Vol.2022-CE-165 No.7
2022/6/5

ビジュアルプログラミングツールの特徴を取り入れたディジタル回路学習ツール

木村 元[1]　藤枝 直輝[1,a)]

概要：プログラミング教育の必修化に伴い，初学者が容易にプログラミングの基礎を学習するために，様々なビジュアルプログラミングツールが提案されている．我々は，こうしたツールの特徴を取り入れた，ディジタル設計の基礎を学ぶための学習ツールを提案する．AND，OR などのゲートの入力数をシームレスに変更できること，素子の内部に説明や他素子との接続情報を記載することで，ディジタル回路に関する知識が少なくても扱えるツールを開発した．描画負荷に関する評価の結果，簡単な回路であれば操作中のフレームレートは秒間 60 フレーム以上を維持でき，スムーズな操作が可能であることを確認した．ツールを利用した者へのアンケート調査の結果も併せて報告する．

A digital design learning tool that incorporates features of visual programming tools

Gen KIMURA[1]　Naoki FUJIEDA[1,a)]

Abstract: As programming education became compulsory in Japan, various visual programming tools have been proposed so that beginners can easily learn a basis of programming. In this report, we propose a learning tool to learn a basis of digital design, which incorporates features of the tools. The tool can change the number of gate inputs seamlessly and show explanation of logic elements and connection to other elements. These features enable users with little knowledge on digital circuits to deal with the tool. According to an evaluation of drawing load on editing a simple circuit, we confirmed that the frame rate more than 60 fps was maintained in the most cases. The result of a survey conducted to users is also reported.

1. はじめに

わが国では 2020 年度より小学校においてプログラミング教育が必修化され，意図する行動を実現するためにどのような動作の組み合わせを与えればよいのかを論理的に考える能力，すなわちプログラミング的思考の能力を育むことが求められるようになった [1]．またそれに伴い，初学者が容易にプログラミングの基礎を学べるよう，様々なビジュアルプログラミングツールが提案されている．そこでは，文字を使うのではなく，ブロックやノードといったオブジェクトを組み合わせることでプログラミングを行う．

一方で，近年の計算機システムでは，領域に特化したハードウェアとソフトウェアとの協調が求められるようになっている [2]．ソフトウェアのプログラミングは，しばしば動作の**直列的な**組み合わせと捉えられる．それに対して，ハードウェア，ディジタル回路の設計における重要な考え方は，構成要素には入力と出力があり，それらが全て**同時に**動作するということである．こうしたハードウェアの考え方の素養を早いうちから理解しておくことは，将来的にハードウェア・ソフトウェアの両方の知見を持った技術者を養うために重要であると考えられる．

既存のビジュアルプログラミングツールの中にも，ハードウェアの考え方の素養の理解に適したものがいくつかある [3], [4]．しかし，これらは物理的な教材の付属品として提供されているもので，利用へのハードルが高い．また，既存のディジタル回路の設計・シミュレーションのためのツール [5], [6] は，ゲートの入力数をあらかじめ指定しなけ

[1] 愛知工業大学
Aichi Institute of Technology, Toyota, Aichi 470–0392, Japan
[a)] nfujieda@aitech.ac.jp

ればならないこと，各ゲートの動作はあらかじめ知識として持っていないと扱うことが難しいなど，初学者への導入には課題がある．

そこで本研究では，ビジュアルプログラミングツールの要素を取り入れることにより，既存のディジタル回路設計ツールの課題点を解決した，初学者向けのディジタル回路学習ツールを提案する．ツールに求める要件として，マウスによる直感的な操作によって回路を構成できること，回路素子の動作についての事前知識を持っていなくても扱えること，構成した回路を Verilog HDL のハードウェア記述にエクスポートできること，の 3 つを挙げる．本稿では，これらの要件をどのように実現したかを中心に，ツールの設計と実装について述べる．また，性能評価としてツールの主要な操作にかかる処理時間の測定を，主観評価としてツール利用者へのアンケートの実施を，それぞれ行った．本稿ではこれらの結果についても報告する．

