRd7),
309                 .out(WB_RtRd)
310                 );
311
312
313
314     endmodule
315
```

Figură 2-43 Codul verilog pentru MEMWB8



Figură 2-44 Chip planner MEMWB8

```
        MEMWB8_Stage-Post-Synthesis Netlist Statistics for Top Partition
Post-Synthesis Netlist Statistics for Top Partition report for MEMWB8_Stage
Wed Sep 23 14:20:32 2015
Quartus II 64-Bit Version 15.0.0 Build 145 04/22/2015 SJ Web Edition

+-----------------------------------------------------+
; Post-Synthesis Netlist Statistics for Top Partition ;
+----------------------+------------------------------+
; Type                 ; Count                        ;
+----------------------+------------------------------+
; arriav_ff            ; 284                          ;
;     ENA SCLR         ; 284                          ;
; arriav_lcell_comb    ; 77                           ;
;     normal           ; 77                           ;
;         2 data inputs ; 5                          ;
;         3 data inputs ; 1                          ;
;         6 data inputs ; 71                         ;
; boundary_port        ; 157                          ;
;                      ;                              ;
; Max LUT depth        ; 1.00                         ;
; Average LUT depth    ; 0.84                         ;
+----------------------+------------------------------+
```

MEMWB8_Stage-Routing Usage Summary

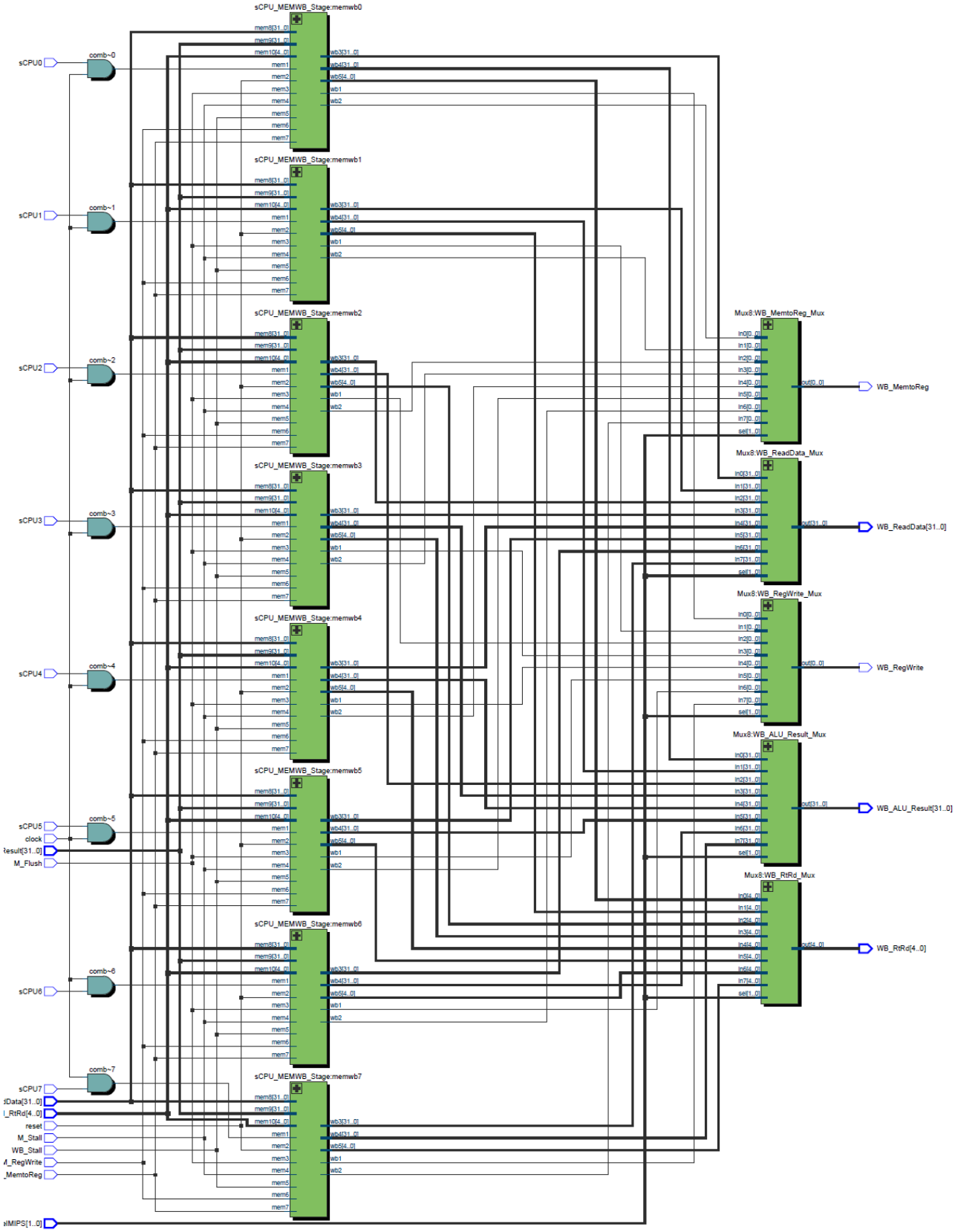
Routing Usage Summary report for MEMWB8_Stage
Wed Sep 23 14:21:47 2015
Quartus II 64-Bit Version 15.0.0 Build 145 04/22/2015 SJ Web Edition

Routing Usage Summary	
Routing Resource Type	Usage
Block interconnects	708 / 374,484 (< 1 %)
C12 interconnects	20 / 16,664 (< 1 %)
C2 interconnects	207 / 155,012 (< 1 %)
C4 interconnects	168 / 72,600 (< 1 %)
DQS bus muxes	0 / 30 (0 %)
DQS-18 I/O buses	0 / 30 (0 %)
DQS-9 I/O buses	0 / 30 (0 %)
Direct links	26 / 374,484 (< 1 %)
Global clocks	0 / 16 (0 %)
Horizontal periphery clocks	0 / 72 (0 %)
Local interconnects	4 / 112,960 (< 1 %)
