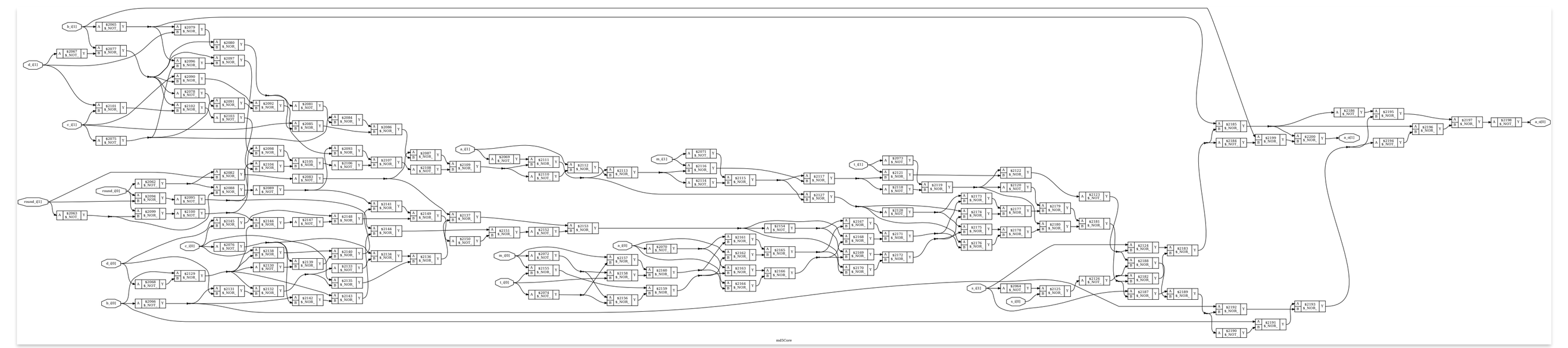

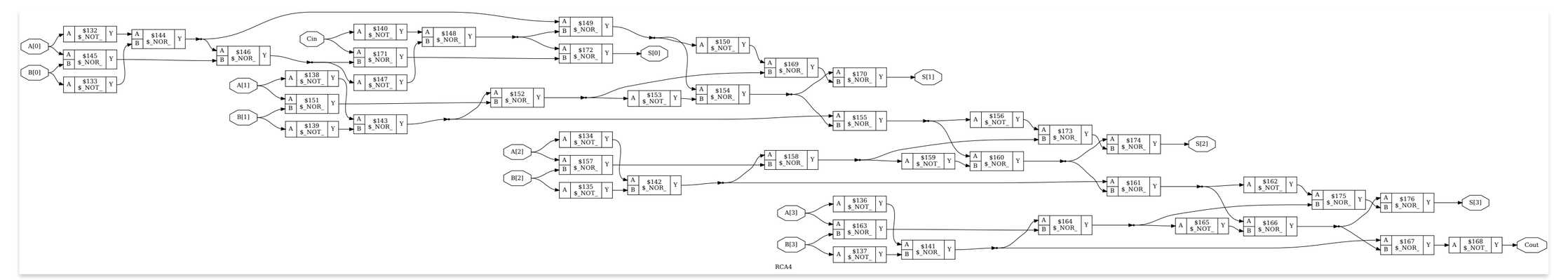

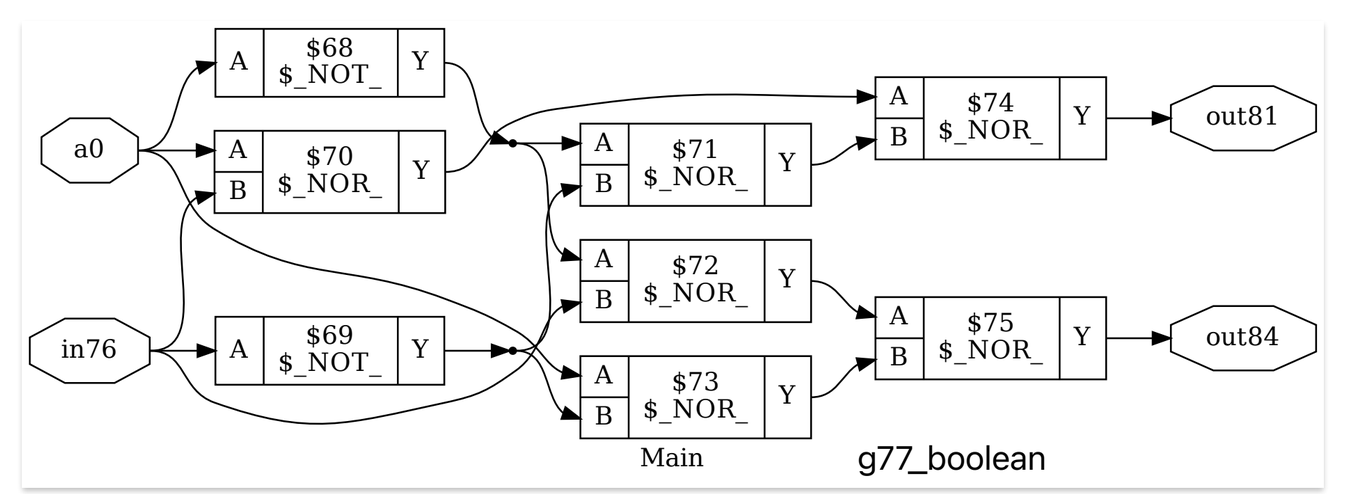

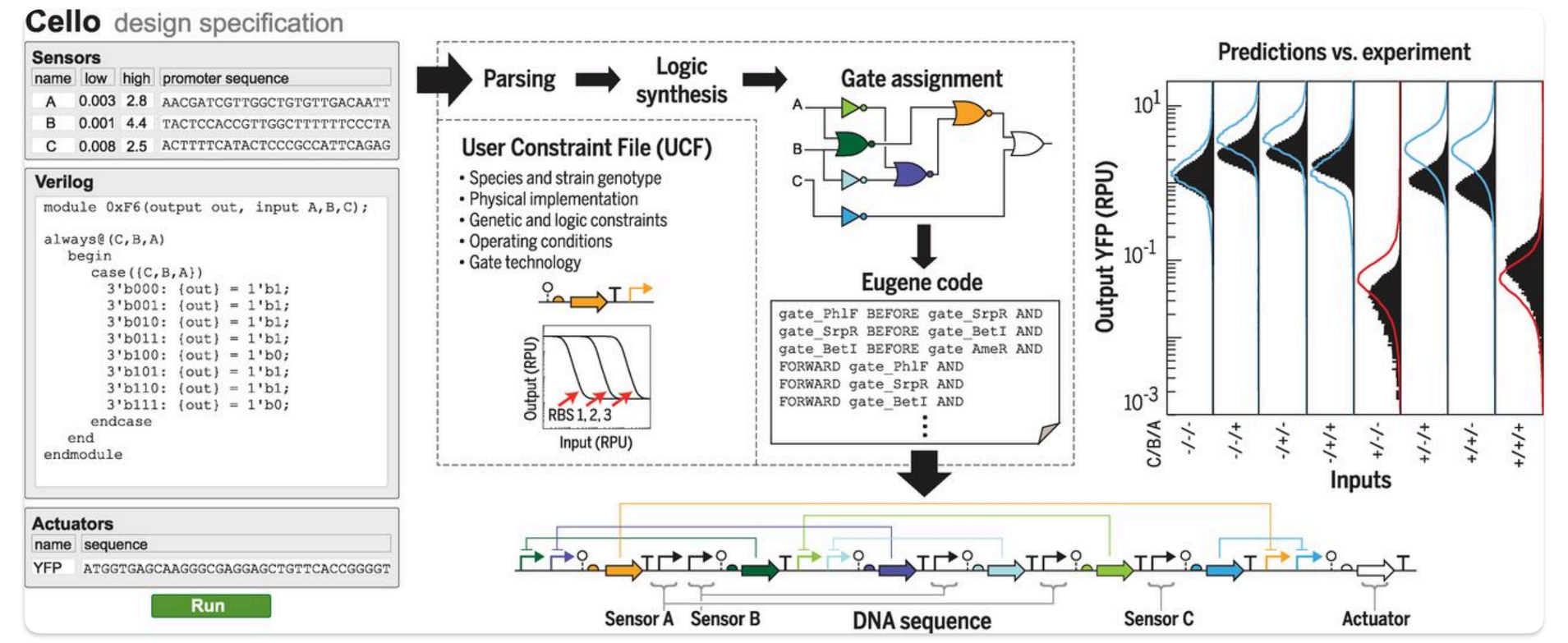

Figure 4.8 shows an example for the *output* collection. Each entry of an output collection consists of: a name, and DNA sequence. The name is used for reference; and, the DNA sequence is used when building the DNA sequence output of Cello.

```
{
        "collection": "quorum_pair",
        "name": "NONCE_Q_Q0",
        "input" :{
                "name": "NONCE_Q_Q0",
                "off_reu": 0.013412352,
                "on_reu": 3.976450405,
                "K": 0.042456227,
                "n": 1.33128967,
                "a": 0.963695611,
                "b": 1.0,
                "seq":
