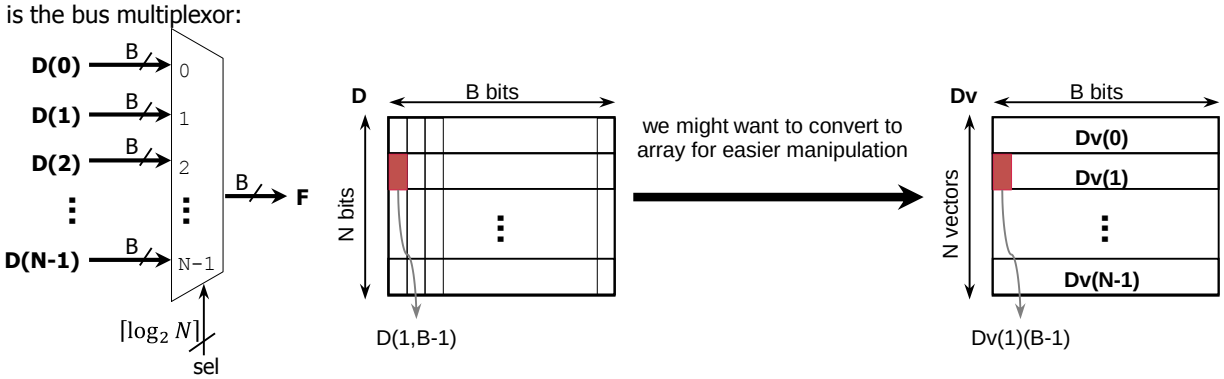
Notes - Unit 3

Advanced Coding Features in VHDL

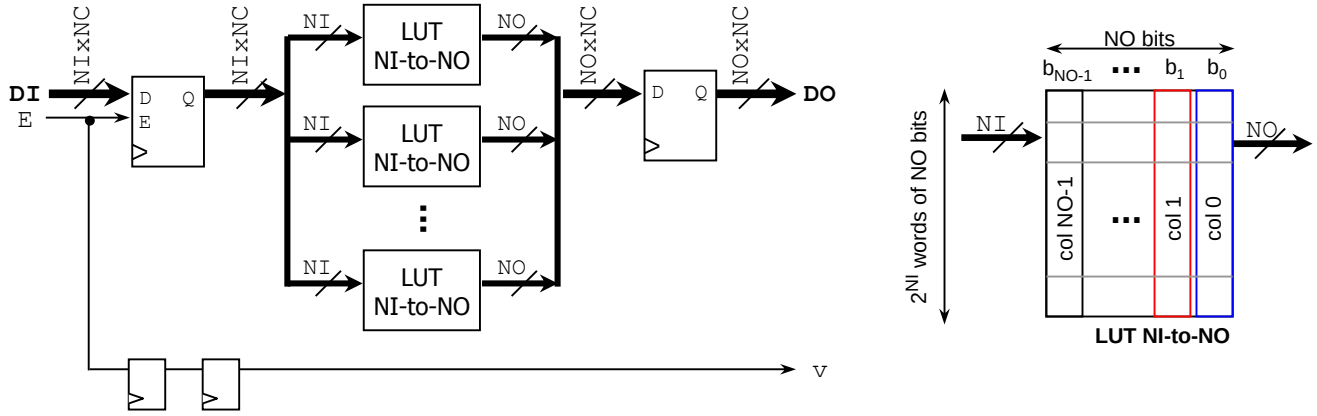
Parameterization

- For-generate: See here for parametric adder.
- If-generate: This is useful whenever we want to pick between two different components. This is NOT a multiplexor, since the decision of which circuit is implemented is made at the Synthesis Level.
- `std_logic_2d`: VHDL does not allow for generic arrays as inputs in your entity section. A common example that requires this is the bus multiplexor:



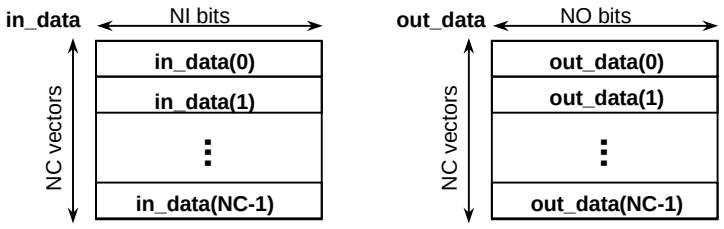
LUT Example

- **LUT with I/O registers**: VHDL code.



- Use of custom-defined data types:

```
chunkI is array (NC-1 downto 0) of std_logic_vector (NI-1 downto 0);
chunkO is array (NC-1 downto 0) of std_logic_vector (NO-1 downto 0);
signal in_data: chunkI;
signal out_data: chunkO;
```



- Recursive VHDL structure (see `LUT_NIto1.vhd`).
- Reading text files for Synthesis (see `LUT_NItoNO.vhd`): We can load each LUT $NI-to-NO$ with a particular function (this is selected by the parameter F). The text file has 2^{NI} NO-bit words for every desired function. In VHDL, we use an *impure function* to read the text file (the impure qualifier is required by the VHDL syntax so that the VHDL function is dependent on an external text file). The `std.textio` library is also required.
- Reading/Writing text files for Simulation: In Vivado 2015.2, make sure to include the input text file as a Simulation Source. The output text file will be written in `/sim/sim_1/behav`.

ADDER TREE

- Parameterized signed/unsigned adder tree with enable: VHDL code.
- Use of custom-defined data types (3D arrays, `std_logic_2d`):

```
chunk_3D is array (natural range <>, natural range <>) of std_logic_vector (T-1 downto 0);
signal yf: chunk_3D (LV downto 0, N-1 downto 0);
```