2. 関連研究

ビジュアルプログラミングツールは，画面上でブロックを組み合わせたり，ブロック同士を線で結んだりなどの方法でプログラミングを行えるツールである．遊び感覚でプログラミングについて学ぶことができ，初心者でもプログラミングを容易に行える特徴がある．既存のビジュアルプログラミングツールは，大きく 3 つに分けられる．1 つ目は，動作に対応するブロックを組み合わせてプログラミングを行う，ブロックベースのツールである．2 つ目は，オブジェクトや機能に対応したブロック（ノード）同士を線で結んでプログラミングを行う，ノードベースのツールである．3 つ目は，オブジェクトの変化する条件と結果を 1 つのルールとし，それらを組み合わせてプログラミングを行う，ルールベースのツールである．

ブロックベースのツールとして，Scratch [7] や MOONBlock [8], Blockly [9] が挙げられる．いずれのツールでも，「10 歩動かす」「90 度回す」といったオブジェクトの動作を表すブロックや，「マウスがクリックされたとき」「10 回繰り返し」などの制御を行うブロックなどが用意されている．これらのブロックをマウス操作で組み合わせることでプログラミングを行う．MOONBlock には，やや複雑な移動をブロックで指定できるなど，簡単なゲームを制作することに特化した機能が含まれている．また，作成したプログラムを JavaScript に変換することができ，その内容を確認できることも，MOONBlock の特徴である．Blockly には，あらかじめ迷路を通る，星形を書く，といったプログラムの内容が指定され，それに沿ってゲーム感覚でプログラミングを進める点に特徴がある．ブロックベースのツールは，動作を適切な順序で直列的に組み合わせるという，ソフトウェアのプログラミングに近い発想で設計されている．

ノードベースのツールに，MESH [3] や SAM Labs [4] がある．これらのツールでは，入力系（ボタンやセンサなど）や出力系（LED やスピーカなど）のブロックが用意されており，これらを接続して入力と出力を関連付けることでプログラミングを行う．また，条件分岐や条件同士の演算，カウンタやタイマなどといった中間のブロックも用意されており，こうしたブロックを用いることで，センサの入力値に応じて異なる色の LED を点灯させるといった機能も実現できる．ノードベースのツールの発想は，よりハードウェア設計の同時並列的な考え方に近い．しかし，MESH や SAM Labs はいずれも物理的な教材の付属品としてプログラミングツールが提供されている．具体的には，コンピュータと Bluetooth で接続できる入力系・出力系のさまざまなパーツが教材として販売されており，ツールはこれらのパーツの挙動をプログラミングするためのものと位置づけられている．そのため，導入のためのハードルが高い．

ルールベースのツールには Viscuit [10] がある．Viscuit のプログラミングの単位は，変化の対象とするオブジェクトを左側に，変化後のオブジェクトを右側に記載した，メガネとよばれるブロックである．メガネによって記述されたオブジェクトの変化のルールを組み合わせることにより，プログラミングを行う．Viscuit ではオブジェクトの動作もグラフィカルに表現できるため，未就学児に対しても導入が可能 [11] なほど簡単に利用できる．また，ルール同士の順序は問わないため，ハードウェア設計の同時並列的な考え方とも親和性が高い．ただし，Viscuit はオブジェクトの変化のルールしか記述ができないため，演算を伴うようなプログラミングには不向きである．

こうしたプログラミングツールの中には，作成したプログラムを共有・公開する機能をもつものもある．これを使い，児童が作成したプログラムを収集し，プログラミング能力を児童がどのように獲得しているかを分析する研究もみられる [11], [12].

