Design

Captures attributes of an LWC hardware implementations to enable automated benchmarking and validation.

Properties:

- **name** *(string)*: A unique identifier for the design. It can consist of English letters, digits, dashes, and underscores and must start with a letter.
- **description** *(string)*: A short description of the design.
- **authors** *(array of strings)*: A list of names of author(s) for this implementation.
- **url** *(string)*: Uniform Resource Locator pointing to a webpage or source repository associated with this design or its author(s).
- **version** *(string)*: Design version. *Examples:* `"0.0.1"`
- **rtl**: Details of the synthesizable RTL design. Cannot contain additional properties.
  - **sources** *(array of strings)*: Non-empty list of HDL source file paths in correct compilation order. All paths must be relative to the location of the configuration file. Path separator is `/` (slash) on all platforms. Paths are case-sensitive and should not contain whitespaces. HDL language is inferred from the file extension. NOTE: Xeda `sources` fields can be an `object` containing meta-data such as HDL type and version.
  - **includes** *(array of strings)*: Non-empty list of HDL include file paths, such as Verilog headers. Include files are not directly compiled but need to be present for elaboration of design. Order is arbitrary. All paths must be relative to the location of the configuration file. Path separator is `/` (slash) on all platforms. Paths are case-sensitive and should not contain whitespaces.
  - **top** *(string)*: Name of top-level RTL entity/module. *Default:* `LWC`
  - **clock**: Top level RTL clock signal. Only a single clock is supported by LWC API.
    - **port** *(string)*: Name of the top-level RTL clock input. *Default:* `clk`
  - **reset**: Top level reset signal. Only a single reset is supported by LWC API.
    - **port** *(string)*: Name of top-level RTL reset input. *Default:* `reset`
    - **active_high** *(boolean)*: Polarity of the reset signal. Active-high (positive) if true, otherwise active-low. *Default:* `true`
    - **synchronous** *(boolean)*: Reset is synchronous (positive) if true, otherwise asynchronous. *Default:* `true`
  - **parameters**: Top-level design parameters or generics specified as a key-value map. The default value of each parameter is overridden by synthesis tool, simulator, testbench, or wrapper. For the best tool compatibility, we only support integer and string values.
  - **language**: Information about Hardware Description/Design Language(s) used.
    - **vhdl**: VHDL language standard supported by all VHDL source files. VHDL files should have a `.vhd` or `.vhdl` extension.
      - **version** *(string)*: *Supported values:* `1993`, `2000`, `2002`, `2008` *Default:* `2002`
      - **synopsys** *(boolean)*: Use of non-standard Synopsys packages in IEEE namespace. (NOTE: The use of non-standard `IEEE` libraries is *strongly* discouraged!). *Default:* `false`
    - **verilog**: Verilog language standard supported by all Verilog source files. Covers pre-SystemVerilog standards. Verilog files should have a `.v` extension.
      - **version** *(string)*: *Supported values:* `1995`, `2001` *Default:* `2001`
    - **systemverilog**: SystemVerilog (IEEE 1800-2005 and onwards) language standard supported by all SystemVerilog files. Verilog files should have a `.sv` extension.
      - **version** *(string)*: *Supported values:* `2005`, `2009` *Default:* `2009`