Quadrant clocks	0 / 88 (0 %)
R14 interconnects	158 / 15,868 (< 1 %)
R14/C12 interconnect drivers	158 / 27,256 (< 1 %)
R3 interconnects	367 / 169,296 (< 1 %)
R6 interconnects	637 / 330,800 (< 1 %)
Spine clocks	0 / 480 (0 %)
Wire stub REs	0 / 20,834 (0 %)

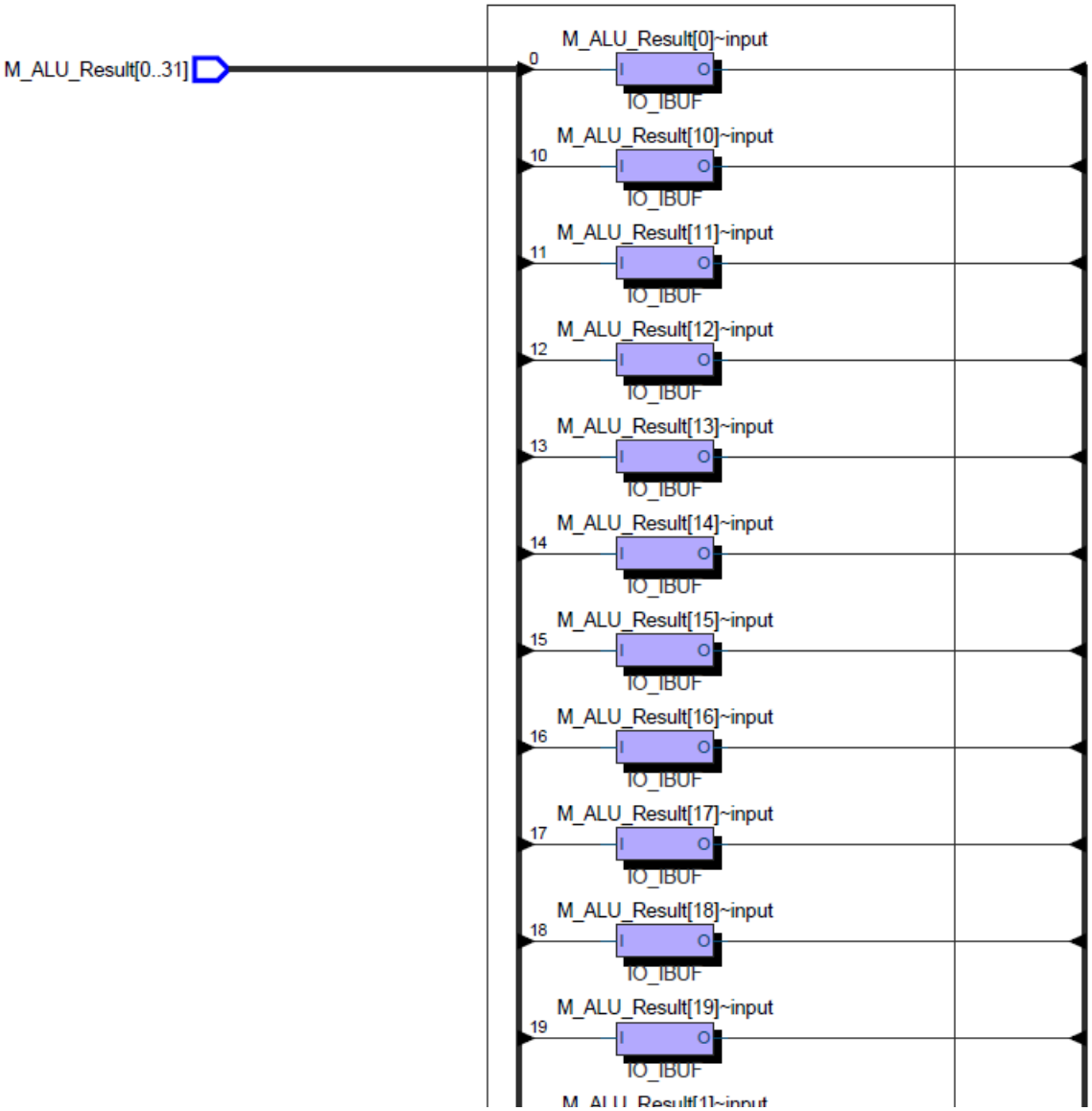
MEMWB8_Stage-Flow Summary

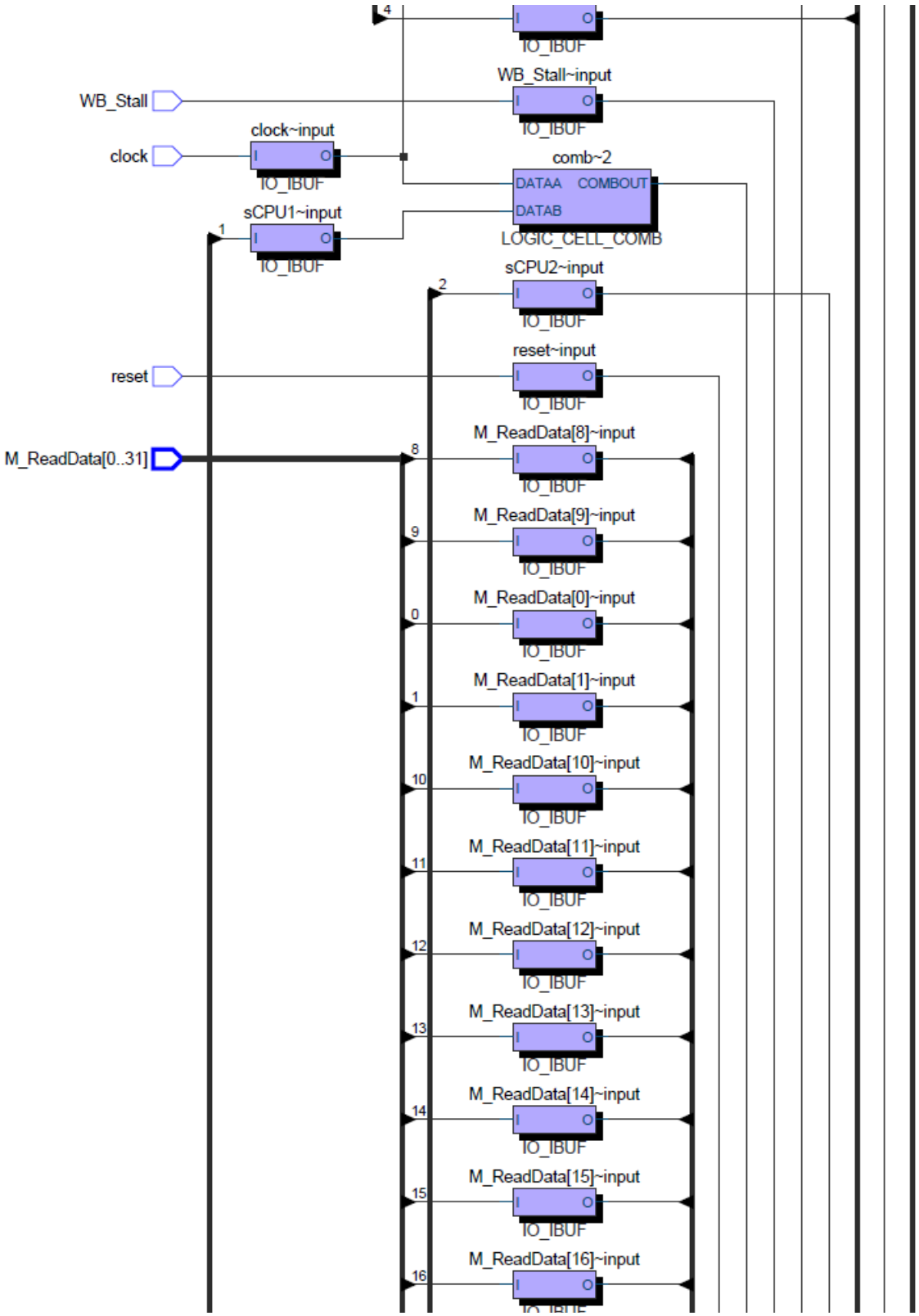
Flow Summary report for MEMWB8_Stage
Wed Sep 23 14:19:47 2015
Quartus II 64-Bit Version 15.0.0 Build 145 04/22/2015 SJ Web Edition

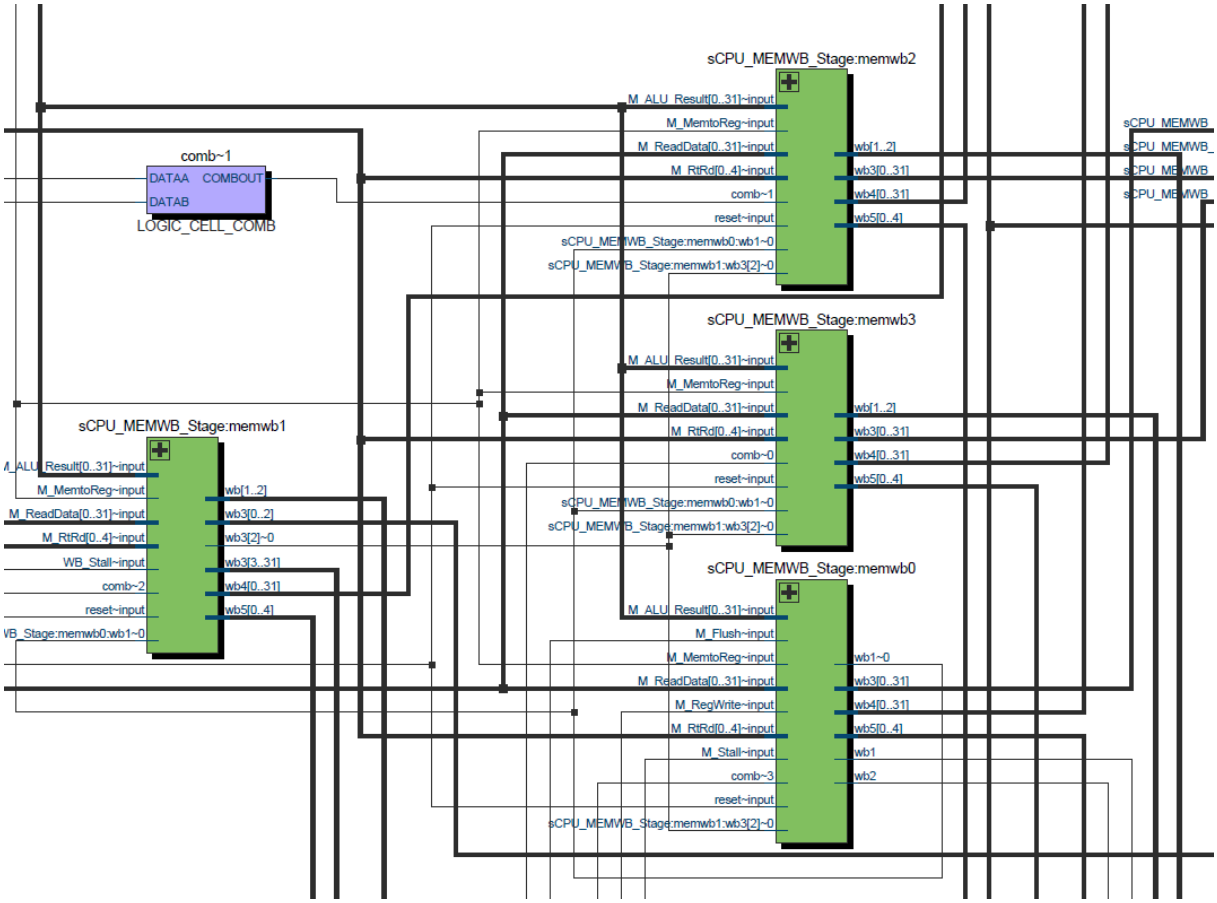
Flow Summary	
Flow Status	Successful - Mon Sep 21 23:30:02 2015
Quartus II 64-Bit Version	15.0.0 Build 145 04/22/2015 SJ Web Edition
Revision Name	MEMWB8_Stage
Top-level Entity Name	MEMWB8_Stage
Family	Cyclone V
Device	5CGXFC7D7F31C8
Timing Models	Final
Logic utilization (in ALMs)	134 / 56,480 (< 1 %)
Total registers	284
Total pins	157 / 522 (30 %)
Total virtual pins	0
Total block memory bits	0 / 7,024,640 (0 %)
Total DSP Blocks	0 / 156 (0 %)
Total HSSI RX PCSs	0 / 9 (0 %)
Total HSSI PMA RX Deserializers	0 / 9 (0 %)
Total HSSI TX PCSs	0 / 9 (0 %)
Total HSSI PMA TX Serializers	0 / 9 (0 %)
Total PLLs	0 / 16 (0 %)
Total DLLs	0 / 4 (0 %)



Figură 2-45 RTL MEMWB8


M_ALU_Result[0..31]
M_ALU_Result[0]~input
0
IO_IBUF
M_ALU_Result[10]~input
10
IO_IBUF
M_ALU_Result[11]~input
11
IO_IBUF
M_ALU_Result[12]~input
12
IO_IBUF
M_ALU_Result[13]~input
13
IO_IBUF
M_ALU_Result[14]~input
14
IO_IBUF
M_ALU_Result[15]~input
15
IO_IBUF
M_ALU_Result[16]~input
16
IO_IBUF
M_ALU_Result[17]~input
17
IO_IBUF
M_ALU_Result[18]~input
18
IO_IBUF
M_ALU_Result[19]~input
19
IO_IBUF


WB_Stall
clock
reset
M_ReadData[0..31]
clock~input
sCPU1~input
WB_Stall~input
comb~2
DATAA
COMBOUT
DATAB
LOGIC_CELL_COMB
sCPU2~input
reset~input
M_ReadData[8]~input
M_ReadData[9]~input
M_ReadData[0]~input
M_ReadData[1]~input
M_ReadData[10]~input
M_ReadData[11]~input
M_ReadData[12]~input
M_ReadData[13]~input
M_ReadData[14]~input
M_ReadData[15]~input
M_ReadData[16]~input
IO_IBUF



Figură 2-46 Parţial Tehnology map MEMWB8

2.3.5. PC8

PC8-Flow Summary