3. 提案ツール

3.1 既存のツールの課題点と解決案

既存のディジタル回路の設計・シミュレーションのためのツールとして，Deeds-DcS [5] や SimcirJS [6] が挙げられる．Deeds-DcS では，入出力やゲートのオブジェクトを選択し，画面上をクリックしてそれらを配置し，配線モードでオブジェクト同士を配線することで，回路図を作成できる．作成した回路は，シミュレーションモードで回路図画面上の入力を切り替え，それに対する出力を画面上で確認することで検証できる．タイミングチャートを用いたシミュレーションも可能である．また，回路を FPGA（Field Programmable Gate Array）の実習ボード上に実装しての動作確認もできるよう，VHDL によるハードウェア記述

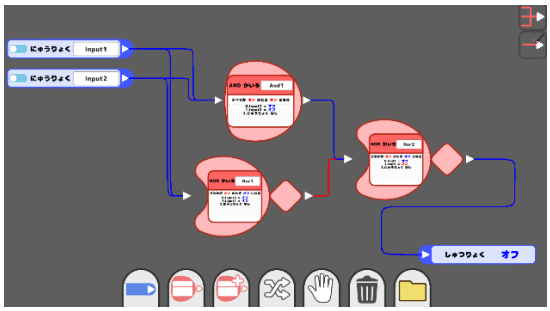


図 1 提案ツールの画面の例.
Fig. 1 Example of appearance of the proposed tool.

へのエクスポートなどの機能ももつ．SimcirJS は Web ブラウザ上で動作するディジタル回路シミュレータであり，Deeds-DcS の回路図作成モード，シミュレーションモードに相当する機能が用意されている．

これらのツールを初学者が使うにあたっては，大きく 3 つの課題点がある．1 つ目は，ユーザインタフェースが回路図を意識しており，配線の入力に煩雑な操作が必要な点である．特に Deeds-DcS では，配線は 1 度に水平方向または垂直方向に 1 本しか追加できず，1 つの接続あたり数回のマウスクリックが必要になる．2 つ目は，論理ゲートを使用する際に，その入力数をあらかじめ指定しなければならない点である．例えば，回路設計を行う中で，2 入力の AND ゲートを用いればよいと考えていた箇所に，実は 3 入力の AND ゲートを用いる必要があった，といったケースはしばしば存在する．この場合，既存のツールでは，一度 2 入力の AND ゲートを削除してから，3 入力の AND ゲートを配置し，それに合わせて配線をやり直す必要がある．3 つ目は，各回路素子の動作はあらかじめ知識として持っている必要がある点である．素子の動作を十分に理解していないと，各素子の出力，ひいては回路全体の動作を予測することが難しい．

以上の課題点を解決するため，我々は，ビジュアルプログラミングツールの要素を取り入れたディジタル回路学習ツールを開発した．インタラクティブなユーザインタフェースを実現するため，開発には C#および Unity を使用する．オブジェクト志向の利点を活かし，ゲートの入力を要素数が可変のリストで管理することにより，ゲートの入力数をシームレスに切り替えることができる．また，各回路素子の動作に関する説明や，他の素子との接続情報を画面上に記載することで，素子に関する知識が少ない場合でも利用可能なツールとする．

3.2 ツールの基本操作

図 1 に，提案ツールの画面の例を示す．提案ツールの基本操作は画面下部のボタンで行う．基本操作ボタンは左から順に，「入出力ノードの追加」「基本ノードの追加」「応用ノードの追加」「ノード接続」「カメラ移動」「ノード削除」「セーブ・ロード」のボタンになっている．これらを使い，ノードを画面上部のキャンバス上に配置したり，キャンバス上のノード同士を接続することで，回路を組み上げる．

入出力ノード，基本ノード，応用ノードを選択した場合には，ボタンの下にノードを選択するためのツールボックスが表示される．ツールボックスから追加したいノードをクリックしたら，キャンバス上の配置したい場所で再度クリックすることで，ノードが追加される．なお，基本ノードはゲートなどの組合せ回路の素子，応用ノードはフリップフロップなどの順序回路の素子を含む．

ノード接続のモードでは，各ノードの出力端子を別のノードの入力端子へとドラッグアンドドロップすることで，ノード同士を接続する．出力端子は，接続先である入力端子付近に移動すると自動で接着するスナップ機能を持っており，スナップされた状態でマウスをドロップするとノード同士の接続が完了する．スナップされていない状態でマウスを離した場合，接続はキャンセルされる．このモードでは，接続状態になっている端子の接続を変更したり，接続を解除することもできる．また，ノード自身もドラッグアンドドロップで移動可能である．

