

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 resp]y]
 - 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 resp]y]
 - 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 to memory efficiency kindly use Registers, Counters(parallel load), Comparators accordingly.

PARALLEL ADDER SUM output is exposed in LSB -> MSB order to reflect carry-propagation behavior. Other bits follow MSB -> LSB order for readability. Representation preferences may evolve on user feedback.

THANK YOU