```
Flow Summary report for PC8
Wed Sep 23 14:28:16 2015
Quartus II 64-Bit Version 15.0.0 Build 145 04/22/2015 SJ Web Edition

+---------------------------------------------------------------------------------+
; Flow Summary                                                                    ;
+---------------------------------+-----------------------------------------------+
; Flow Status                     ; Successful - Mon Sep 21 23:45:30 2015         ;
; Quartus II 64-Bit Version       ; 15.0.0 Build 145 04/22/2015 SJ Web Edition    ;
; Revision Name                   ; PC8                                           ;
; Top-level Entity Name           ; PC8                                           ;
; Family                          ; Cyclone V                                     ;
; Device                          ; 5CGXFC7D7F31C8                                ;
; Timing Models                   ; Final                                         ;
; Logic utilization (in ALMs)     ; 1 / 56,480 ( < 1 % )                          ;
; Total registers                 ; 0                                             ;
; Total pins                      ; 78 / 522 ( 15 % )                             ;
; Total virtual pins              ; 0                                             ;
; Total block memory bits         ; 0 / 7,024,640 ( 0 % )                         ;
; Total DSP Blocks                ; 0 / 156 ( 0 % )                               ;
; Total HSSI RX PCSs              ; 0 / 9 ( 0 % )                                 ;
; Total HSSI PMA RX Deserializers ; 0 / 9 ( 0 % )                                 ;
; Total HSSI TX PCSs              ; 0 / 9 ( 0 % )                                 ;
; Total HSSI PMA TX Serializers   ; 0 / 9 ( 0 % )                                 ;
; Total PLLs                      ; 0 / 16 ( 0 % )                                ;
; Total DLLs                      ; 0 / 4 ( 0 % )                                 ;
+---------------------------------+-----------------------------------------------+
```

```
1   module   PC8(
2
3       input    [ 1: 0]selPC,
4       input    en_MIPS,
5       input    clock,
6       input    reset,
7       input    enable,
8       input    [ 31: 0]IF_PCIn,
9
10      output    sCPU0,
11      output    sCPU1,
12      output    sCPU2,
13      output    sCPU3,
14      output    sCPU4,
15      output    sCPU5,
16      output    sCPU6,
17      output    sCPU7,
18      output    [ 31: 0]IF_PCOut
19
20  );
21
22  wire  [ 31: 0]Q0,Q1,Q2,Q3,Q4,Q5,Q6,Q7;
23
24
25
26          /*** Program Counter (MIPS spec is 0xBFC00000 starting address) ***/
27          Register#(.WIDTH(        32))PC0(
28                  .clock(clock),
29                  .reset(reset),
30                  //.enable  (~IF_Stall),   // XXX verify. HERE. Was 1 but on stall latches PC+4, ad
    nauseum.
31                  .enable((enable&sCPU0)),
32                  .D(IF_PCIn),
33                  .Q(Q0)
34          );
35
36
37          /*** Program Counter (MIPS spec is 0xBFC00000 starting address) ***/
38          Register#(.WIDTH(        32))PC1(
39                  .clock(clock),
40                  .reset(reset),
41                  //.enable  (~IF_Stall),   // XXX verify. HERE. Was 1 but on stall latches PC+4, ad
    nauseum.
42                  .enable((enable&sCPU1)),
43                  .D(IF_PCIn),
44                  .Q(Q1)
45          );
46
47          /*** Program Counter (MIPS spec is 0xBFC00000 starting address) ***/
48          Register#(.WIDTH(        32))PC2(
49                  .clock(clock),
50                  .reset(reset),
51                  //.enable  (~IF_Stall),   // XXX verify. HERE. Was 1 but on stall latches PC+4, ad
    nauseum.
52                  .enable((enable&sCPU2)),
53                  .D(IF_PCIn),
54                  .Q(Q2)
55          );
56
57          /*** Program Counter (MIPS spec is 0xBFC00000 starting address) ***/
58          Register#(.WIDTH(        32))PC3(
59                  .clock(clock),
```