カメラ移動モードでは，左クリックをした状態で，マウス操作で画面の表示範囲を変更できる．マウスホイールによって，ズームイン・ズームアウトも可能である．また，削除モードでは，ノードをクリックするとそのノードを削除し，接続線をクリックするとその接続を解除する．

セーブ・ロードモードでは，ファイル名入力と保存済みファイルリストからなる追加のインタフェースが表示される．ファイル名を入力してセーブボタンを押すと，回路の情報がファイルに保存される．また，保存済みファイルリストからセーブボタンを押すと上書き保存が行われ，ロードボタンを押すと現在の回路を破棄し，保存した回路の情報を読み込んで表示する．

3.3 ツールの機能とその内部動作

提案ツールの使いやすさのためには，マウスによる直感的な操作で回路が構成できること，素子の動作を事前に知っていなくても扱えること，回路をハードウェア記述にエクスポートできること，の 3 つが必要である．本節では，これらを実現するためのツールの内部動作について説明する．

3.3.1 ノードオブジェクトとその接続

図 2 に，本ツールにおけるノードや接続線に対応するオブジェクトの構成を示す．提案ツールのノードオブジェクトは，ノードの動作を定義するノードメイン部分と，入力端子を管理するオブジェクト（入力端子グループ）と，出力端子を管理するオブジェクト（出力端子グループ）から構成される．出力端子には，始点用と終点用の 2 種類が存在する．始点用出力端子は，1 つの出力端子グループに 1

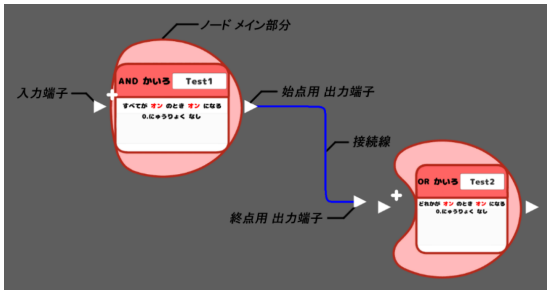


図 2 ノードと接続線に関連するオブジェクトの構成.
Fig. 2 Organization of objects related to nodes and connections.

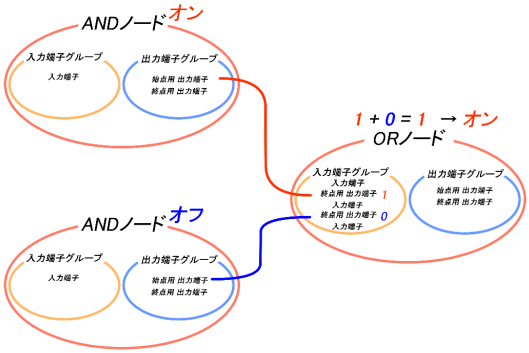


図 3 ノードオブジェクトの内部接続.
Fig. 3 Internal connections of node objects.

つだけ存在し，端子のみの移動はできない．終点用出力端子は，ノードオブジェクトの接続数+1 個存在し，端子をドラッグアンドドロップで移動可能である．始点用出力端子と終点用出力端子は接続線で結ばれる．接続モードでは，終点用出力端子がマウスのドラッグに追従して移動する．これにより，マウス操作に追従して配線を伸ばす挙動を実現している．

図 3 に，ノードオブジェクトの内部接続を示す．接続モードで終点用出力端子が別のノードの入力端子へと接続されると，それらの端子がペアリングされる．また，AND ゲートのように入力数が可変である素子の場合，入力端子グループは常に未接続の入力端子を 1 つだけもつように，入力端子の生成・削除を行う．これにより，入力数のシームレスな変更を可能としている．ノードメイン部分では，入力端子グループ内の各入力端子に対し，ペアリングされた終点用出力端子の親ノードを調べ，その出力を読み取る．その後，取得した値を用いてノードに対応する演算（例えば AND ゲートであれば論理積）を行い，ノードの出力を変更する．これにより，論理素子の動作を実現する．

