Automated Design of Robust Genetic Circuits: Structural Variants and Parameter Uncertainty Supporting Information

Tobias Schladt,[†,¶] Nicolai Engelmann,[†,¶] Erik Kubaczka,[†,¶] Christian Hochberger,[†] and Heinz Koeppl[*,†,‡]

†Department of Electrical Engineering and Information Technology, TU Darmstadt, Darmstadt, 64283, Germany

‡Centre for Synthetic Biology, TU Darmstadt, Darmstadt, 64283, Germany

¶The authors contributed equally to this research.

E-mail: heinz.koeppl@bcs.tu-darmstadt.de

A Algorithms

A.1 Enumeration of Structural Circuit Variants

The following pseudo codes depict the enumeration and pruning procedure for synthesizing structural circuit variants and its recursive enumeration kernel.

input : A gate library $\mathcal{L}$ containing gate types $\mathcal{S}$, a Boolean function specification ϕ, maximum circuit weight ω and depth δ
output: A set C_ϕ of circuits implementing ϕ covered by $\mathcal{L}$

```
                                                        Initialization
new C ← ∅; C_φ ← ∅;
new γ ← ∅; γ_m ← ∅;      Circuits are arrays of rows of gates and terminal elements
new b ∈ 𝔹;
                                             Enumerate with online pruning
enumerate(γ, S, ω, δ, n(φ), C);     n() returns the support size of a Boolean function

            Wire combinations of primary inputs P and circuit inputs I
foreach γ ∈ C, m ∈ M ⊂ P × I do
    γ_m ← wire_inputs(γ, m);
                                          Match circuit and target function
    if ¬(γ_m ⊣ φ) then
        continue;
    end if
                                                      Remove redundancies
    foreach v ∈ V(γ_m) do
        foreach u ∈ V(γ_m) do
                     f() returns the function of a gate with respect to P
            if v ≠ u ∧ f(v) = f(u) then
                substitute_gate(v, u);          Replaces u by a fan out of v
            end if
        end foreach
    end foreach
                                         Final check of library constraints
    b ← true;
    foreach s ∈ S do
        if |v ∈ V(γ_m) : s_v = s| > |g ∈ L : s_g = s| then
            b ← false;
            break;
        end if
    end foreach
                  If circuit is implementable with L, add to output set
    if b then
        C_φ ← C_φ ∪ γ_m;
    end if
end foreach
return C_φ;
```

input: A circuit γ, gate types $\mathcal{S}$, maximum circuit weight ω and depth δ, the minimum number of inputs n

inout: A a set of fan-out free circuits C

```
Function enumerate(γ, S, ω, δ, n, C)
    new l ← length(γ);                              l: depth of the circuit γ
    new I_γ ← get_unconnected_inputs(γ);             I_γ: set of unconnected gate inputs of γ
                                                     Abort criterion
    if l ≥ δ then
        return;
    end if
    Iterate permutations of gates that match the number of unconnected inputs
    foreach r ∈ R ⊂ {S, ∅}! : |R| = max(|I_γ|, 1) do
        new γ′ ← γ;                                  Copy γ and add new row of gates
        new I_γ′ ← get_unconnected_inputs(γ′);
        γ′[l] ← r;
                                                     Prune circuits that are too big
        if ω_γ′ > ω then
            return;
        end if
                              Check, if γ′ supports ϕ and prune isomorph circuits
        if |I_γ′| ≥ n ∧ ¬∃γ ∈ C : γ′ ≃ γ then
            C ← C ∪ γ′;
        end if
        enumerate(γ′, S, ω, δ, n, C);                Recurse
    end foreach
```

A.2 Generation of an Equivalent Envelope-Free Circuit

The equivalent envelope-free circuit is just a 'common' circuit C^*, which is capable of carrying out the propagation of intervals through an original circuit C. Exploiting the monotonicity of all gate transfer functions in an extended gate library $\mathcal{L}_e$, which contains tuples $(g, \overline{g}, \underline{g}) \in \mathcal{L}_e$ for each $g \in \mathcal{L}$, the circuit C^* contains twice as many gates, only twice as many edges and its result is valid on the whole input domain.

For details on envelopes and the interval-based scoring, please refer to the Methods section from the original manuscript.

input : A circuit $C \equiv (\gamma, a)$, a gate library $\mathcal{L}_e$ with additional envelope specifications
output: A circuit $C^* \equiv (\gamma^*, a^*)$ propagating the intervals of C

Initialization

new $V^* \leftarrow \varnothing$; $E^* \leftarrow \varnothing$; $a^* \leftarrow \varnothing$;

Build new circuit

new $D \leftarrow \varnothing$; *Helper D associates $v^* \in V^*$ with $v \in V$*
$\gamma \equiv (V, E)$ consists of vertices V and edges $E \subset V \times V$
foreach $v \in V$ **do**
  new v_h^*; v_l^*;
  $V^* \leftarrow V^* \cup \{v_h^*, v_l^*\}$;
  Associate new nodes with old ones to connect correctly later
  $D \leftarrow D \cup \{v, \{v_h^*, v_l^*\}\}$;
  Elements $(v, g) \in a$ consist of a $v \in V$ and a $g \in \{g, g_h, g_l\} \in \mathcal{L}_e$
  $a^* \leftarrow a^* \cup (v_h^*, g_h)$;
  $a^* \leftarrow a^* \cup (v_l^*, g_l)$;
end foreach
foreach $v \in V$ **do**
  Add crossed incoming edges between corresponding node pairs in V^*
  foreach $e \in E$ **where** $e = (w, v)$, $w \in V$ **do**
    $\{v_h^*, v_l^*\} \leftarrow$ `get_associated`(v, D);
    $\{w_h^*, w_l^*\} \leftarrow$ `get_associated`(w, D);
    $E^* \leftarrow E^* \cup (w_h^*, v_l^*)$;
    $E^* \leftarrow E^* \cup (w_l^*, v_h^*)$;
  end foreach
end foreach