```
60              .reset(reset),
61              //.enable  (~IF_Stall),   // XXX verify. HERE. Was 1 but on stall latches PC+4, ad
    nauseum.
62              .enable((enable&sCPU3)),
63              .D(IF_PCIn),
64              .Q(Q3)
65          );
66
67
68           /*** Program Counter (MIPS spec is 0xBFC00000 starting address) ***/
69          Register#(.WIDTH(       32))PC4(
70              .clock(clock),
71              .reset(reset),
72              //.enable  (~IF_Stall),   // XXX verify. HERE. Was 1 but on stall latches PC+4, ad
    nauseum.
73              .enable((enable&sCPU4)),
74              .D(IF_PCIn),
75              .Q(Q4)
76          );
77
78           /*** Program Counter (MIPS spec is 0xBFC00000 starting address) ***/
79          Register#(.WIDTH(       32))PC5(
80              .clock(clock),
81              .reset(reset),
82              //.enable  (~IF_Stall),   // XXX verify. HERE. Was 1 but on stall latches PC+4, ad
    nauseum.
83              .enable((enable&sCPU5)),
84              .D(IF_PCIn),
85              .Q(Q5)
86          );
87
88           /*** Program Counter (MIPS spec is 0xBFC00000 starting address) ***/
89          Register#(.WIDTH(       32))PC6(
90              .clock(clock),
91              .reset(reset),
92              //.enable  (~IF_Stall),   // XXX verify. HERE. Was 1 but on stall latches PC+4, ad
    nauseum.
93              .enable((enable&sCPU6)),
94              .D(IF_PCIn),
95              .Q(Q6)
96          );
97
98           /*** Program Counter (MIPS spec is 0xBFC00000 starting address) ***/
99          Register#(.WIDTH(       32))PC7(
100             .clock(clock),
101             .reset(reset),
102             //.enable  (~IF_Stall),   // XXX verify. HERE. Was 1 but on stall latches PC+4, ad
    nauseum.
103             .enable((enable&sCPU7)),
104             .D(IF_PCIn),
105             .Q(Q7)
106         );
107
108
109         /*** PC Source Non-Exception Mux ***/
110         Mux8#(.WIDTH(    32))PCsCPUi_Mux(
111             .sel(selPC),
112             .in0(Q0),                           //(IF_PCout0),
113             .in1(Q1),                          // (IF_PCout1),
114             .in2(Q2),                           // (IF_PCout2),
115             .in3(Q3),                           //(IF_PCout3),
116             .in4(Q4).                           //(IF  PCout4).
117             .in5(Q5),                          // (IF_PCout5),
118             .in6(Q6),                           // (IF_PCout6),
119             .in7(Q7),                           //(IF_PCout7),
120             .out(IF_PCOut)
121         );
122
123 endmodule
124
```

Figură 2-47 Codul verilog pentru PC8

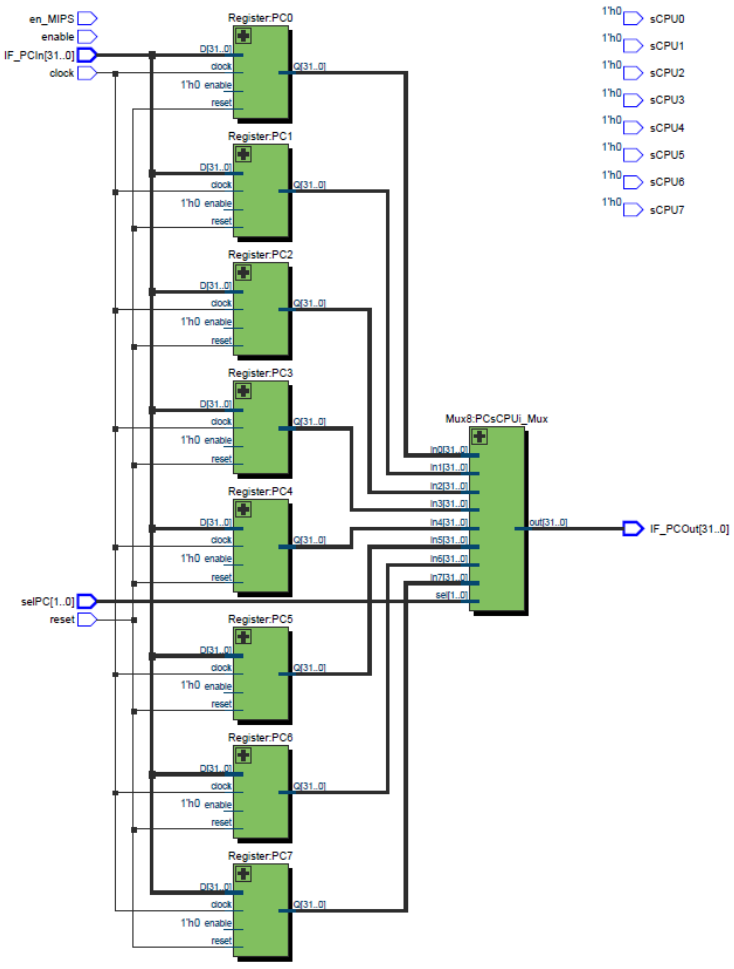
PC8-Post-Synthesis Netlist Statistics for Top Partition
Post-Synthesis Netlist Statistics for Top Partition report for PC8
Wed Sep 23 14:29:16 2015
Quartus II 64-Bit Version 15.0.0 Build 145 04/22/2015 SJ Web Edition

Post-Synthesis Netlist Statistics for Top Partition	
Type	Count
arriav_lcell_comb	1
normal	1
0 data inputs	1
boundary_port	78
Max LUT depth	0.00
Average LUT depth	0.00

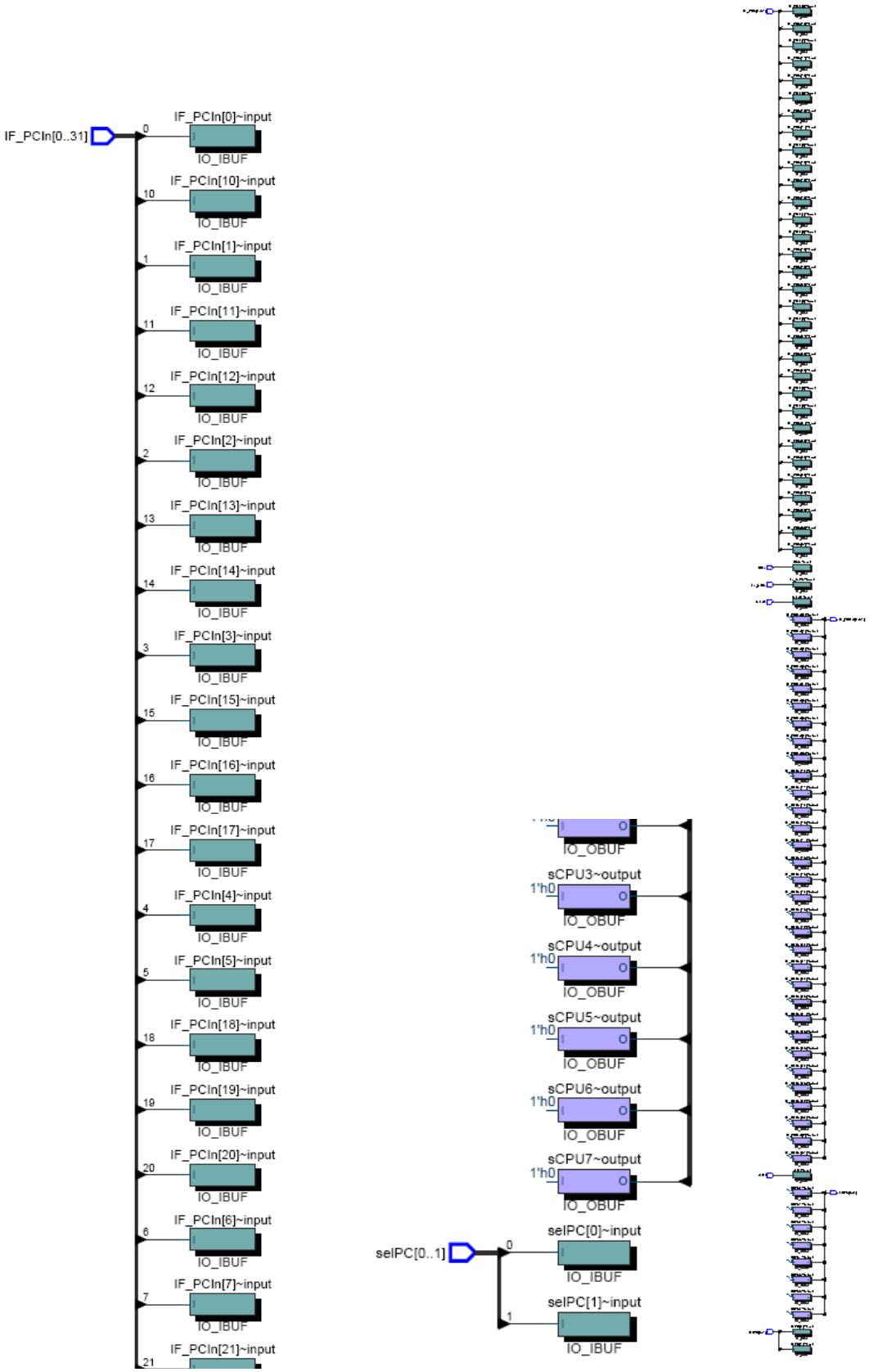
PC8-Partition Statistics
Partition Statistics report for PC8
Wed Sep 23 14:29:56 2015
Quartus II 64-Bit Version 15.0.0 Build 145 04/22/2015 SJ Web Edition

Fitter Partition Statistics		
Statistic	Top	hard_block:auto_generated_inst
Logic utilization (ALMs needed / total ALMs on device)	1 / 56480 (< 1 %)	0 / 56480 (0 %)
ALMs needed [=A-B+C]	1	0
[A] ALMs used in final placement [=a+b+c+d]	1 / 56480 (< 1 %)	0 / 56480 (0 %)
[a] ALMs used for LUT logic and registers	0	0
[b] ALMs used for LUT logic	1	0
[c] ALMs used for registers	0	0
[d] ALMs used for memory (up to half of total ALMs)	0	0
[B] Estimate of ALMs recoverable by dense packing	0 / 56480 (0 %)	0 / 56480 (0 %)
[C] Estimate of ALMs unavailable [=a+b+c+d]	0 / 56480 (0 %)	0 / 56480 (0 %)
[a] Due to location constrained logic	0	0
[b] Due to LAB-wide signal conflicts	0	0
[c] Due to LAB input limits	0	0
[d] Due to virtual I/Os	0	0
Difficulty packing design	Low	Low
Total LABs: partially or completely used	1 / 5648 (< 1 %)	0 / 5648 (0 %)
-- Logic LABs	1	0
-- Memory LABs (up to half of total LABs)	0	0
Combinational ALUT usage for logic	1	0
-- 7 input functions	0	0
-- 6 input functions	0	0
-- 5 input functions	0	0
-- 4 input functions	0	0
-- <=3 input functions	1	0
Combinational ALUT usage for route-throughs	0	0
Memory ALUT usage	0	0
-- 64-address deep	0	0
-- 32-address deep	0	0
Dedicated logic registers	0	0
-- By type:		
-- Primary logic registers	0 / 112960 (0 %)	0 / 112960 (0 %)
-- Secondary logic registers	0 / 112960 (0 %)	0 / 112960 (0 %)

-- By function:		
-- Design implementation registers	0	0
-- Routing optimization registers	0	0
Virtual pins	0	0
I/O pins	78	0
I/O registers	0	0
Total block memory bits	0	0
Total block memory implementation bits	0	0
Connections		
-- Input Connections	0	0
-- Registered Input Connections	0	0
-- Output Connections	0	0
-- Registered Output Connections	0	0
Internal Connections		
-- Total Connections	78	0
-- Registered Connections	0	0
External Connections		
-- Top	0	0
-- hard_block:auto_generated_inst	0	0
Partition Interface		
-- Input Ports	38	0
-- Output Ports	40	0
-- Bidir Ports	0	0
Registered Ports		
-- Registered Input Ports	0	0
-- Registered Output Ports	0	0
Port Connectivity		
-- Input Ports driven by GND	0	0
-- Output Ports driven by GND	0	0
-- Input Ports driven by VCC	0	0
-- Output Ports driven by VCC	0	0
-- Input Ports with no Source	0	0
-- Output Ports with no Source	0	0
-- Input Ports with no Fanout	0	0
-- Output Ports with no Fanout	0	0



Figură 2-48 RTL PC8



Figură 2-49 Tehnology map PC8

2.3.6. Register file8