3.3.2 ノードの情報表示

図 1，図 2 からも読み取れる通り，提案ツールの各ノードには，ノードの動作の説明や各入力の状態を示すテキストが記載されている．ノードの動作の説明はノードの種類に対応づけられているので，それを表示する．入力の状態

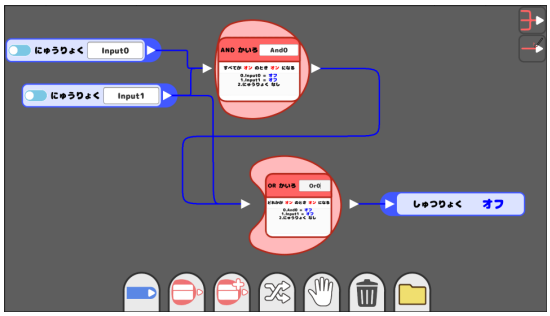


図 4 コード変換の例に利用する回路.
Fig. 4 Example circuit for code generation.

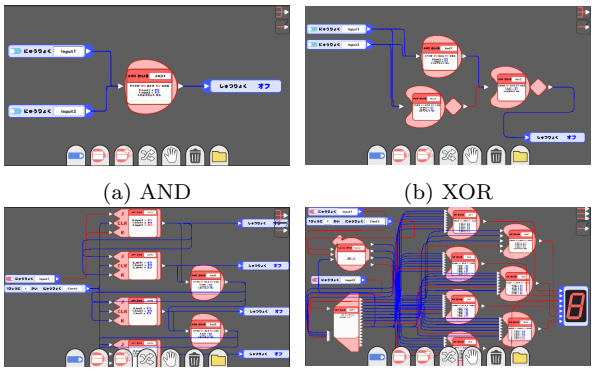
```
‘timescale 1ns / 1ps
module test(Input_Toggle0,Input_Toggle1,Output0);
input Input_Toggle0,Input_Toggle1;
output Output0;
wire AndNode0,OrNode0;

assign AndNode0 = Input_Toggle0 & Input_Toggle1;
assign OrNode0 = AndNode0 | Input_Toggle1;
endmodule
```

図 5 生成された Verilog HDL 記述.
Fig. 5 Generated Verilog description.

を示すテキストを生成する際は，まずノードの出力を変更するときと同じ要領で，接続されたノードとその出力を求める．その後，それらの情報をテキスト生成用のクラスに渡して，所望のテキストを得る．これにより，ノードの動作の説明を確認できるとともに，ノード内での演算の様子をリアルタイムに確認できる．

3.3.3 コード変換機能

提案ツールには，作成した回路から Verilog HDL のハードウェア記述へのエクスポート機能も用意されている．この機能の内部動作について説明する．まず，画面上に配置されているノードを，入力ノード，出力ノード，基本ノード，応用ノードの順に，それぞれリストで取得する．次に，入力ノードに対しては `input` 宣言，出力ノードに対しては `output` 宣言とそれに接続する `assign` 文，基本ノードに対してはその出力に対する `wire` 宣言と動作に対する `assign` 文，応用ノードに対してはその出力に対する `reg` 宣言と動作に対する `always` 節を，順に生成する．これらを全て連結してテキストファイルとして出力することで，Verilog HDL によるハードウェア記述が完成する．**図 4** に示す回路をもとに，提案ツールが生成した Verilog HDL 記述を，**図 5** に示す．ここでは，AND ゲートと OR ゲートの出力に，それぞれ `AndNode0`，`OrNode0` と名前がつけられている．なお，現在のエクスポート機能には，出力ノードに対する `assign` 文が正常に生成されない問題がある．その解決は今後の課題である．



(a) AND (b) XOR

(c) CNT (d) 7SEG

図 6 性能評価に用いた回路.

Fig. 6 Circuits used in performance evaluation.