"TATATATATATATATAATAGCTTCTTACCGGACCTGTAGGATCGTACAGGTTTACGCAAG
AAAATGGTTTGTTACTTTCGAATAAA"
        },
        "output" :{
                "name": "NONCE_Q_Q0",
                "seq":
"GAAGCTGTCACCGGATGTGCTTTCCGGTCTGATGAGTCCGTGAGGACGAAACAGCCT
CTACAAATAATTTTGTTTAATACTAGAGAAAGAGGGG..."
        },
}
```

Figure 4.9: an example for the *quorum_pair* collection

Figure 4.9 shows an example for the *quorum_pair* collection. Each entry of an *quorum_pair* collection consists of two entries: 1) input referring to quorum sensors and 2) output referring to quorum inducers. The input consists of: a name, off_reu, on_reu, K, n, a, b and DNA sequence. The name is used for reference; the off_reu and on_reu is used for gate assignment; the K, n, a, and b values are used in Jonghyeon's (shinx097@mit.edu) tandem promoter model for gate assignment; and, the DNA sequence is used when building the DNA sequence output of Cello. The output consist of: a name, and DNA sequence. The name is used for reference; and, the DNA sequence is used when building the DNA sequence output of Cello.

The algorithm assigns quorum pairs using a round-robin mechanism and ensures that a either the sensor or inducer of the quorum pair is assigned to a cell (block from the perspective of partitioning).

The algorithm assigned the *GateType* attribute with promoters separated with colon followed by the repressor. For example, the assignment *pBAD:pLux:P2_PhlF* refers to an implementation with promoters *pBAD* and *pLux*, and, the gate implementation *P2_PhlF*.