```
module  RegisterFile8(
        input   sCPU0,
        input   sCPU1,
        input   sCPU2,
        input   sCPU3,
        input   sCPU4,
        input   sCPU5,
        input   sCPU6,
        input   sCPU7,
        input   [ 1: 0]selMIPS,
        input   clock,
        input   reset,
        input   [ 4: 0]ReadReg1,
        input   [ 4: 0]ReadReg2,
        input   [ 4: 0]WriteReg,
        input   [ 31: 0]WriteData,
        input   RegWrite,

        output   [ 31: 0]ID_ReadData1_RF,
        output   [ 31: 0]ID_ReadData2_RF
);

      wire   [ 31: 0]ID_ReadData1_RF0;
      wire   [ 31: 0]ID_ReadData2_RF0;
      wire   [ 31: 0]ID_ReadData1_RF1;
      wire   [ 31: 0]ID_ReadData2_RF1;
      wire   [ 31: 0]ID_ReadData1_RF2;
      wire   [ 31: 0]ID_ReadData2_RF2;
      wire   [ 31: 0]ID_ReadData1_RF3;
      wire   [ 31: 0]ID_ReadData2_RF3;
      wire   [ 31: 0]ID_ReadData1_RF4;
      wire   [ 31: 0]ID_ReadData2_RF4;
      wire   [ 31: 0]ID_ReadData1_RF5;
      wire   [ 31: 0]ID_ReadData2_RF5;
      wire   [ 31: 0]ID_ReadData1_RF6;
      wire   [ 31: 0]ID_ReadData2_RF6;
      wire   [ 31: 0]ID_ReadData1_RF7;
      wire   [ 31: 0]ID_ReadData2_RF7;


       /*** Register File ***/
       RegisterFileRegisterFile_sCPU0(
            .clock(clock&sCPU0),
            .reset(reset),
            .ReadReg1(ReadReg1),
            .ReadReg2(ReadReg2),
            .WriteReg(WriteReg),
            .WriteData(WriteData),
            .RegWrite(RegWrite),
            .ReadData1(ID_ReadData1_RF0),
            .ReadData2(ID_ReadData2_RF0)
       );

       /*** Register File ***/
       RegisterFileRegisterFile_sCPU1(
            .clock(clock&sCPU1),
            .reset(reset),
            .ReadReg1(ReadReg1),
            .ReadReg2(ReadReg2),
            .WriteReg(WriteReg),
            .WriteData(WriteData),
            .RegWrite(RegWrite),
            .ReadData1(ID_ReadData1_RF1),
            .ReadData2(ID_ReadData2_RF1)
       );

       /*** Register File ***/
       RegisterFileRegisterFile_sCPU2(
            .clock(clock&sCPU2),
            .reset(reset),
            .ReadReg1(ReadReg1),
            .ReadReg2(ReadReg2),
            .WriteReg(WriteReg),
            .WriteData(WriteData),
            .RegWrite(RegWrite),
            .ReadData1(ID_ReadData1_RF2),
            .ReadData2(ID_ReadData2_RF2)
       );

       /*** Register File ***/
       RegisterFileRegisterFile_sCPU3(
            .clock(clock&sCPU3),
            .reset(reset),
            .ReadReg1(ReadReg1),
            .ReadReg2(ReadReg2),
            .WriteReg(WriteReg),
            .WriteData(WriteData),
            .RegWrite(RegWrite),
            .ReadData1(ID_ReadData1_RF3),
            .ReadData2(ID_ReadData2_RF3)
       );

      /*** Register File ***/
       RegisterFileRegisterFile_sCPU4(
            .clock(clock&sCPU4),
            .reset(reset),
            .ReadReg1(ReadReg1),
            .ReadReg2(ReadReg2),
            .WriteReg(WriteReg),
            .WriteData(WriteData),
            .RegWrite(RegWrite),
            .ReadData1(ID_ReadData1_RF4),
            .ReadData2(ID_ReadData2_RF4)
       );

      /*** Register File ***/
       RegisterFileRegisterFile_sCPU5(
            .clock(clock&sCPU5),
            .reset(reset),
            .ReadReg1(ReadReg1),
            .ReadReg2(ReadReg2),
            .WriteReg(WriteReg),
            .WriteData(WriteData),
            .RegWrite(RegWrite),
            .ReadData1(ID_ReadData1_RF5),
            .ReadData2(ID_ReadData2_RF5)
       );

      /*** Register File ***/
       RegisterFileRegisterFile_sCPU6(
            .clock(clock&sCPU6),
            .reset(reset),
            .ReadReg1(ReadReg1),
            .ReadReg2(ReadReg2),
            .WriteReg(WriteReg),
            .WriteData(WriteData),
            .RegWrite(RegWrite),
            .ReadData1(ID_ReadData1_RF6),
            .ReadData2(ID_ReadData2_RF6)
       );

      /*** Register File ***/
       RegisterFileRegisterFile_sCPU7(
            .clock(clock&sCPU7),
            .reset(reset),
            .ReadReg1(ReadReg1),
            .ReadReg2(ReadReg2),
            .WriteReg(WriteReg),
            .WriteData(WriteData),
            .RegWrite(RegWrite),
            .ReadData1(ID_ReadData1_RF7),
            .ReadData2(ID_ReadData2_RF7)
       );

      /*** ID Rs Forwarding/Link Mux ***/
       Mux8#(.WIDTH(   32 ))RegFile_Data2(
            .sel(selMIPS),
            .in0(ID_ReadData2_RF0),
            .in1(ID_ReadData2_RF1),
            .in2(ID_ReadData2_RF2),
            .in3(ID_ReadData2_RF3),
            .in4(ID_ReadData2_RF4),
            .in5(ID_ReadData2_RF5),
            .in6(ID_ReadData2_RF6),
            .in7(ID_ReadData2_RF7),
            .out(ID_ReadData2_RF)
            );