Done. Return new circuit

new $\gamma^* \leftarrow (V^*, E^*)$;
new $C^* \leftarrow (\gamma^*, a^*)$;
return C^*

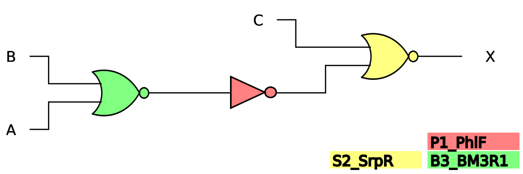
B Synthesized Circuit Designs

B.1 Structural Variants, Classical Assignment Optimization

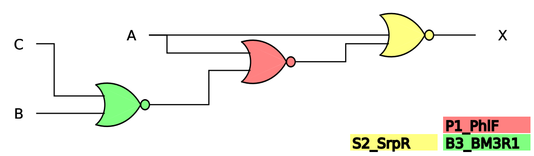
In the following, circuits synthesized by Cello and their structural variants synthesized by the proposed method are depicted, together with the optimal gate assignment found using the Cello score. Their corresponding final Cello scores are written below each. The diagrams have been automatically generated from the synthesis results.

Function 0x80

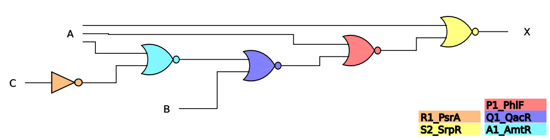
Cello	Structural Variants



Score: 610.96



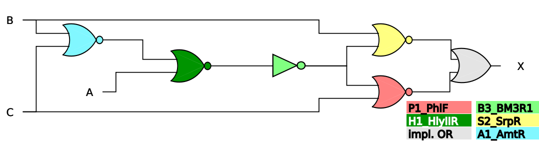
Score: 654.26



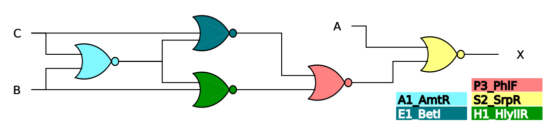
Score: 654.26

Function 0x60

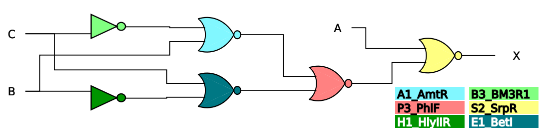
Cello	Structural Variants



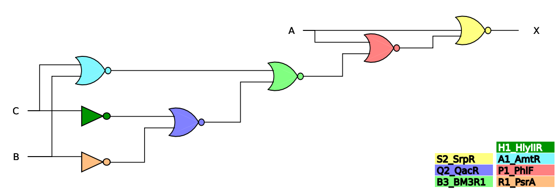
Score: 145.77



Score: 550.78



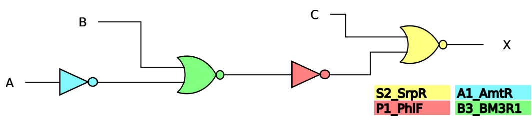
