

# 実行順序の制御と表示

1セクション毎に内容を確認しながら進めてください。 `c t r l + Enter`です。

## 標準モデルの実行

レートの設定

タイミング凡例

強調

なし

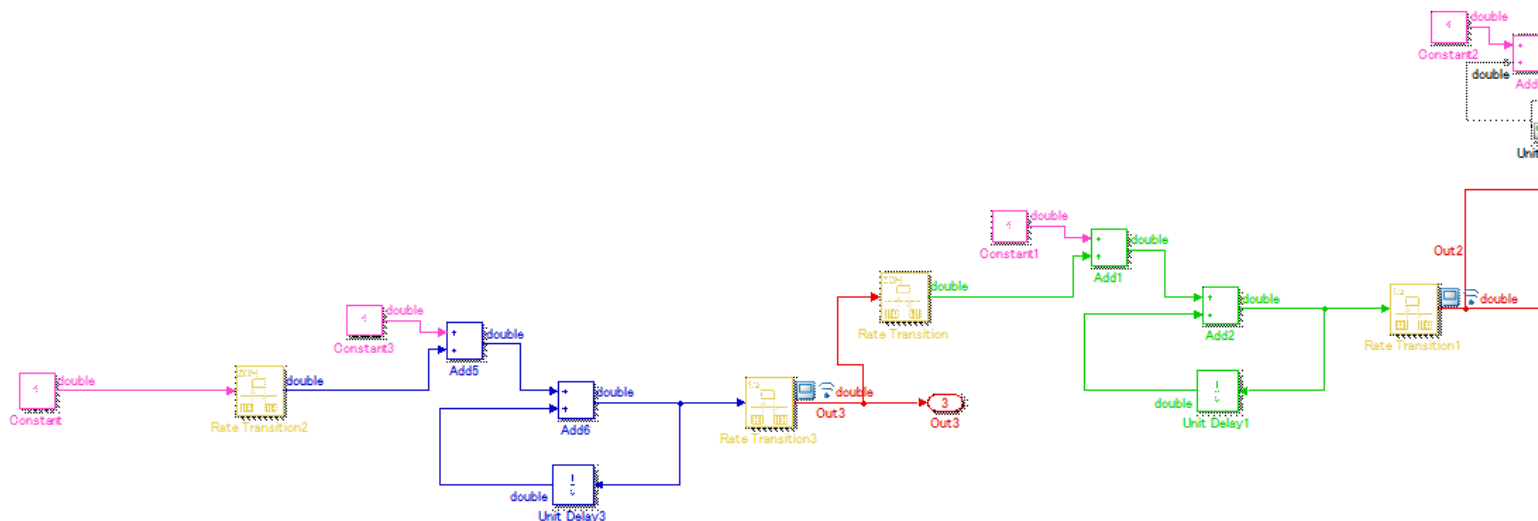
▼

▼ 離散		
		0.01
		0.1
		0.2
▼ その他		
		定数
		マルチレート

モデルの左側に遅いレート（0.02）、右側が最速レート（0.01）

それを直列に結合した特殊なモデルを作成した。

```
open_system("timebasedmodel0.slx");
set_param(bdroot,'SampleTimeColors','on')
sim("timebasedmodel0.slx");
save('matlog_model0.mat','logout_model0');
q_ans=questdlg({'モデルの表示を確認して先に進んでください。','先に進みますか'});
if ~strcmp(q_ans,'Yes')
    return;
end
```