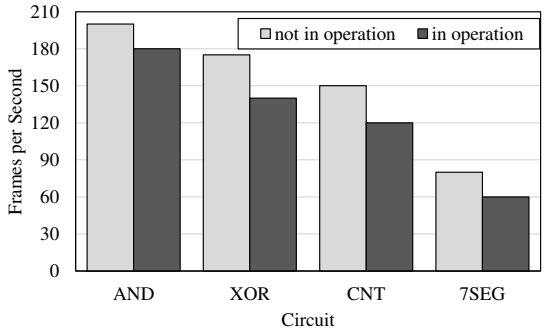


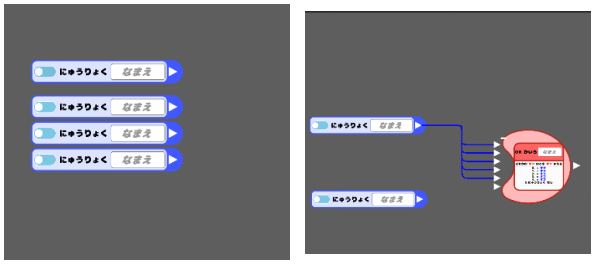
図 7 性能評価の結果.

Fig. 7 Results of performance evaluation.

4. 性能評価

本章では，提案ツールがスムーズに動作するかどうかを確認するために実施した，性能評価実験について述べる．性能評価実験は，いくつかの回路を画面上に作成したときのアプリケーションのフレームレートを測定することにより行った．フレームレートの測定は，Unity で用意されているフレームレート表示機能を用いた．性能評価に用いた回路を**図 6** に示す．用いた回路は，(a) AND ゲートのみを配置した回路 AND，(b) XOR ゲートを AND ゲートと NOR ゲートにより実装した回路 XOR，(c) JK フリップフロップを用いた 4 ビットのカウンタ CNT，(d) 1 桁の 7 セグメント LED のデコーダ回路 7SEG の 4 種類である．それぞれに対して，マウス操作を行わないときを非操作時，最も接続関係の多いノードをドラッグアンドドロップで移動しているときを操作時として，フレームレートを測定した．評価は Core i5 7200U CPU と 8 GB のメモリを搭載した Windows 10 のラップトップ上で行った．

図 7 に性能評価の結果を示す．横軸は回路名，縦軸はフレームレート（fps; Frames per Seconds）である．また，非操作時のフレームレートを薄いグレー，操作時のフレームレートを濃いグレーで示す．7SEG 回路の操作時を除いて，スムーズな動作の目安とされる 60 fps を大きく上回るフレームレートを得ており，簡単な回路であればスムーズな操作が可能であることが確認できた．



(e) Input nodes (f) Connections

図 8 詳細な性能評価に用いた（不完全な）回路.

Fig. 8 (Incomplete) circuits used in detailed performance evaluation.

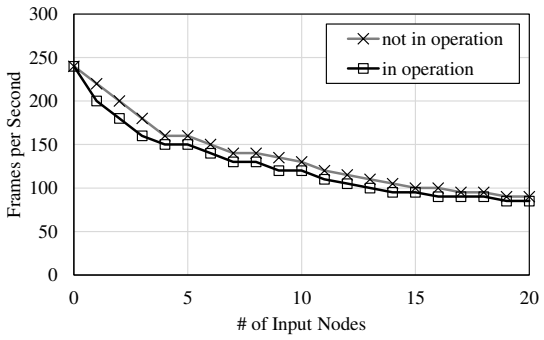


図 9 (e) の回路における性能評価の結果.

Fig. 9 Results of performance evaluation with circuit (e).

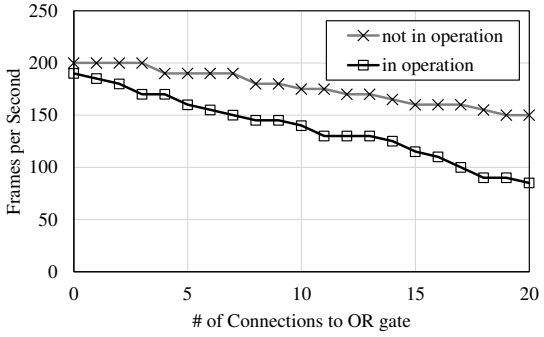


図 10 (f) の回路における性能評価の結果.

