

# Digi-Arsenal

## Command Book

### **Device name: Primary Logic Gates**

- **AND:**  
Command: AND <Dev-Name> <values [separated by spaces]>  
AND\_OUT Dev-Name [for output]  
Dev-Name.OUT [to use the output ]
- **OR:**  
Command: OR <Dev-Name> <values [separated by spaces]>  
OR\_OUT Dev-Name [for output]  
Dev-Name.OUT [to use the output ]
- **NOT:**  
Command: NOT <Dev-Name> <values [separated by spaces]>  
NOT\_OUT Dev-Name [for output]  
Dev-Name.OUT [to use the output ]
- **NAND:**  
Command: NAND <Dev-Name> <values [separated by spaces]>  
NAND\_OUT Dev-Name [for output]  
Dev-Name.OUT [to use the output ]
- **NOR:**  
Command: NOR <Dev-Name> <values [separated by spaces]>  
NOR\_OUT Dev-Name [for output]  
Dev-Name.OUT [to use the output ]
- **XOR:**  
Command: XOR <Dev-Name> <values [separated by spaces]>  
XOR\_OUT Dev-Name [for output]  
Dev-Name.OUT [to use the output ]
- **XNOR:**  
Command: XNOR <Dev-Name> <values [separated by spaces]>  
XNOR\_OUT Dev-Name [for output]  
Dev-Name.OUT [to use the output ]

### **Device name: Half Adder:**

- Command : HALF\_ADDER <dev-name> [creates the half adder]
  - INPUT <dev-name> <value a> <value b>
  - OUTPUT <dev-name> [display the output]
  - TRUTH\_TABLE <dev-name> [display the truth\_table]
  - dev-name.SUM, dev-name.CARRY[to use the sum , carry bits]

### **Device name: Full Adder:**

- Command : FULL\_ADDER <dev-name> [creates the full adder]
  - INPUT <dev-name> <value a> <value b> <value cin>
  - OUTPUT <dev-name> [display the output]
  - TRUTH\_TABLE <dev-name> [display the truth\_table]
  - dev-name.SUM , dev-name.CARRY[to use the sum , carry bits]

#### **Device name: Binary Parallel bit Adder:**

- Command : PARALLEL\_ADDER <dev-name> <size> [creates the adder]
  - PA\_INPUT <dev-name> <value set a [no space]> <value set b [no space]>
  - PA\_OUTPUT <dev-name> [display the output]  
dev-name-OUTPUT, dev-name.CARRY[to use the entire sum array , carry bit]  
dev-name.SUM[index] [to use the nth bit]

#### **Device name: Half Subtractor:**

- Command : HALF\_SUBTRACTOR <dev-name> [creates the half subtractor]
  - INPUT <dev-name> <value a> <value b>
  - OUTPUT <dev-name> [display the output]
  - TRUTH\_TABLE <dev-name> [display the truth\_table]
  - dev-name.DIFFERENCE, dev-name.BORROW[to use the sum , carry bits]

#### **Device name: Full Adder:**

- Command : FULL\_SUBTRACTOR <dev-name> <size> [creates the full subtractor]
  - INPUT <dev-name> <value a> <value b> <value cin>
  - OUTPUT <dev-name> [display the output]
  - TRUTH\_TABLE <dev-name> [display the truth\_table]
  - dev-name.DIFFERENCE , dev-name.BORROW[to use the difference, carry bits]

#### **Device name: Binary Parallel bit Subtractor:**

- Command : PARALLEL\_ADDER <dev-name> <size> [creates the subtractor]
  - PS\_INPUT <dev-name> <value set a [no space]> <value set b [no space]>
  - PS\_OUTPUT <dev-name> [display the output]  
dev-name-DIFFERENCE , dev-name.BORROW[to use the entire difference array , carry bit]  
dev-name.DIFFERENCE[index] [to use the nth bit]

#### **Device name: N to M line Decoder:**

- Command: DECODER <dev-name> <size n> <mode> [creates the decoder n to m, mode 1 for high active, 0 for low active]
  - INPUT <dev-name> val-set
  - OUTPUT <dev-name>
  - Dev-name-OUTPUT, dev-name.OUTPUT[index] [to use the entire output and seperate bits respaly]

- The output is an array from d0 to dn respectively

### **Device name: Demultiplexer:**

- Command: DEMUX <dev-name> <size n> [creates the demux n to m]
  - INPUT <dev-name> <enable> <val-set-select-lines>
  - OUTPUT <dev-name>
  - Dev-name-OUTPUT, dev-name.OUTPUT[index] [to use the entire output and separate bits resply]
  - The output is an array from d0 to dn respectively

### **Device name: Encoder:**

- Command: ENCODER <dev-name> <size n> <mode> [creates the encoder n to m , mode 1 for first active, 0 for priority]
  - INPUT <dev-name> <val-set >
  - OUTPUT <dev-name>
  - Dev-name-OUTPUT, dev-name.OUTPUT[index] [to use the entire output and separate bits resply]
  - The output is the index of the data line in binary

### **Device name: Multiplexer:**

- Command: MUX <dev-name> <size n> [creates the mux n to m]
  - INPUT <dev-name> <select-lines> <data-set>
  - OUTPUT <dev-name>
  - Dev-name-OUTPUT [to use the bit]
  - The output is the value of the data line

