Contents

[**Instruction Set Architecture of DPUT PROCESSOR** 2](#_Toc7808344)

[**2.1 Design Features** 2](#_Toc7808345)

[**2.1.1 Instruction types** 2](#_Toc7808346)

[**2.1.2 Registers & Flags** 3](#_Toc7808347)

[**2.1.3 Instruction Set** 4](#_Toc7808348)

**CHAPTER 2**

# **Instruction Set Architecture of DPUT PROCESSOR**

The design objective of our ISA is to optimize the ISA for image processing applications and Serial Communication through UART interface. Since image is a 2D-array of data and requires loops to go through pixels and execute operations, we have introduced an instruction called JUMPDEC which can be used to execute loop operations very fast.

DPUT ISA is mainly based on RISC instruction set. ISA contains only 15 simple instructions of equal length and all instructions requires only one clock cycle to execute. As in RISC, each instruction performs specific task and separate instructions to access data from memory (Load/Store Architecture).

**Key features of ISA**

* No of instructions **= 15**
* No of instruction types **= 5**
* Avg. clock cycles per instruction (CPI) **= 1**
* Memory used per instruction **= 2 Bytes**

# **2.1 Design Features**

## **2.1.1 Instruction types**

We have divided instructions into 5 main types based on their operation.

* **I-type** – When this type of instruction is executed processor is mainly in idle state (i.e. no internal operation is done.) UART module control instructions also fall into this category because in our processor UART module is operated as a separate module.
* **A-type** – These instructions are used to perform ALU operations
* **S-type** – These instructions perform shift register operations. Here AC register acts as a shift register
* **T-type** – These instructions are used to move data in one register to another.
* **M-type** – Deal with Memory. Contains two sub categories
* **M1-type** – deal with data memory. LOAD/STORE instructions.
* **M2-type** – deal with instruction memory. These instructions are used to perform branching operations

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Instruction Type** |  | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |  |
| **type – I** |  | **OPCODE** | | | | **X** | **X** | **X** | **X** | **X** | **X** | **X** | **X** | **X** | **X** | **X** | **X** |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **type – A** |  | **OPCODE** | | | | **R** | | | | | **CONST** | | | | | | |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **type - S** |  | **OPCODE** | | | | **X** | **X** | **X** | **X** | **X** | **X** | **X** | **X** | **N** | | | |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **type - T** |  | **OPCODE** | | | | **X** | **S** | | | | | **X** | **D** | | | | |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **type - M1** |  | **OPCODE** | | | | **M** | | | | | | | | | | | |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **type - M2** |  | **OPCODE** | | | | **INST** | | | | | | | | | | | |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

Table 2.1: Instruction Types

* **OPCODE (4 bits)**  – Specifies the operation to be performed.
* **R, D, S (5 bits)** – Specifies Register address
* **CONST (7 bits)** – unsigned integer (0-127)
* **N (4 bits)**  – unsigned integer (0-15) specifies amount of bit shift to be performed
* **M (12 bits)** – Data memory offset
* **INST (12 bits)** – Jump instruction address (0-4095)
* **X (1 bit) -** Unused bit

## **2.1.2 Registers & Flags**

There are 19 registers (including user inaccessible registers) in DPUT Processor. Most of them are 16-bit registers.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Register** | **Name** | **Size (bits)** | **Address** | **Description** |
| PC | PROGRAM COUNTER | 12 | - | Contains the location of the next instruction to be executed |
| IR | INSTRUCTION REGISTER | 16 | - | Holds the current instruction being executed. |
| MBR | MEMORY BASE REGISTER | 16 | 00001 | Contains Base ([15:1] bits) of the dram address |
| MDR | MEMORY DATA REGISTER | 8 | 00010 | Stores the data being transferred in and from dram |
| UARTTX | UART TX REGISTER | 8 | 00011 | Contains data to be sent through UART interface |
| UARTRX | UART RX REGISTER | 8 | 00100 | Contains data came through UART interface |
| LR | LOOP REGISTER | 16 | 00110 | Keep the current cycle of the loop |
| ZR | ZERO REGISTER | 16 | 00000 | Contains constant zero value |
| AC | ACCUMULATOR | 16 | 00101 | Stores results of ALU |
| R0 – R15 | GP REGISTERS | 16 | 10000 –  11111 | General purpose registers - can be used to store values |
| Z | ZERO FLAG | 1 | - | Set to one if AC is zero |
| LRZ | LOOP OVERFLOW FLAG | 1 | - | Set to one if LR is zero |
| TXBUSY | TX BUSY FLAG | 1 | - | Indicate UART module is transmitting data |
| RXREADY | RX READY FLAG | 1 | - | Indicate UART module is being receiving data |

Table 2.1: Registers and Flags

## **2.1.3 Instruction Set**

Typically, all instructions are executed within one clock cycle. But for synchronization purposes we are advised to add NOP instruction after UARTTX/UARTRX is used. Since block RAM module of the FPGA can be accessed within one clock cycle LOAD/STORE instructions are also executed within one clock cycle.

|  |  |  |  |
| --- | --- | --- | --- |
| **Instruction** | **Type** | **Opcode** | **Description** |
| NOP | type - I | 0000 | No Operation |
| UARTSEND | type - I | 1101 | Wait for UART output to complete |
| UARTREAD | type - I | 1110 | Wait for UART input to complete |
| ADD | type - A | 0001 | AC ← AC + ( [ R ]+ [ CONST ] ) |
| SUB | type - A | 0010 | AC ← AC - ( [ R ]+ [ CONST ] ) |
| MUL | type - A | 0010 | AC ← AC \* ( [ R ]+ [ CONST ] ) |
| DIV | type - A | 0100 | AC ← AC / ( [ R ]+ [ CONST ] ) |
| SHR | type - S | 0101 | AC Shift right N bits |
| SHL | type - S | 0110 | AC Shift left N bits |
| LOAD | type - M1 | 0111 | [ M + 2\*MBR ] ← MDR |
| STORE | type - M1 | 1000 | MDR ← [ M + 2\*MBR ] |
| JUMP | type - M2 | 1001 | Jump to [INST] in iram |
| JUMPZ | type - M2 | 1010 | Jump to [INST] in iram IF Z FLAG =1 |
| JUMPDEC | type - M2 | 1011 | Decrement LR by ONE. Jump to [INST] in iram IF LRZ = 0 |
| MOVE | type - T | 1100 | [ D ] ← [ S ] |

Table 2.3 : Instruction Set of DPUT ISA