Fig. 10 Results of performance evaluation with circuit (f).

図 7 で確認された操作時と非操作時のフレームレートの違いについて詳しく分析するため，追加の性能評価を行った．追加の性能評価に用いた回路を**図 8** に示す．これらは純粋に性能評価のために作成したもので，出力ノードが存在しておらず回路として意味のあるものではない．(e) では，画面上に入力ノードのみを 0~20 個配置する．(f) では，画面上に入力ノード 2 個と OR ゲートを配置し，入力ノードから OR ゲートへ 0~20 本の接続を行う．それ以外の測定条件は同一である．

(e)，(f) の回路を用いてフレームレートを測定した結果を，それぞれ**図 9**，**図 10** に示す．縦軸はフレームレート

表 1 アンケートの質問内容.
Table 1 Enquete questions.

項目	経験のある人への質問	経験のない人への質問
(1) 満足度	このツールに対する満足度を教えて下さい.	このツールに対する満足度を教えて下さい.
(2) 設計の容易さ	既存のツールより回路設計が容易でしたか？	回路設計が容易でしたか？
(3) 検証の容易さ	回路の動作確認は容易でしたか？	回路の動作確認は容易でしたか？
(4) 興味	簡単な回路を設計する場合，既存のツールではなくこのツールを使用したいと思いますか？	論理回路に対して興味を持ちましたか？

(fps)，横軸は (e) では入力ノードの個数，(f) では接続の本数である．いずれのグラフでも，非操作時と操作時のフレームレートをそれぞれグレーと黒の線で示している．図 9 では，ノード数の増加に対してフレームレートが比較的大きく減少していることが確認できる．しかし，操作時と非操作時とでフレームレートはほとんど変化しない．図 10 では，非操作時のフレームレートの減少は比較的少ない一方，操作時にはフレームレートが大きく減少する．これらの結果から，非操作時のフレームレートはノード数に，操作時のフレームレートは接続線の本数に，より強く依存していることが確認できた．

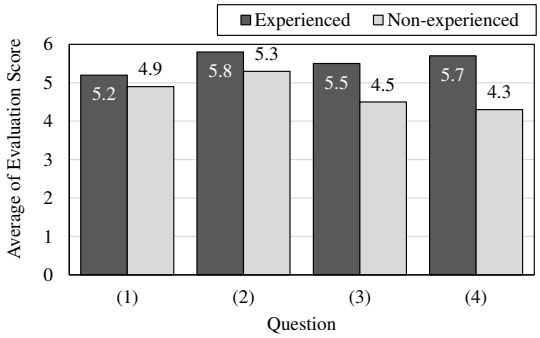


図 11 アンケートの回答結果.
Fig. 11 Evaluation Scores to enquete questions.

5. 主観評価

本章では主観評価として，提案ツールを実際にユーザに利用してもらい，ツールの使いやすさなどをアンケートにより評価した結果について述べる．主観評価においては，まず各ユーザに提案ツールの使用方法を記したドキュメントに目を通してもらった．次に，実際に提案ツールを操作し，簡単な回路の作成と動作確認のタスクを実施してもらった．その後，ツールに対する満足度などに関するアンケートに回答してもらった．ユーザは，ディジタル回路設計経験のある人として本学の電気学科電子情報工学専攻の学生 10 名，経験のない人として他大学の電気・情報系でない学生や社会人 10 名の，計 20 名とした．

表 1 に，アンケートの質問内容を示す．設問は全 4 問であり，それぞれの設問について 6 段階で評価してもらった．回路設計経験のある人とない人とで設問の文言は異なるものの，それぞれ，ツールに対する満足度，ツールを使用したときの設計の容易さ，ツールを使用したときの検証の容易さ，ツールや回路設計に対する興味，を問うものとなっている．またこれらの設問とは別に，自由記述でツールを使用した感想や要望についての回答も求めた．

図 11 に，各アンケート設問に対する評価の平均を示す．横軸は設問番号，縦軸はその設問に対する評価の平均であり，回路設計経験のある人は濃いグレーで，ない人は薄いグレーでそれぞれ示している．