- **tb**: Details of test-bench used for verification of top-level design. [Optional]. Cannot contain additional properties.
  - **sources** *(array of strings)*: Source files used only for verification. Should *not* contain any of the files included in 'rtl.sources'.
  - **includes** *(array of strings)*: HDL include file paths.
  - **top** *(string)*: Name of top-level test entity or module.
  - **parameters**: Testbench parameter or generics specified as a key-value map. The default value of each parameter is overridden by the simulator. For the best tool compatibility, we only support integer and string values.
  - **language**: Information about HDL or programming languages used in the testbench.
    - **vhdl**: VHDL language standard supported by all VHDL source files. VHDL files should have a `.vhd` or `.vhdl` extension.
      - **version** *(string)*: *Supported values:* `1993`, `2000`, `2002`, `2008` *Default:* `2002`
      - **synopsys** *(boolean)*: Use of non-standard Synopsys packages in IEEE namespace. (NOTE: The use of non-standard `IEEE` libraries is *strongly* discouraged!). *Default:* `false`
    - **verilog**: Verilog language standard supported by all Verilog source files. Covers pre-SystemVerilog standards. Verilog files should have a `.v` extension.
      - **version** *(string)*: *Supported values:* `1995`, `2001` *Default:* `2001`
    - **systemverilog**: SystemVerilog (IEEE 1800-2005 and onwards) language standard supported by all SystemVerilog files. Verilog files should have a `.sv` extension.
      - **version** *(string)*: *Supported values:* `2005`, `2009` *Default:* `2009`
    - **python**
      - **version** *(string)*
      - **framework** *(string)*: *Supported values:* `cocotb`
- **lwc**: LWC-specific meta-data.
  - **algorithms** *(string or array of strings)*: LWC AEAD/Hash algorithm(s) supported by this design. Should follow SUPERCOP naming conventions and uniquely identify the scheme's variant and version. In case of duplicates, the second instance indicates support for AEAD and Hash algorithms with the same name. *Examples:* `["giftcofb128v1"]`, `["romulusn1v12"]`, `["gimli24v1", "gimli24v1"]`
  - **input_sequence**: Order in which different input segment types should be fed to PDI.
    - **encrypt** *(array of strings)*: *Default:* `['npub', 'ad', 'pt', 'tag']`
    - **decrypt** *(array of strings)*: *Default:* `['npub', 'ad', 'ct', 'tag']`
  - **pt_block_bits** *(integer)*: Algorithm's size of plaintext/ciphertext 'blocks' in bits. This is the number of bits that the algorithm operates upon during its basic operations. *Minimum:* `1`
  - **ad_block_bits** *(integer)*: Algorithm's size of associated-data 'blocks' in bits. This is the number of bits that the algorithm operates upon during its basic operations. *Minimum:* `1`
  - **key_bits** *(integer)*: *Default:* `128`
  - **npub_bits** *(integer)*: *Default:* `128`
  - **tag_bits** *(integer)*: *Default:* `128`
  - **hash_bits** *(integer)*: *Default:* `128`
  - **ports**: Description of LWC ports.
    - **pdi**: Public Data Input port.
      - **bit_width** *(integer)*: Width of each share of PDI data (bits). Width of 'pdi*data*' *port is* '*pdi.bit*width * pdi.shares'. *Minimum:* `8` *Maximum:* `32`
      - **num_shares** *(integer)*: Number of PDI shares. Default is 1 (unprotected). *Minimum:* `1` *Maximum:* `8` *Default:* `1`
    - **sdi**: Secret Data Input port.
      - **bit_width** *(integer)*: Width of each share of SDI data (bits). Width of `sdi_data` port is `sdi.bit_width * sdi.shares`. Default is the same value as `pdi.bit_width`. *Minimum:* `8` *Maximum:* `32`
      - **num_shares** *(integer)*: Number of SDI shares. Default is the same as `pdi.num_shares`. *Minimum:* `1` *Maximum:* `8`

- **do**: Data Output port. Width of each share is always the same as `pdi.bit_width`.
  - **num_shares** *(integer)*: Number of `DO` output shares. Default is the same as `pdi.num_shares`. *Minimum:* `1` *Maximum:* `8`
- **rdi**: Random Data Input port.
  - **bit_width** *(integer)*: *Minimum:* `0` *Maximum:* `2048` *Default:* `0`

- **sca_protection**: Implemented countermeasures against side-channel attacks. Cannot contain additional properties.
  - **attacks** *(array of strings)*: Family and type of side-channel analysis attack(s) this design is claims to be secure against. *Examples:* `["spa", "dpa", "cpa", "timing"]`, `["dpa", "sifa", "dfia"]`
  - **dpa_order** *(integer)*: Claimed order of protectcion against differential power analysis. 0 means unprotected. *Minimum:* `0` *Maximum:* `7` *Default:* `0`