endmodule
```

Figură 2-50 Codul verilog pentru Register file8

RegisterFile8-Flow Summary

Flow Summary report for RegisterFile8
Wed Sep 23 14:35:45 2015
Quartus II 64-Bit Version 15.0.0 Build 145 04/22/2015 SJ Web Edition

Flow Summary	
Flow Status	Successful - Mon Sep 21 23:56:54 2015
Quartus II 64-Bit Version	15.0.0 Build 145 04/22/2015 SJ Web Edition
Revision Name	RegisterFile8
Top-level Entity Name	RegisterFile8
Family	Cyclone V
Device	5CGXFC7D7F31C8
Timing Models	Final
Logic utilization (in ALMs)	4,218 / 56,480 (7 %)
Total registers	3968
Total pins	124 / 522 (24 %)
Total virtual pins	0
Total block memory bits	0 / 7,024,640 (0 %)
Total DSP Blocks	0 / 156 (0 %)
Total HSSI RX PCSs	0 / 9 (0 %)
Total HSSI PMA RX Deserializers	0 / 9 (0 %)
Total HSSI TX PCSs	0 / 9 (0 %)
Total HSSI PMA TX Serializers	0 / 9 (0 %)
Total PLLs	0 / 16 (0 %)
Total DLLs	0 / 4 (0 %)

RegisterFile8-Post-Synthesis Netlist Statistics for Top Partition

Post-Synthesis Netlist Statistics for Top Partition report for RegisterFile8
Wed Sep 23 14:36:34 2015
Quartus II 64-Bit Version 15.0.0 Build 145 04/22/2015 SJ Web Edition

Post-Synthesis Netlist Statistics for Top Partition	
Type	Count
arriav_ff	3968
ENA SCLR	3968
arriav_lcell_comb	4115
extend	64
7 data inputs	64
normal	4051
2 data inputs	42
3 data inputs	6
4 data inputs	2
5 data inputs	258
6 data inputs	3743
boundary_port	124
Max LUT depth	4.00
Average LUT depth	3.56

RegisterFile8-Routing Usage Summary

Routing Usage Summary report for RegisterFile8
Wed Sep 23 14:37:47 2015
Quartus II 64-Bit Version 15.0.0 Build 145 04/22/2015 SJ Web Edition

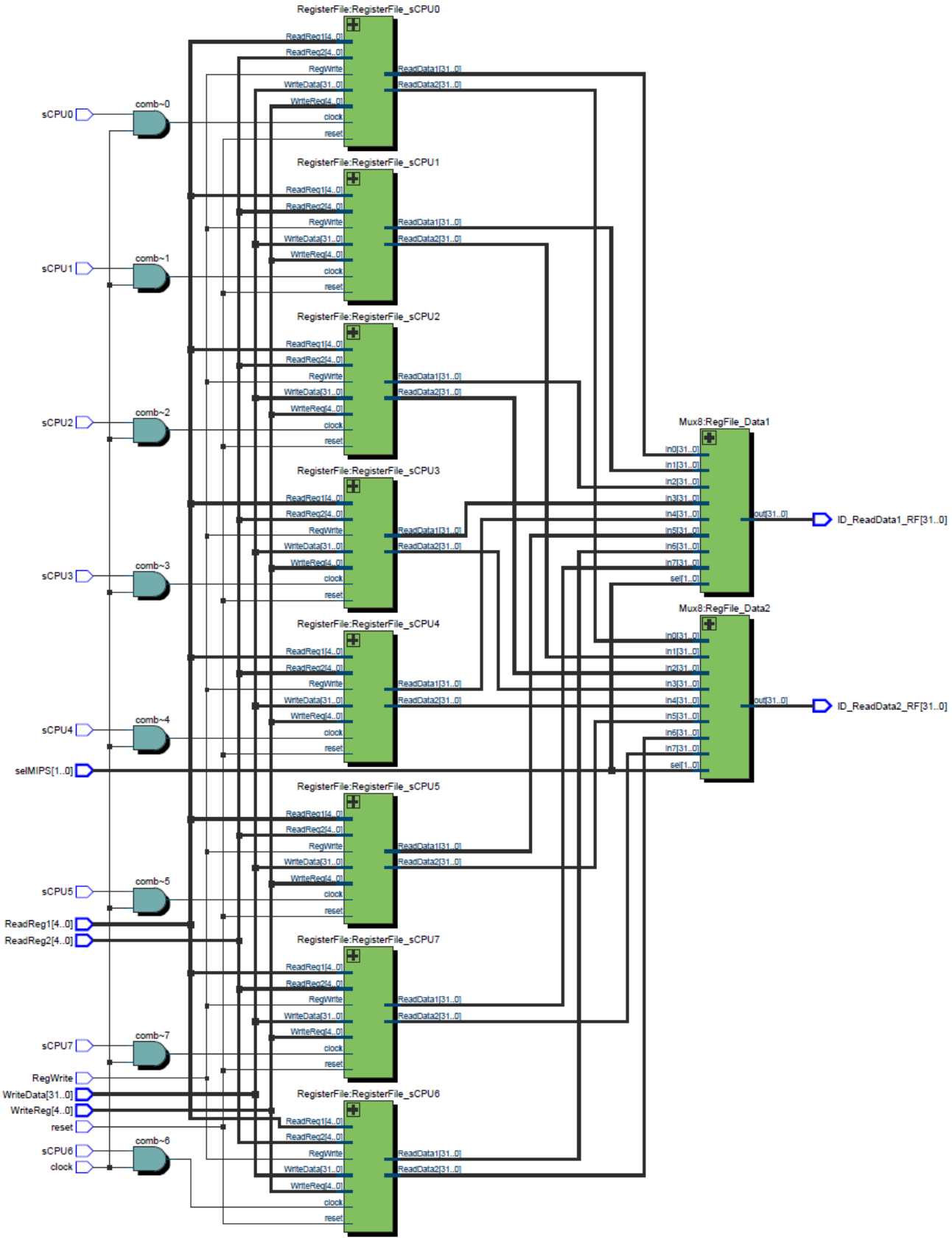
Routing Usage Summary	
Routing Resource Type	Usage
Block interconnects	20,100 / 374,484 (5 %)
C12 interconnects	374 / 16,664 (2 %)
C2 interconnects	6,399 / 155,012 (4 %)
C4 interconnects	3,588 / 72,600 (5 %)
DQS bus muxes	0 / 30 (0 %)
DQS-18 I/O buses	0 / 30 (0 %)
DQS-9 I/O buses	0 / 30 (0 %)
Direct links	698 / 374,484 (< 1 %)
Global clocks	0 / 16 (0 %)
Horizontal periphery clocks	0 / 72 (0 %)
Local interconnects	3,267 / 112,960 (3 %)
Quadrant clocks	0 / 88 (0 %)
R14 interconnects	711 / 15,868 (4 %)
R14/C12 interconnect drivers	846 / 27,256 (3 %)
R3 interconnects	8,420 / 169,296 (5 %)
R6 interconnects	13,199 / 330,800 (4 %)
Spine clocks	0 / 480 (0 %)
Wire stub REs	0 / 20,834 (0 %)

RegisterFile8-Resource Usage Summary

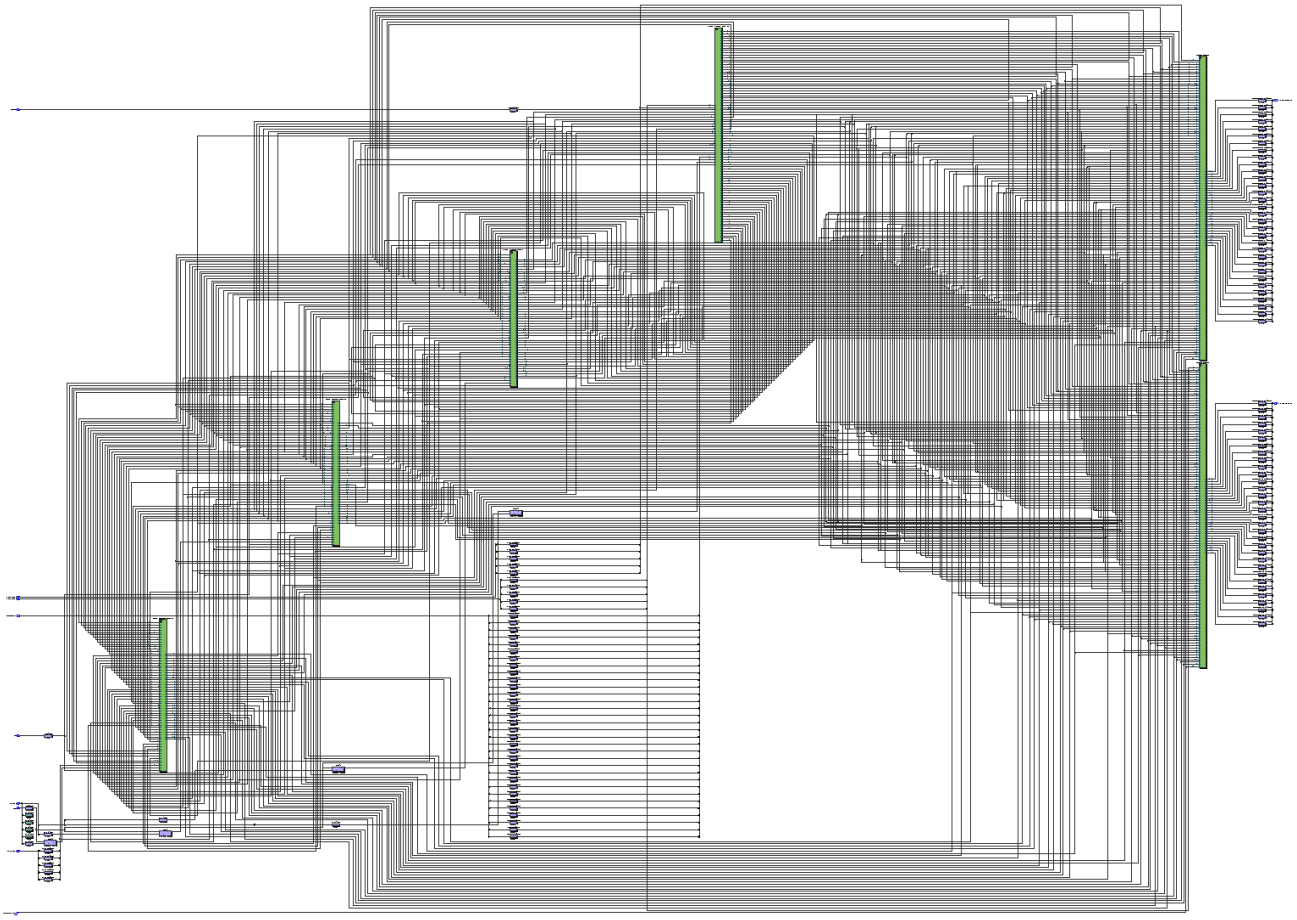
Resource Usage Summary report for RegisterFile8
Wed Sep 23 14:37:15 2015
Quartus II 64-Bit Version 15.0.0 Build 145 04/22/2015 SJ Web Edition

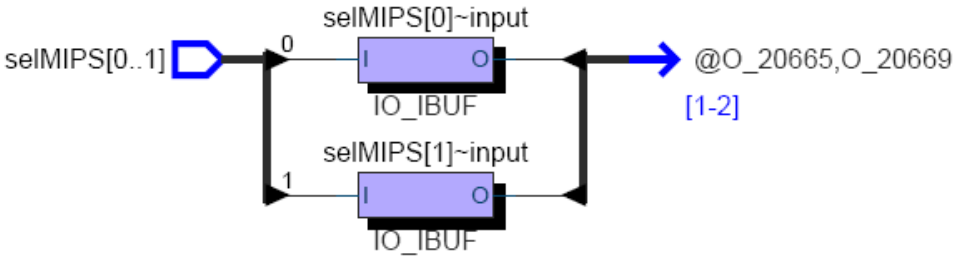
; Fitter Resource Usage Summary

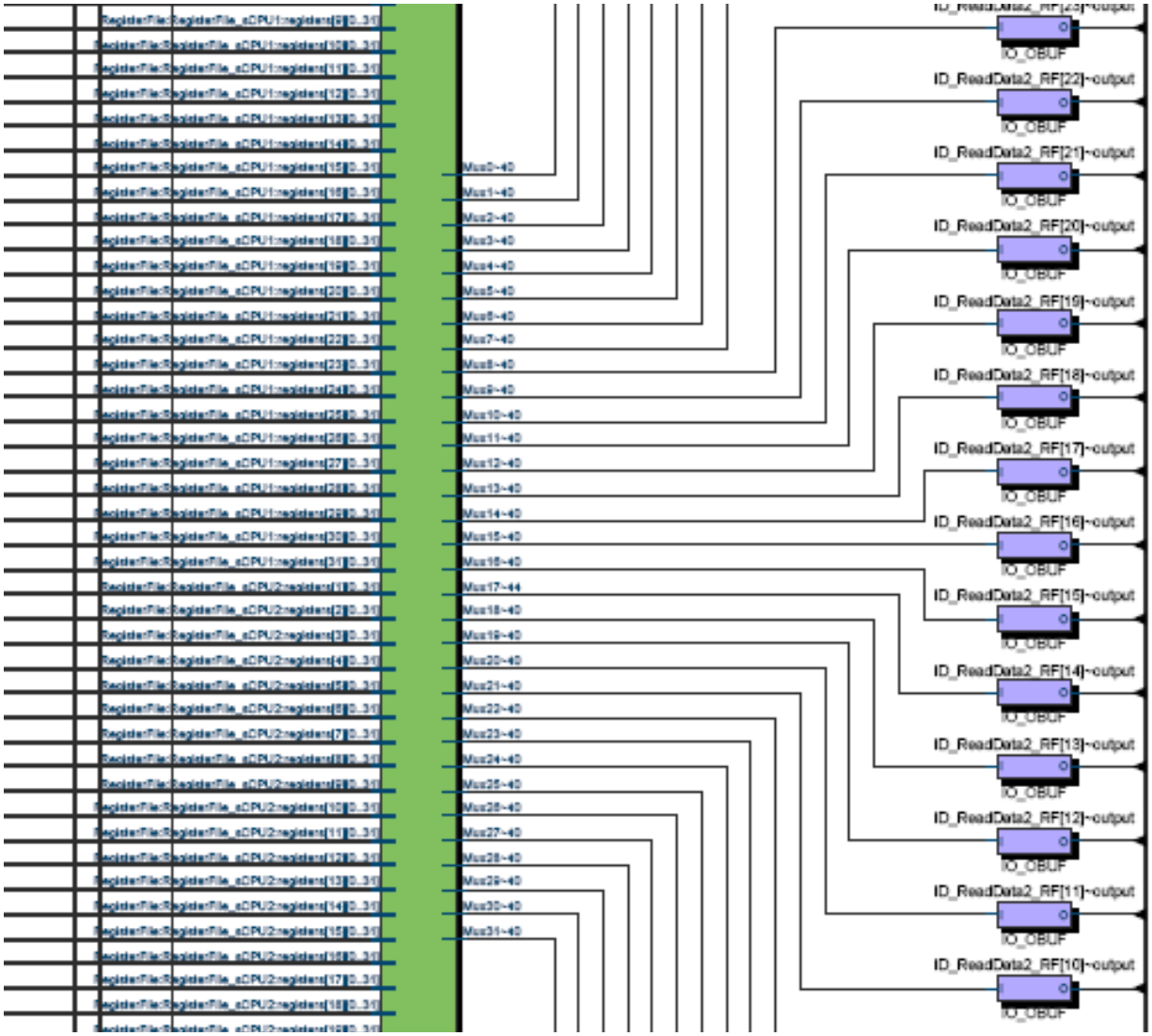
Resource	Usage	%
Logic utilization (ALMs needed / total ALMs on device)	4,218 / 56,480	7 %
ALMs needed [=A-B+C]	4,218	
[A] ALMs used in final placement [=a+b+c+d]	5,062 / 56,480	9 %
[a] ALMs used for LUT logic and registers	241	
[b] ALMs used for LUT logic	3,095	
[c] ALMs used for registers	1,726	
[d] ALMs used for memory (up to half of total ALMs)	0	
[B] Estimate of ALMs recoverable by dense packing	980 / 56,480	2 %
[C] Estimate of ALMs unavailable [=a+b+c+d]	136 / 56,480	< 1 %
[a] Due to location constrained logic	0	
[b] Due to LAB-wide signal conflicts	117	
[c] Due to LAB input limits	19	
[d] Due to virtual I/Os	0	
Difficulty packing design	Low	
Total LABs: partially or completely used	831 / 5,648	15 %
-- Logic LABs	831	
-- Memory LABs (up to half of total LABs)	0	
Combinational ALUT usage for logic	4,116	
-- 7 input functions	64	
-- 6 input functions	3,743	
-- 5 input functions	258	
-- 4 input functions	2	
-- <=3 input functions	49	
Combinational ALUT usage for route-throughs	1,659	
Dedicated logic registers	3,968	
-- By type:		
-- Primary logic registers	3,934 / 112,960	3 %
-- Secondary logic registers	34 / 112,960	< 1 %
-- By function:		
-- Design implementation registers	3,968	
-- Routing optimization registers	0	
Virtual pins	0	
I/O pins	124 / 522	24 %
-- Clock pins	5 / 15	33 %
-- Dedicated input pins	0 / 29	0 %