Score: 554.00



Score: 654.25

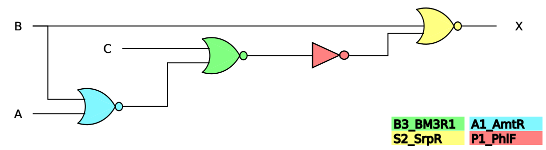
Function 0x08

Cello



Score: 610.96

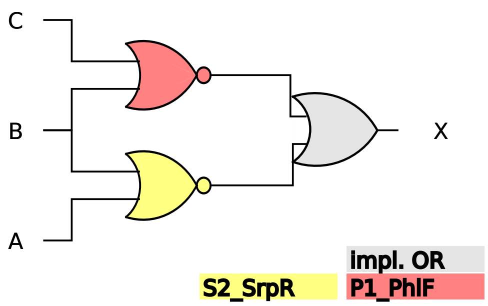
Structural Variant



Score: 671.49

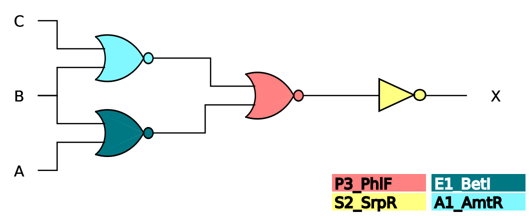
Function 0xC8

Cello



Score: 161.28

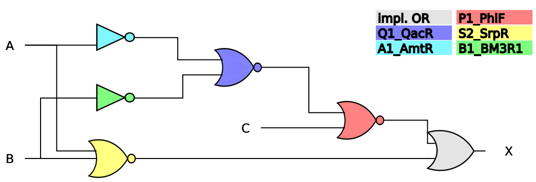
Structural Variant



Score: 598.99

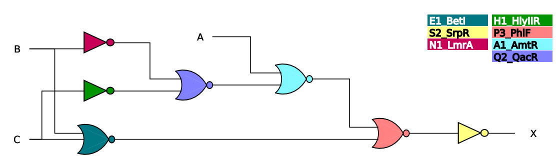
Function 0xE8

Cello



Score: 161.28

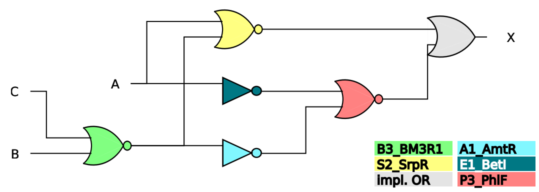
Structural Variant



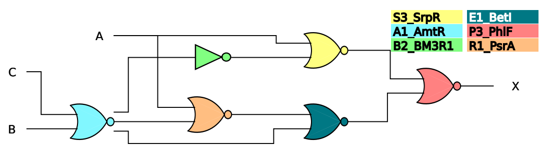
Score: 597.15

Function 0x78

Cello

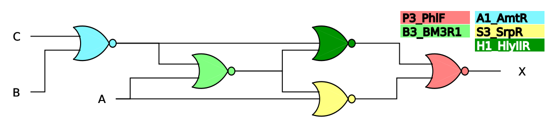


Score: 83.54

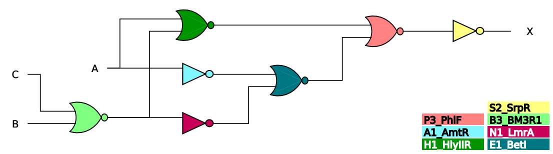


Score: 324.49

Structural Variants



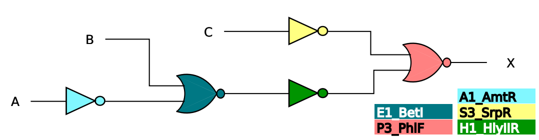
Score: 308.10



Score: 593.95

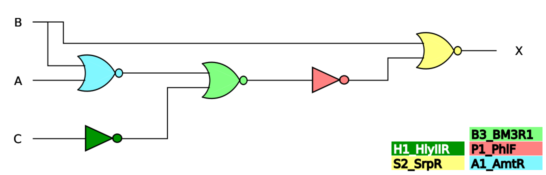
Function 0x04

Cello



Score: 327.87

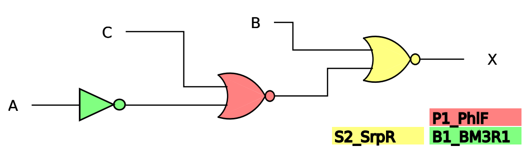
Structural Variant



Score: 671.47

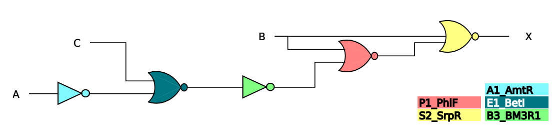
Function 0xC4

Cello



Score: 671.16