See folders you vie

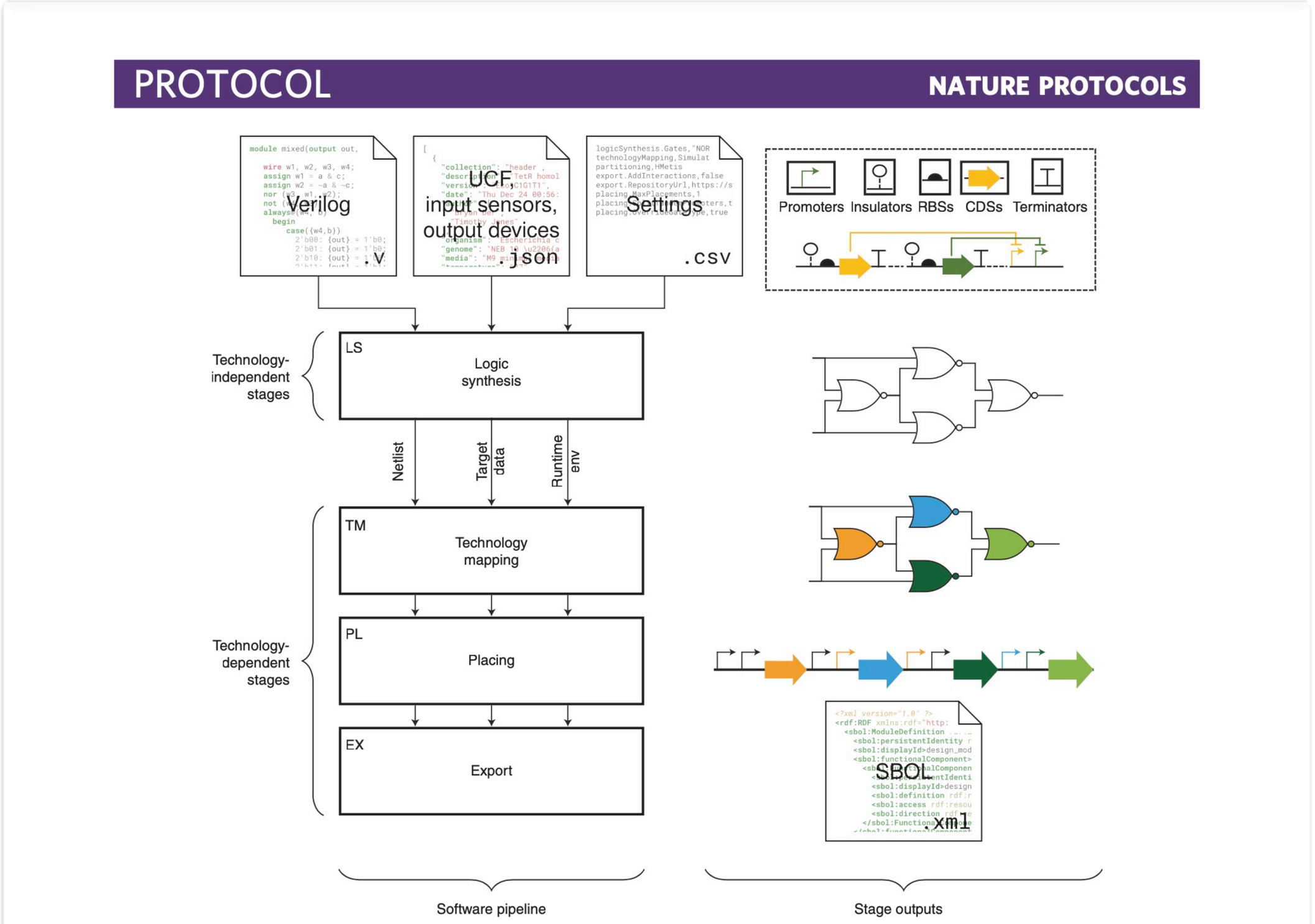

Fig. 1 | Basic software architecture of Cello 2.0. The workflow is divided into four stages, each with one or more algorithms to perform that stage's tasks. Cello ingests different input files: a Verilog file, the 'target data' (a composition of the UCF, the input sensor file and the output device file), and a stage configuration file. Each stage produces some characteristic output: the LS stage produces a Boolean gate-level network ('netlist') that implements the circuit behavior described by the Verilog code and the 'runtime environment' (env) of the synthesis, the TM stage assigns biological gates to each Boolean gate from the previous stage, the PL stage orders gates and parts into a linear DNA sequence and the EX stage produces interchange files (SBOL). In the dashed-line box, the legend shows the different modular components ('DNA parts') that compose Cello's transcriptional units ('DNA logic gates'), i.e., promoters, insulators, ribosome binding sites (RBSs), coding sequences (CDSs) and terminators.