Global signals	0	
M10K blocks	0 / 686	0 %
Total MLAB memory bits	0	
Total block memory bits	0 / 7,024,640	0 %
Total block memory implementation bits	0 / 7,024,640	0 %
Total DSP Blocks	0 / 156	0 %
Fractional PLLs	0 / 7	0 %
Global clocks	0 / 16	0 %
Quadrant clocks	0 / 88	0 %
Horizontal periphery clocks	0 / 18	0 %
SERDES Transmitters	0 / 120	0 %
SERDES Receivers	0 / 120	0 %
JTAGs	0 / 1	0 %
ASMI blocks	0 / 1	0 %
CRC blocks	0 / 1	0 %
Remote update blocks	0 / 1	0 %
Oscillator blocks	0 / 1	0 %
Hard IPs	0 / 2	0 %
Standard RX PCSs	0 / 9	0 %
HSSI PMA RX Deserializers	0 / 9	0 %
Standard TX PCSs	0 / 9	0 %
HSSI PMA TX Serializers	0 / 9	0 %
Channel PLLs	0 / 9	0 %
Impedance control blocks	0 / 3	0 %
Hard Memory Controllers	0 / 2	0 %
Average interconnect usage (total/H/V)	4.5% / 4.6% / 4.2%	
Peak interconnect usage (total/H/V)	26.9% / 28.3% / 28.9%	
Maximum fan-out	3999	
Highest non-global fan-out	3999	
Total fan-out	42025	
Average fan-out	4.21	



Figură 2-51 RTL Register file8




selMIPS[0]~input
selMIPS[0..1]
0
I
O
IO_IBUF
@O_20665,O_20669
[1-2]
selMIPS[1]~input
1
I
O
IO_IBUF



Figură 2-52 Tehnology map Register file8

3. CONCLUZII

Tabel 3-1 Analiză implementare MPRA, MPRA4, MPRA8

Cyclone V 5CGXFC9E7F31C8	**MPRA**	**MPRA4**	**MPRA8**
Delay added (ns)	19,3	3,7	3,8
Total thermal power dissipation (mW)	530,98	536,17	365,20
Core static thermal power dissipation (mW)	518,51	518,57	348,96
I/O thermal power dissipation (mW)	12,47	17,60	16,24
Total pins	180 (34%)	302 (56%)	302 (56%)
Total registers design implementation	1858	64	64
Logic utilization (în ALMs)	2957 (3%)	17 (1%)	17 (1%)
Clock pins	7 (44%)	9 (56%)	12 (80%)
Global clocks	1 (6%)	1 (6%)	1 (6%)
Maximum fan-out	1858	64	64

Highest non-global fan-out	1798	64	64
Total fan-out	27305	607	608
Difficulty packing design	low	low	low
Combinational ALUT usage for logic	3778	1	1

Tabel 3-2 Puterea consumată de implementarile MPRA pentru 4 şi 8 task-uri

Arhitectura	Ceas mW	Logica mW	Semnale mW	IO mW	MMCM mW	DSP mW	Quiescent mW	Total mW	Utilizare resurse
Blocuri logice									
MPRA4	59,8	56	67	30	83	2,8	3430	3731	12%
MPRA8	230	151	207	58	77	6,1	3444	4175	22%

Quiescent power reprezintă puterea consumată de FPGA la pornire, când dispozitivul este configurat cu logica de lucru, dar fără nici o activitate de clock. Această mărime este o sumă dintre *static power*, care este puterea consumată de dispozitiv la pornire, dar fără logica aplicaţiei şi *design static power*, care reprezintă puterea consumată a logicii implementate fără activitate de ceas-referinţă.

Tabel 3-3 Analiză implementare EXMEM, EXMEM4, EXMEM8

Cyclone V 5CGXFC9E7F31C8	**EXMEM**	**EXMEM 4**	**EXMEM8**
Total pins	242 (45%)	248 (46%)	252 (48%)
Total registers design implementation	117	468	468
Logic utilization (în ALMs)	32	220 (1%)	216 (1%)
Clock pins	6 (38%)	7 (44%)	11 (73%)
Global clocks	1	0	0
Maximum fan-out	118	469	469
Highest non-global fan-out	118	469	469
Total fan-out	913	3172	3170
Logic LABs	18	85 (1%)	90 (2%)
Combinational ALUT usage for logic	8	128	128

La o frecvenţă de 50MHz, *nMPRA* utilizează 606 biţi; replicarea registrelor pipeline la ***n=8* cere 0,59 kB de RAM**, **memoria necesară registrelor generale este de 256 B**, deci **memoria totală cerută de replicarea resurselor *n=8* este de 0,84 kB**, ceea ce este mai mult decât acceptabil ţinând seama că sunt arhitecturi de microcontroler care folosesc pentru uzul general 2,6 MB RAM, fără a mai lua în considerare memoria necesară la implementarea procesorului şi a altor componente hardware integrate pe chip [13]. Deşi *nMPRA* este o arhitectură cu resurse partajate, totuşi **costul este mult mai eficient** decât al altor arhitecturi comerciale.

Un alt mare avantaj al *nMPRA* este implementarea hardware, care elimină dezavantajele implementării HW/SW. Viteza foarte mare de comutare de context a task-urilor (1-3 cicli-procesor) este întâlnită doar la *nMPRA*.

MULŢUMIRI