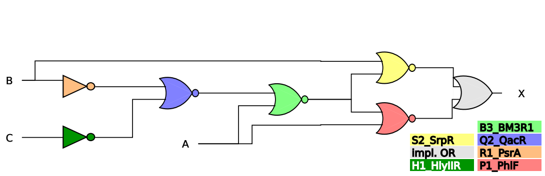
Structural Variant



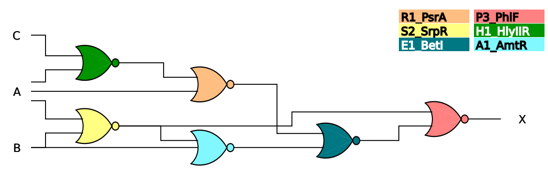
Score: 671.43

Function 0x1C

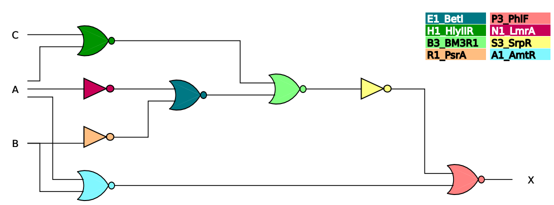
Cello	Structural Variants



Score: 146.34



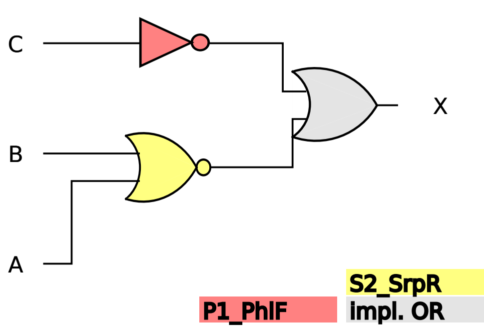
Score: 325.05



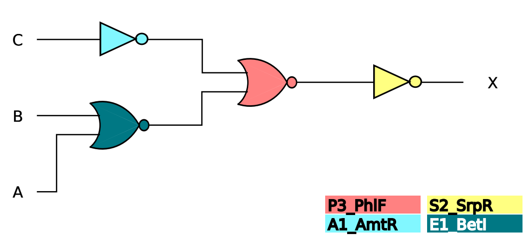
Score: 327.34

Function 0xEA

Cello	Structural Variant



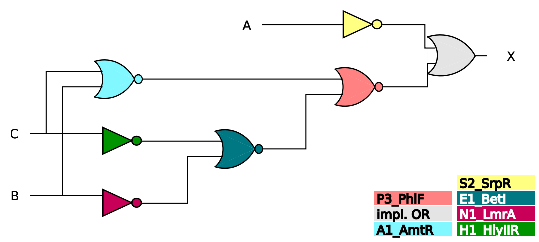
Score: 161.28



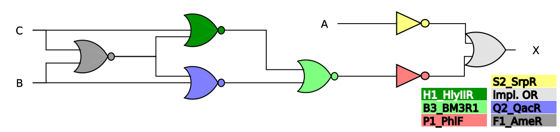
Score: 598.99

Function 0xF6

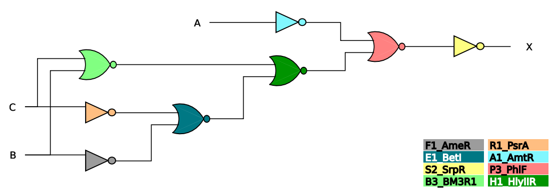
Cello	Structural Variants



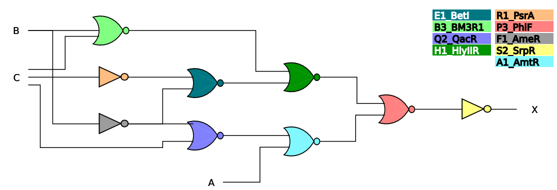
Score: 91.18



Score: 161.51



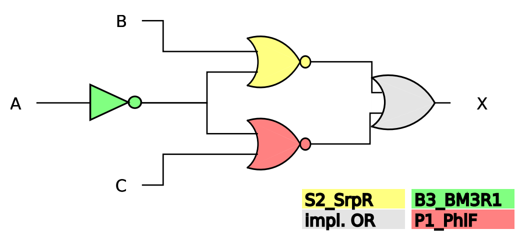
Score: 593.91



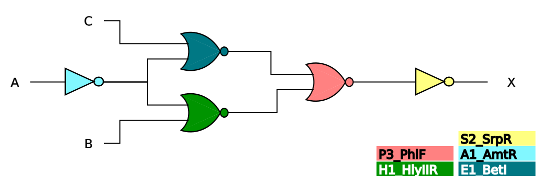
Score: 593.91

Function 0x0E

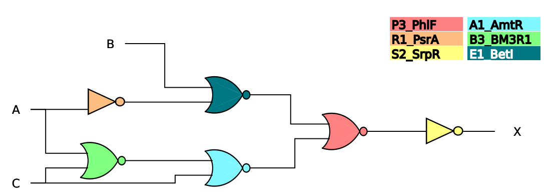
Cello	Structural Variants



Score: 146.36



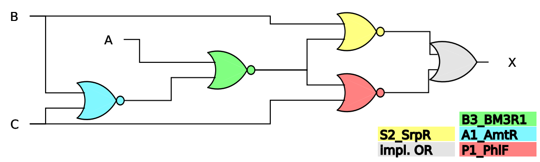
Score: 595.70



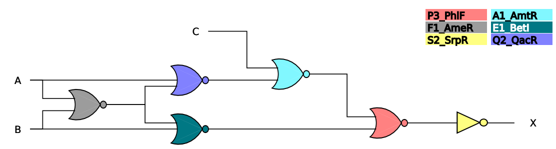
Score: 597.02

Function 0x8E

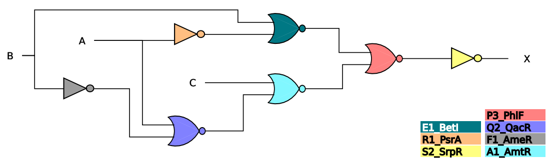