### **Device-name : Flip-flops:**

- Command: D\_FLIPFLOP <dev-name> [creates the flipflop]
  - INPUT dev-name <D value> <Clock value>
  - OUTPUT dev-name
  - dev-name.Q, dev-name.Q', dev-name.CLK [to use Q, Q' , Clock]
- Command: T\_FLIPFLOP <dev-name> [creates the flipflop]
  - INPUT dev-name <T value> <Clock value>
  - OUTPUT dev-name
  - dev-name.Q, dev-name.Q', dev-name.CLK [to use Q, Q' , Clock]
- Command: RS\_FLIPFLOP <dev-name> [creates the flipflop]
  - INPUT dev-name <R value> <S value> <Clock value>
  - OUTPUT dev-name
  - dev-name.Q, dev-name.Q', dev-name.CLK [to use Q, Q' , Clock]
- Command: JK\_FLIPFLOP <dev-name> [creates the flipflop]
  - INPUT dev-name <J value> <K value> <Clock value>
  - OUTPUT dev-name
  - dev-name.Q, dev-name.Q', dev-name.CLK [to use Q, Q' , Clock]

### **Device name : Register:**

- Command: REGISTER <dev-name> <size> [creates the register]
  - REGISTER\_INPUT dev-name value-set
  - LOAD dev-name value-set
  - UNLOAD dev-name value-set
  - CLEAR dev-name [0 to not clear, 1 to clear]
  - REGISTER\_OUTPUT dev-name
  - SHIFT\_LEFT dev-name value
  - SHIFT\_RIGHT dev-name value
  - dev-name.DATA[n], dev-name-DATA, dev-name.serial [to use the output]

### **Device name : Counter:**

- Command: COUNTER <dev-name> <mode [1 for up, 2 for down]> <size> [creates the counter]
  - COUNT dev-name [counts according to the mode]
  - COUNT\_UP dev-name [explicit count up]
  - COUNT\_DOWN dev-name [explicit count down]
  - COUNTER\_LOAD dev-name <value> [parallel lading of the counter]
  - COUNTER\_CLEAR dev-name [clear]
  - COUNTER-OUTPUT dev-name [output]
  - dev-name.VALUE[n], dev-name-VALUE [to use the output]

### **Device name: Comparator:**

- Command: COMPARATOR <dev-name> <size> [creates the comparator]
  - COMP\_INPUT dev-name <value set a> <value set b>
  - COMPARE dev-name
  - COMP\_OUTPUT dev-name
  - COMPARE\_BIT name bit-value {index}
  - COMP\_OUTPUT dev-name
  - COMP\_CLEAR dev-name
  - dev-name.DIFF[n], dev-name-DIFF
  - dev-name.SIMI[n], dev-name-SIMI,
  - dev-name.GREATER[n], dev-name-GREATER  
[to use bits individually and as a whole]

### **Device-name : ROM :**

- Command: ROM <dev-name> <size m> <size n> [creates the rom with size m and n]
  - ROM\_INPUT dev-name <row> <value-set>
  - ROM\_INPUT\_BIT dev-name <row> <index> <value>
  - ROM\_READ dev-name
  - ROM\_READ\_BIT dev-name
  - ROM\_ERASE dev-name
  - ROM\_OUTPUT dev-name
  - **EXPERIMENTAL BETA**
  - dev-name.ROW[index]

- dev-name.BIT[row][index]
- dev-name-ROW[index]

[to use bits individually and as a whole]

### **Device-name: RAM:**

- Command: RAM <dev-name> <size m> <size n> [creates the ram with size m and n]
  - RAM\_INPUT dev-name row value
  - RAM\_INPUT\_BIT dev-name row bit value
  - RAM\_READ dev-name row
  - RAM\_READ\_BIT dev-name row bit
  - RAM\_DEST\_READ dev-name row {destructive read}
  - RAM\_DEST\_READ\_BIT dev-name row bit {destructive bit read}
  - RAM\_LOCK dev-name row
  - RAM\_UNLOCK dev-name row
  - RAM\_OUTPUT dev-name

### **EXPERIMENTAL BETA**

- dev-name.ROW[index]
- dev-name.BIT[row][index]
- dev-name-ROW[index] [to use bits individually and as a whole]

### **Device-name : PLA [EXPERIMENTAL BETA]:**

- Command: PLA <dev-name> <inputs> <products> <outputs> [creates the pla]
  - PLA\_TERM dev-name <product\_index> <index value>
  - PLA\_OR dev-name <outindex> <product\_index> <value>
  - PLA\_OUTNAME dev-name <outindex label>
  - PLA\_INPUT name <binary value>
  - PLA\_INPUT\_BIT dev-name <index> <value>
  - PLA\_OUTPUT dev-name
  - PLA\_OUTPUT\_BIT dev-name <output\_index>
  - PLA\_TABLE dev-name

**NOTE: When using individual bits input them as bit bit example 1 0**

**When using set of bits input them as bit-set bitset example 100 1000**

**When using individual output bit use as “.output” example A.SUM, A.OUTPUT**

**When using output busses or whole arrays use as “-OUTPUT” example A-DIFFERENCE, A-DATA, A-OUTPUT**

**The bit level access for ram , rom and pla are planned , but have not been implemented with respect t memory efficiency kindly use Registers, Counters(parallel load), Comparators accordingly.**

\*\*\*\*\*

**THANK YOU**

\*\*\*\*\*