Această lucrare a fost susținută de proiectul „Performanță durabilă în cercetare doctorală și post-doctorală PERFORM-Contract nr. POSDRU / 159 / 1.5 / S / 138963”, proiect co-finanțat din Fondul Social European prin Programul Operațional Sectorial Resurse Umane 2007-2013.

Listă figuri

Listă tabele

BIBLIOGRAFIE

[1] J. Hennessy, D. Patterson, "*Computer Organization and Design: The Hardware/Software Interface",* Morgan Kaufmann, Waltham, USA, 2012.

[2] E. Dodiu and V.G. Gaitan, "Custom designed CPU architecture based on a hardware scheduler and independent pipeline registers – concept and theory of operation", 2012 IEEE EIT International Conference on Electro-Information Technology, Indianapolis, IN, USA, 6-8 May 2012, ISBN: 978-1-4673-0818-2, ISSN: 2154-0373.

[3] Shi-Hai Zhu, "Hardware Implementation Based on FPGA of Interrupt Management in a Real-time Operating System", Information Technology Journal, 2013.

[4] B.C . Alecsa, "FPGA implementation of a matrix structure for integer division", Proceedings of the 3rd International Symposium on Electrical and Electronics Engineering, Galati, Romania, 2010.

[5] M. Shahbazi, P. Poure, S. Saadate, M.R. Zolghadri, "Fault-Tolerant Five-Leg Converter Topology with FPGA-Based Reconfigurable Control", IEEE Transactions on, vol. 60, no. 6, June 2013.

[6] Gaitan N.C., "Real-time acquisition of the distributed data by using an intelligent system", Electronics And Electrical Engineering, vol. 8, no. 104, pp. 13-18, 2010, ISSN: 1392,1215.

[7] Shawash, J.; Selviah, D.R., "Real-Time Nonlinear Parameter Estimation Using the Levenberg–Marquardt Algorithm on Field Programmable Gate Arrays", Industrial Electronics, IEEE Trans. on, vol.60, no.1, pp.170-176, Jan. 2013.

[8] Shahbazi, M.; Poure, P.; Saadate, S.; Zolghadri, M.R., "FPGA-Based Reconfigurable Control for Fault-Tolerant Back-to-Back Converter Without Redundancy", Industrial Electronics, IEEE Trans. on, vol.60, no.8, pp.3360-3371, Aug. 2013.

[9] Shi-Hai Zhu, "Hardware Implementation Based on FPGA of Interrupt Management in a Real-time Operating System", Information Technology Journal, 2013.

[10] Tran, T.; Ohishi, K.; Yokokura, Y.; Mitsantisuk, C., "FPGA-based High-Performance Force Control System with Friction-Free and Noise-Free Force Observation", Industrial Electronics, IEEE Trans. on, 2013.

[11] Dodiu, E.; Gaitan, V.G.; Graur, A., "Custom designed CPU architecture based on a hardware scheduler and independent pipeline registers – architecture description", IEEE 35'th Jubilee International Convention on Information and Communication Technology, Electronics and Microelectronics, Croatia, 24 May 2012, ISSN: 1847-3946.

[12] Dodiu, E.; Gaitan, V.G., "Custom designed CPU architecture based on a hardware scheduler and independent pipeline registers – concept and theory of operation", 2012 IEEE EIT International Conference on Electro-Information Technology,

Indianapolis, IN, USA, 6-8 May 2012, ISBN: 978-1-4673-0818-2, ISSN: 2154-0373.

[13] V.G. Gaitan, N.C. Gaitan, I. Ungurean, "CPU Arhitecture based on a Hardware Scheduler and Independent Pipeline Registers", unpublished, IEEE Trans. on VLSI System, 2014.

[14] E. Dodiu, V.G. Gaitan, A. Graur, "Custom designed CPU architecture based on a hardware scheduler and independent pipeline registers – architecture description", IEEE 35'th Jubilee International Convention on Information and Communication Technology, Electronics and Microelectronics, Croatia, May 2012.

[15] E. Dodiu and V.G. Gaitan, "Custom designed CPU architecture based on a hardware scheduler and independent pipeline registers – concept and theory of operation", 2012 IEEE EIT International Conference on Electro-Information Technology, Indianapolis, USA, 6-8 May 2012, ISBN: 978-1-4673-0818-2, ISSN: 2154-0373.

[16] E.E. (Ciobanu) Moisuc, Al. B. Larionescu, I. Ungurean, "Hardware Event Handling in the Hardware Real-Time", 18th International Conference on System Theory, Control and Computing to be held in Sinaia, Romania, October 17-19, 2014, ISBN: 978-1-4799-4602-0 ©2014 IEEE.

[17] http://opencores.org/project,mips_enhanced