Cello	Structural Variants



Score: 146.10



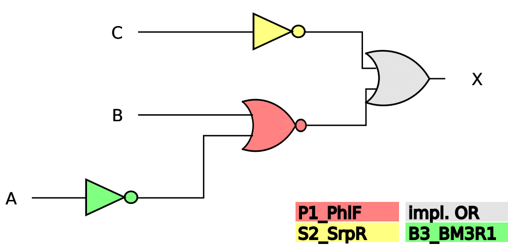
Score: 597.32



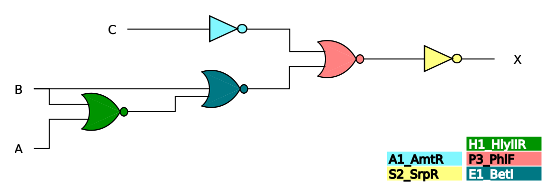
Score: 597.49

Function 0xAE

Cello	Structural Variant



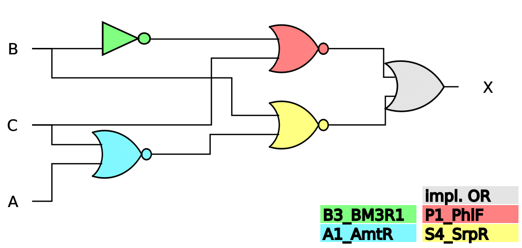
Score: 159.09



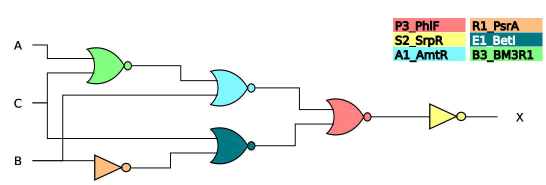
Score: 598.69

Function 0x6E

Cello	Structural Variant



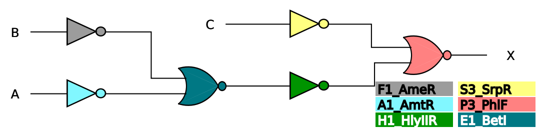
Score: 87.07



Score: 597.17

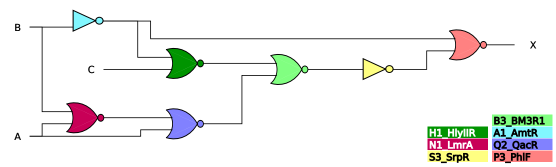
Function 0x01

Cello



Score: 327.66

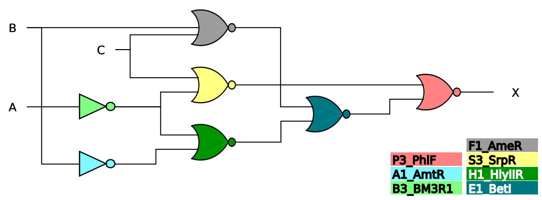
Structural Variants



Score: 328.18

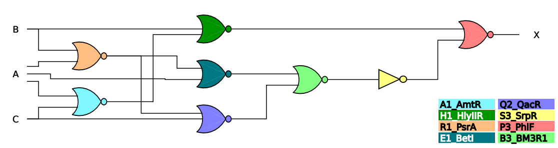
Function Consensus

Cello

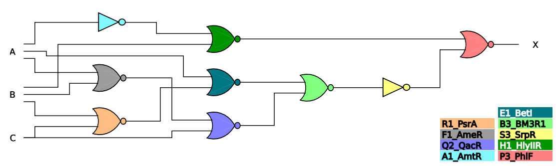


Score: 308.68

Structural Variants



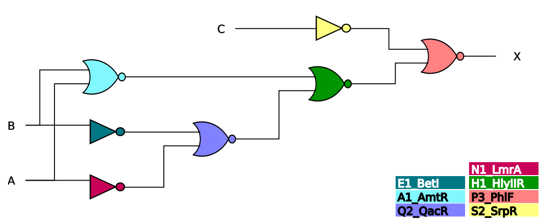
Score: 327.29



Score: 327.31

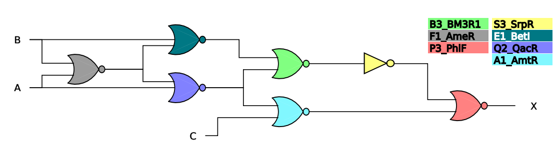
Function 0x41

Cello

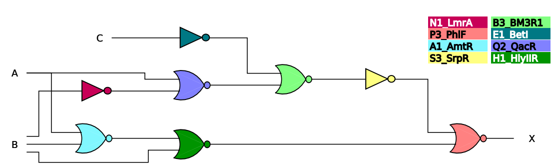


Score: 316.46

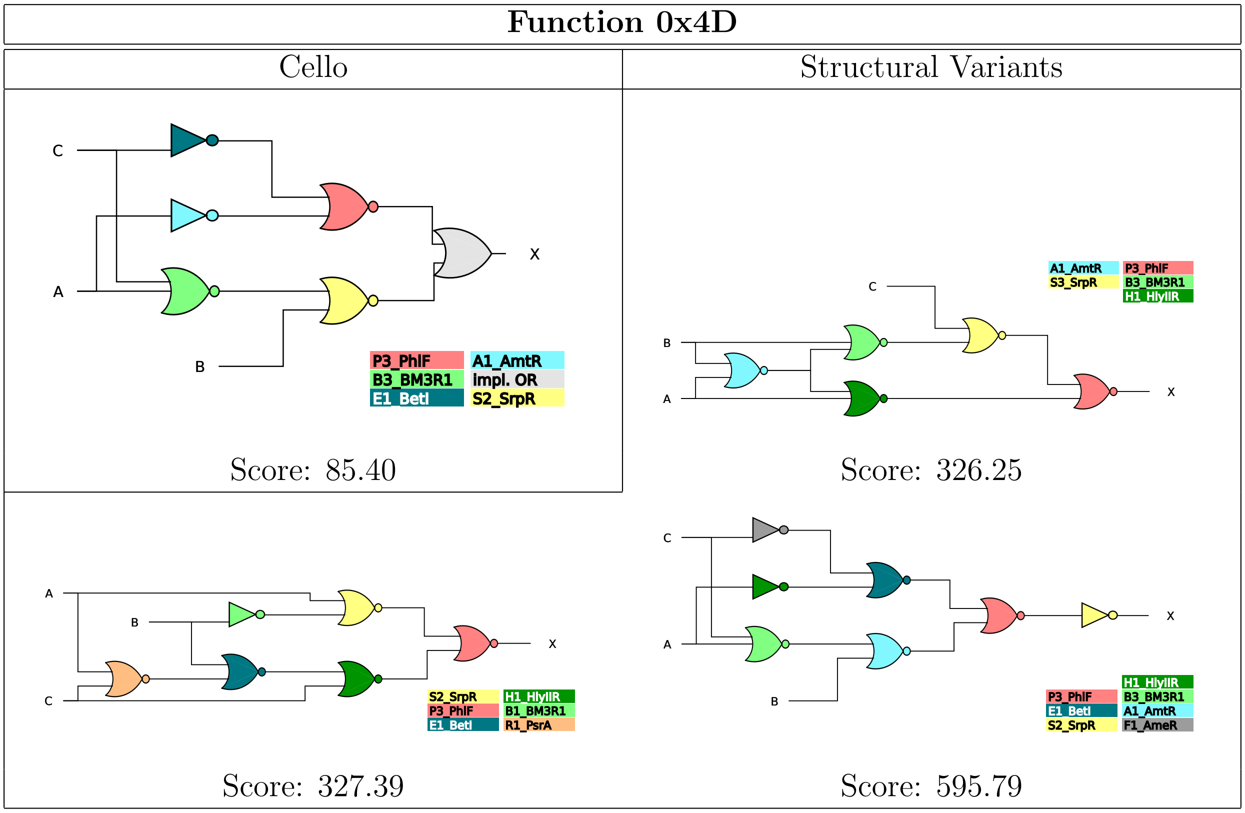
Structural Variants

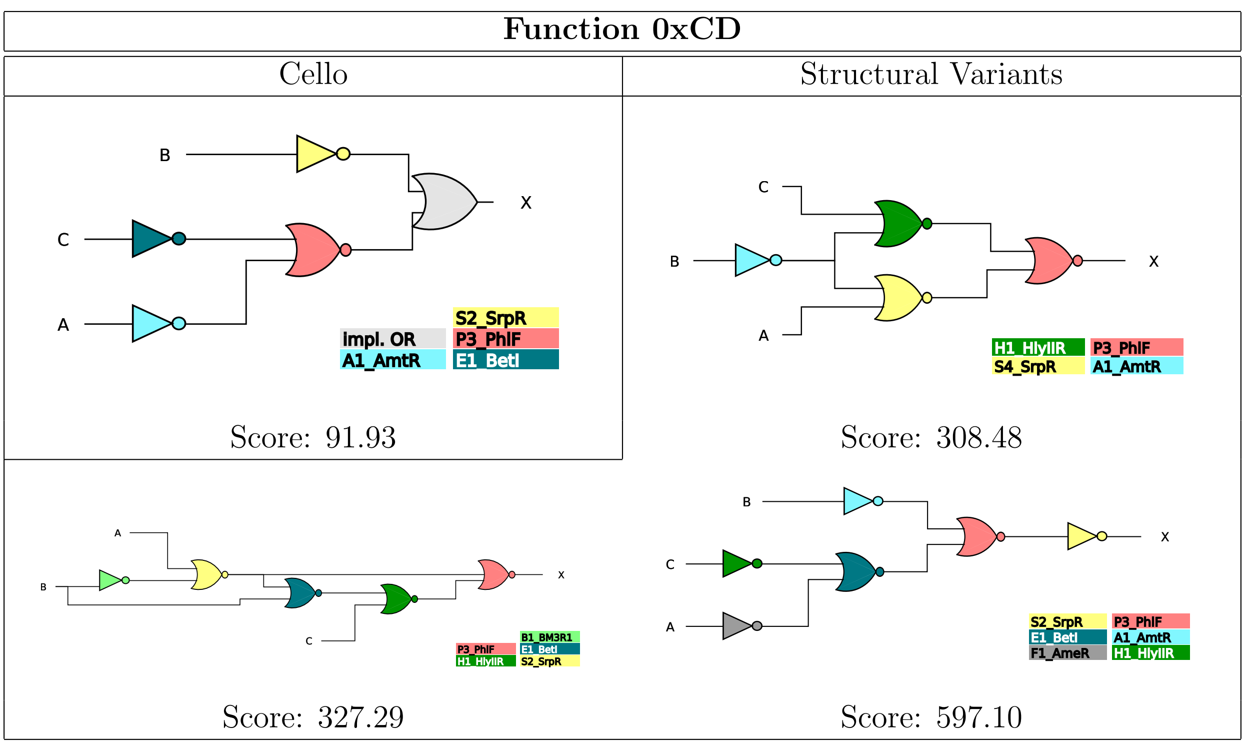