回路設計経験のある人に対する評価結果はいずれも平均で 5 点を超え，おおむね好意的な反応が得られた．自由記述からは，ノードの入力数に制限がない点や，ノードの情報表示が可能である点，接続の変更が行いやすい点を評価する感想が得られた．全体的な満足度の設問では他の設問よりも低い評価値を示しており，ユーザインタフェースに対する改善の提案もいくつか見られた．

回路設計経験のない人に対する評価結果では，ツールに対する満足度や，回路設計の容易さの点では比較的高い評価が得られた一方，動作確認や興味に対する評価は低かった．自由記述の感想でも，ツールの使い方は容易に理解できたものの，どのような回路を組めばよいのかがわからない，問題を出題してほしいなどといったコメントが多く見られた．すなわち，現時点では提案ツール単体でディジタル回路の学習ができる状態にはないことがうかがえる．

以上の結果より，操作の容易さには高い評価が得られた一方，ユーザインタフェースの改善を図っていくことや，問題を解く形式にすることでパズル感覚で学習できるようにすることが今後の課題として抽出された．

6. おわりに

既存のディジタル回路設計ツールは，初学者にとっていくつかの扱いにくい点があった．本稿では，それらを解消した，ビジュアルプログラミングツールの要素を取り入れたディジタル回路学習ツールについて報告した．現在は提案ツールの公開に向けて，プログラムの整理や不具合の修正などの作業を進めている．その後，ユーザインタフェースの改善や問題の出題機能など，主観評価によって抽出できた課題点の改善を進めていく予定である．また，将来的

には提案ツールを授業等で利用することを通じて，提案ツールが実際にハードウェアの考え方の素養の早期理解につながるかどうかも検証していきたい．

謝辞 本研究の一部は JSPS 科研費 21K12164 の支援によるものである．

参考文献

[1] 文部科学省：小学校プログラミング教育の手引（第三版）(2020).
[2] Hennessy, J. L. and Patterson, D. A.: コンピュータアーキテクチャ定量的アプローチ 第 6 版，エス・アイ・ビー・アクセス (2019).
[3] ソニーマーケティング：MESH ｜誰でも簡単にプログラミングができる｜つくって楽しい｜学んで楽しい，（オンライン），入手先 ⟨https://meshprj.com/jp/⟩（参照 2022-03-18）.
[4] SAM Labs: Home – SAM Labs, (online), available from ⟨https://samlabs.com/us/⟩ (accessed 2022-03-18).
[5] Donzellini, G.: Digital Electronics Deeds, (online), available from ⟨https://www.digitalelectronicsdeeds.com/⟩ (accessed 2022-03-18).
[6] Arase, K.: SimcirJS, (online), available from ⟨https://github.com/kazuhikoarase/simcirjs⟩ (accessed 2022-03-18).
[7] Scratch 財団：Scratch – Imagine, Program, Share, (online), available from ⟨https://scratch.mit.edu/⟩ (accessed 2022-03-18).
[8] UEI Corp.: MOONBlock, (online), available from ⟨https://moonblock.jp/⟩ (accessed 2022-03-18).
[9] Google LLC: ブロックリー・ゲーム，（オンライン），入手先 ⟨https://blockly.games/?lang=ja⟩（参照 2022-03-18）.
[10] デジタルポケット：ビスケット Viscuit — コンピュータは粘土だ!!,（オンライン），入手先 ⟨https://www.viscuit.com/⟩（参照 2022-03-18）.
[11] 渡辺勇士，中山佑梨子，原田康徳，久野 靖：幼稚園児のビスケットプログラムにおける繰返し続けるプログラムの理解の分析，情報処理学会論文誌教育とコンピュータ（TCE），Vol. 7, No. 1, pp. 38–49 (2021).
[12] 太田 剛，加藤 浩，森本容介：子供のプログラミング能力の獲得段階に関する定量的分析：小学校 4〜6 年生の Scratch プログラミングを対象として，情報処理学会論文誌教育とコンピュータ（TCE），Vol. 5, No. 3, pp. 35–43 (2019).