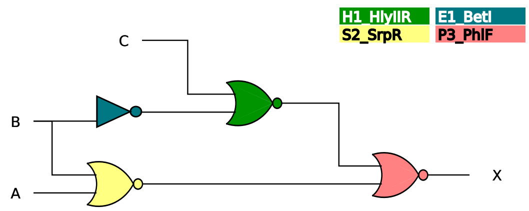
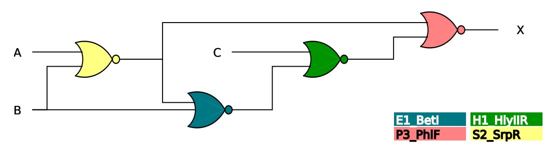
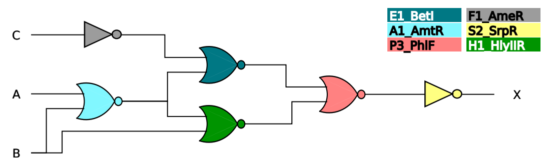
Score: 327.02

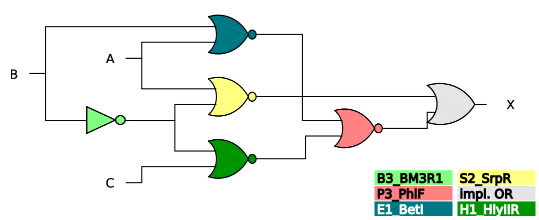
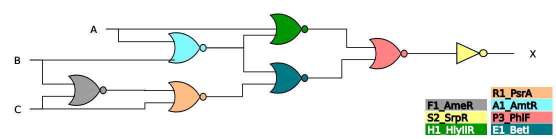


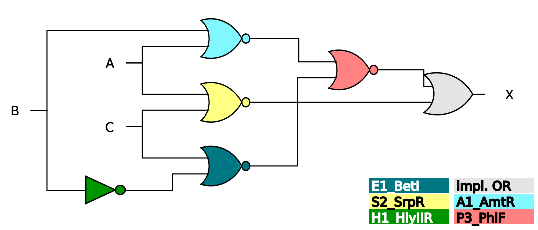
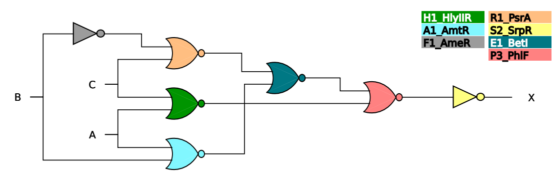
Score: 327.23


Function 0x4D
Cello
Structural Variants
C
A
B
X
P3_PhlF
A1_AmtR
B3_BM3R1
Impl. OR
E1_BetI
S2_SrpR
Score: 85.40
A1_AmtR
P3_PhlF
S3_SrpR
B3_BM3R1
H1_HlyIIR
Score: 326.25
S2_SrpR
H1_HlyIIR
P3_PhlF
B1_BM3R1
E1_BetI
R1_PsrA
Score: 327.39
H1_HlyIIR
P3_PhlF
B3_BM3R1
E1_BetI
A1_AmtR
S2_SrpR
F1_AmeR
Score: 595.79


Function 0xCD
Cello
Structural Variants
B
C
A
X
S2_SrpR
Impl. OR
P3_PhlF
A1_AmtR
E1_BetI
Score: 91.93
H1_HlyIIR
P3_PhlF
S4_SrpR
A1_AmtR
Score: 308.48
B1_BM3R1
P3_PhlF
E1_BetI
H1_HlyIIR
S2_SrpR
Score: 327.29
S2_SrpR
P3_PhlF
E1_BetI
A1_AmtR
F1_AmeR
H1_HlyIIR
Score: 597.10

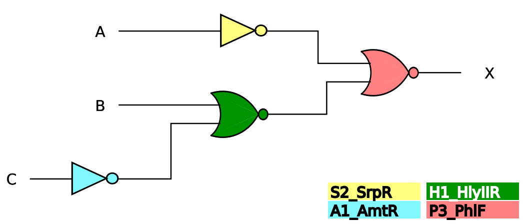
Function Multiplexer	
Cello	Structural Variants
 Score: 327.75	 Score: 327.79
 Score: 596.04	

Function 0x3D	
Cello	Structural Variants
 Score: 83.82	 Score: 595.88

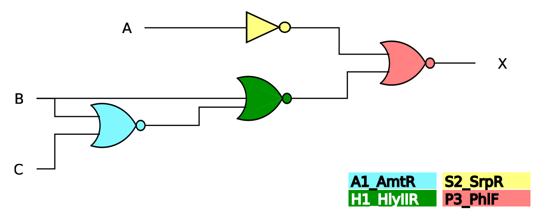
Function 0xBD	
Cello	Structural Variants
 Score: 88.30	 Score: 596.60

Function 0x0B

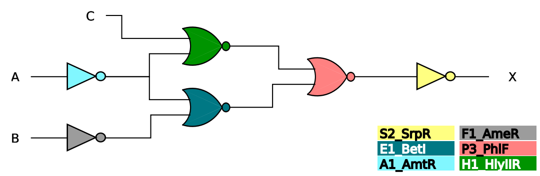
Cello	Structural Variants



Score: 328.09



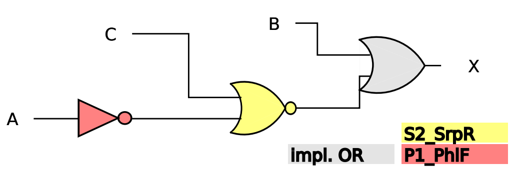
Score: 328.10



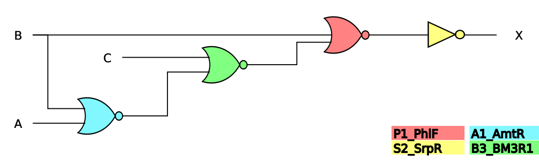
Score: 595.70

Function 0x3B

Cello	Structural Variant



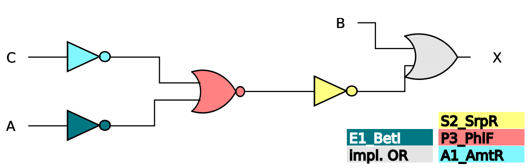
Score: 428.50



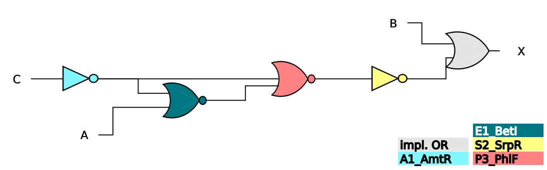
Score: 678.07

Function 0xFB

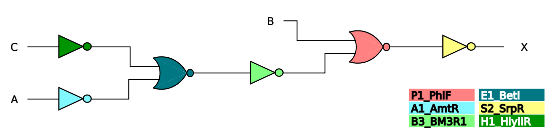
Cello	Structural Variants



Score: 418.52



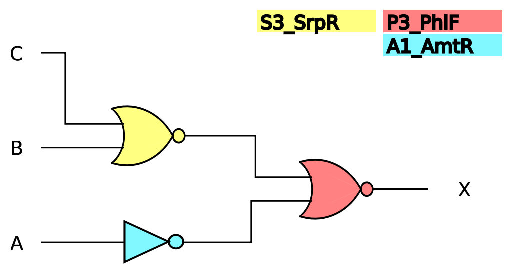
Score: 418.53



Score: 678.07

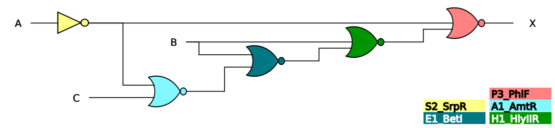
Function 0x07

Cello



Score: 327.93

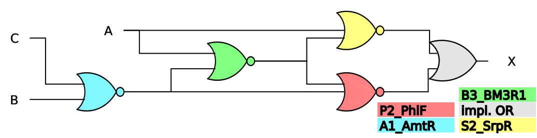
Structural Variant



Score: 328.11

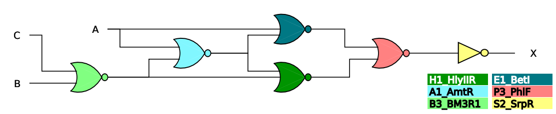
Function 0x87

Cello

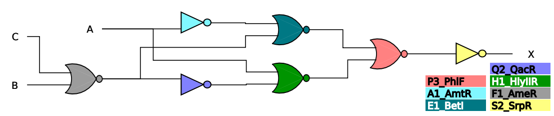


Score: 75.54

Structural Variants



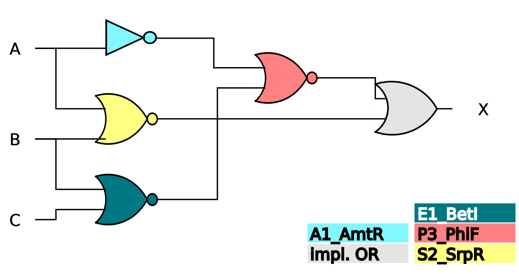
Score: 593.62



Score: 596.57

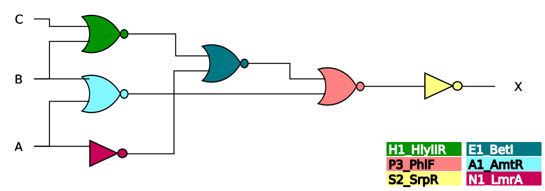
Function 0xC7

Cello



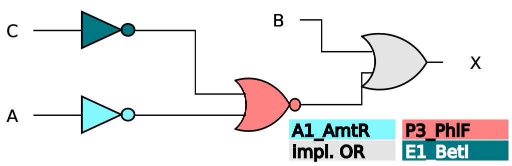
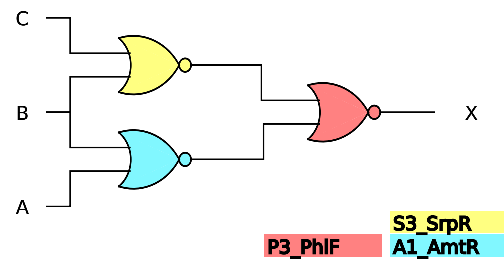
Score: 91.50

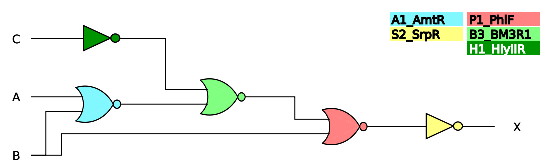
Structural Variant



Score: 597.08

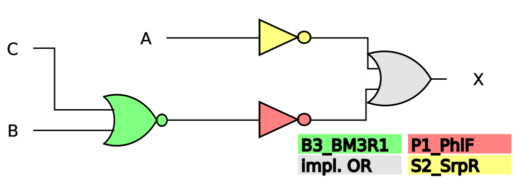
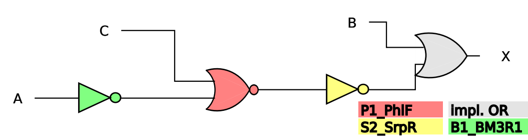
Function 0x37

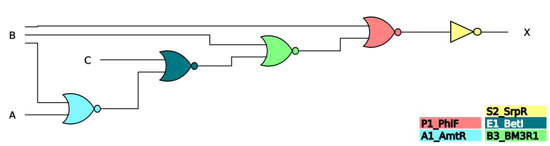
Cello	Structural Variants
 Score: 207.04	 Score: 327.93



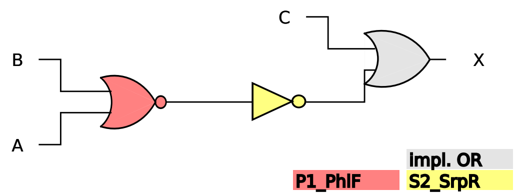
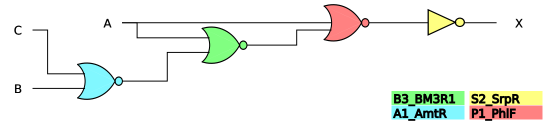
Score: 678.05

Function 0xF7

Cello	Structural Variants
 Score: 161.53	 Score: 473.93



Score: 678.07

Function 0x7F	
Cello	Structural Variant
	
Score: 183.04	Score: 677.91

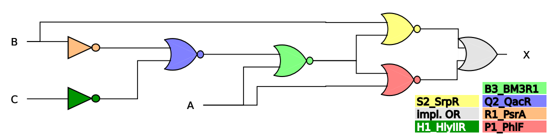
B.2 Classical Structure, Uncertainty-Aware Assignment Optimization

In the following, the three circuits mentioned in the main text 0x1c, 0x81 and 0x41 synthesized by Cello (so the non-modified original circuit structure) are depicted together with the optimal gate assignment found using the Cello score and the expectation-based score. The least separated on and off output histograms and their resulting final Cello and expectation-based scores are written below each.

Like explained in the main text, since the median gate outputs obtained via sampling are of improved accuracy with respect to the true medians compared to those calculated by Cello in the context of a circuit, the toxicity constraints for our assignment for function 0x41 are met using the E-score but not when using Cello's score. Thus, we obtain an assignment with high E-score, which would also exhibit a higher Cello score when calculating the toxicity constraints using the improved accuracy from sampling.

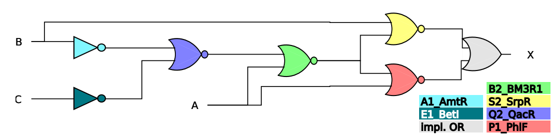
Function 0x1C

Assignment by Cello score



Cello-score: 146.34
E-score: 2.95

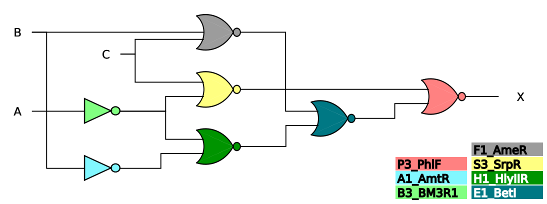
Assignment by expectation-based score



Cello-score: 129.98
E-score: 77.15

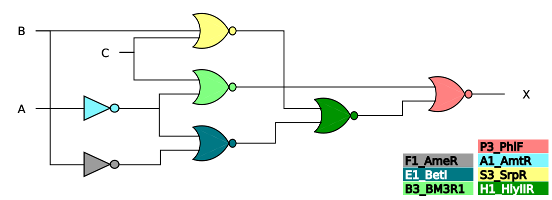
Function 0x81

Assignment by Cello score



Cello-score: 308.68
E-score: 52.48

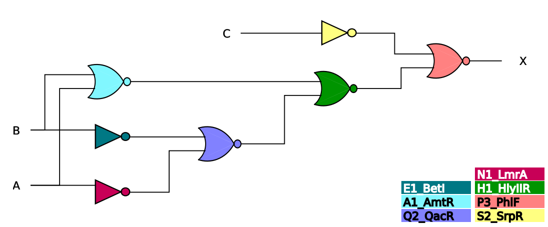
Assignment by expectation-based score



Cello-score: 143.68
E-score: 57.8

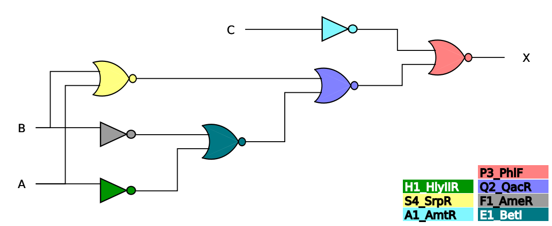
Function 0x41

Assignment by Cello score



Cello-score: 316.46
E-score: 35.96

Assignment by expectation-based score



Cello-score: 317.65 (see above)
E-score: 81.10