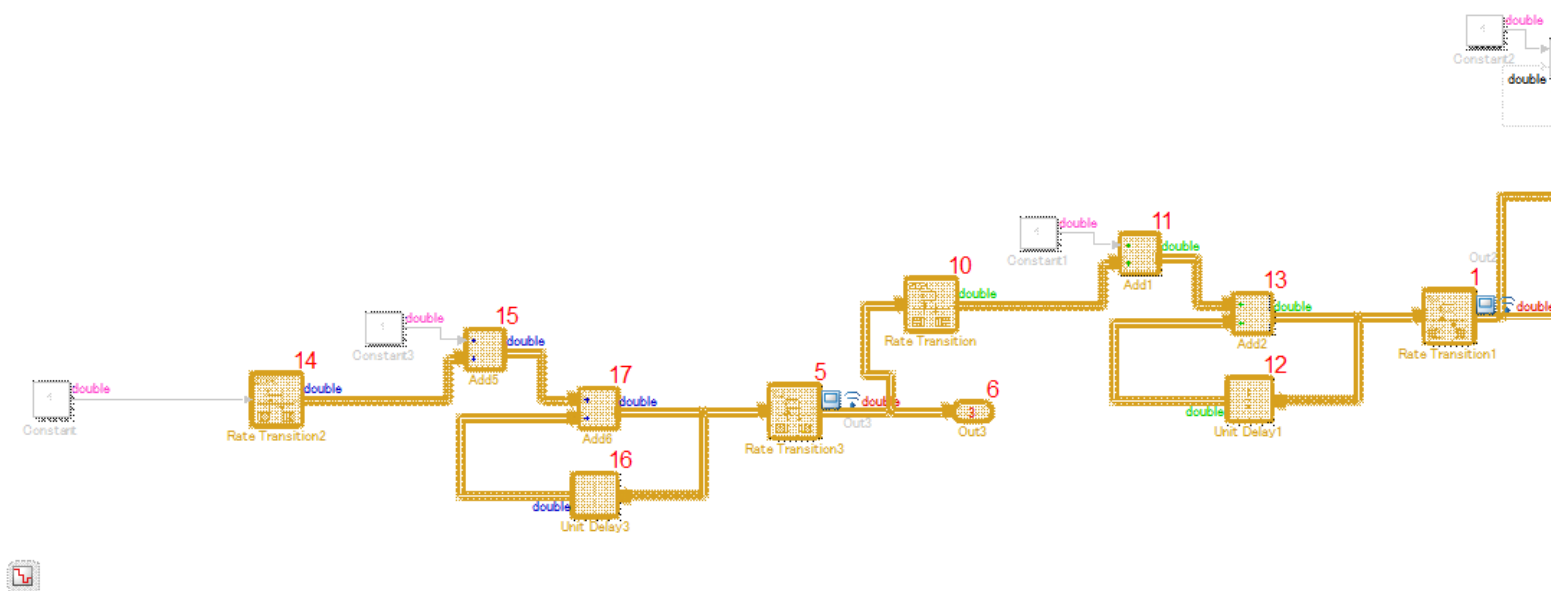


実行順序の表示

【デバッグ】のタブから、実行順序を選択する

timebasedmodel0\* - Simulink

シミュレーション	デバッグ	モデル化	書式設定	アプリ
<p>パフォーマンス アドバイザー</p> <p>パフォーマンス</p>	<p>診断</p> <p>情報の オーバーレイ</p>	<p>信号のトレース</p> <p>コメントアウト</p> <p>出力値</p>	<p>一時停止時間 (秒)</p> <p>ブレークポイントの追加</p> <p>ブレークポイントリスト</p>	
<p>検索</p>				
<p>★ お気に入り</p> <p>★ 凡例</p>				
<p>ブロック</p> <p>05 実行順序</p> <p>V:1 バリエーション</p> <p>V:0 バリエーション</p> <p>V:1 バリエーション</p>				
<p>関数コネクタ</p> <p>実行順序 - シミュレーション中にブロックが呼び出される順序</p> <p>ツールヒント内の説明</p>				



見ての通り、標準では早いレートが優先されて、遅いレートが後で計算されるので、この場合はモデルの流れと

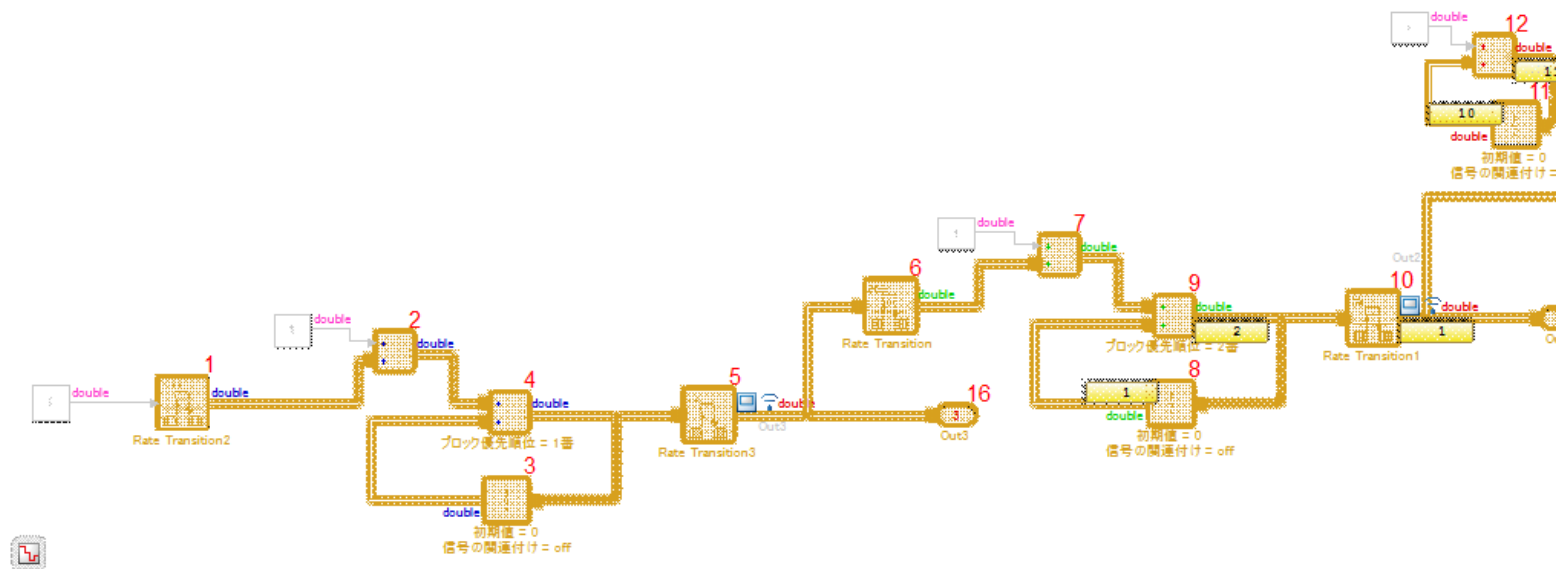
実行順序は逆になる。

```
% 上記の絵が表示されるであろうコマンドを下に記載するが、
% 残念ながら人の手によるウインドウ操作じゃないと表示されません
set_param(bdroot, 'SampleTimeColors', 'off');
set_param(bdroot, 'SortedOrder', 'on')
set_param(bdroot, 'DisplaySortedLists', 'on')
% sim("timebasedmodel0.slx");
```

## 優先順位の変更

**add**ブロックの優先順位を変更し、モデルの構成に合わせた実行順序に変更した。

**Rate Transition** にも設定してある。



```
open_system("timebasedmodel1.slx");
set_param(bdroot,'SampleTimeColors','on')
sim(bdroot);
save('matlog_model1.mat','logout_modle1');
q_ans=questdlg({'モデルの表示を確認して先に進んでください。','先に進みますか'});
if ~strcmp(q_ans,'Yes')
    return;
end
```

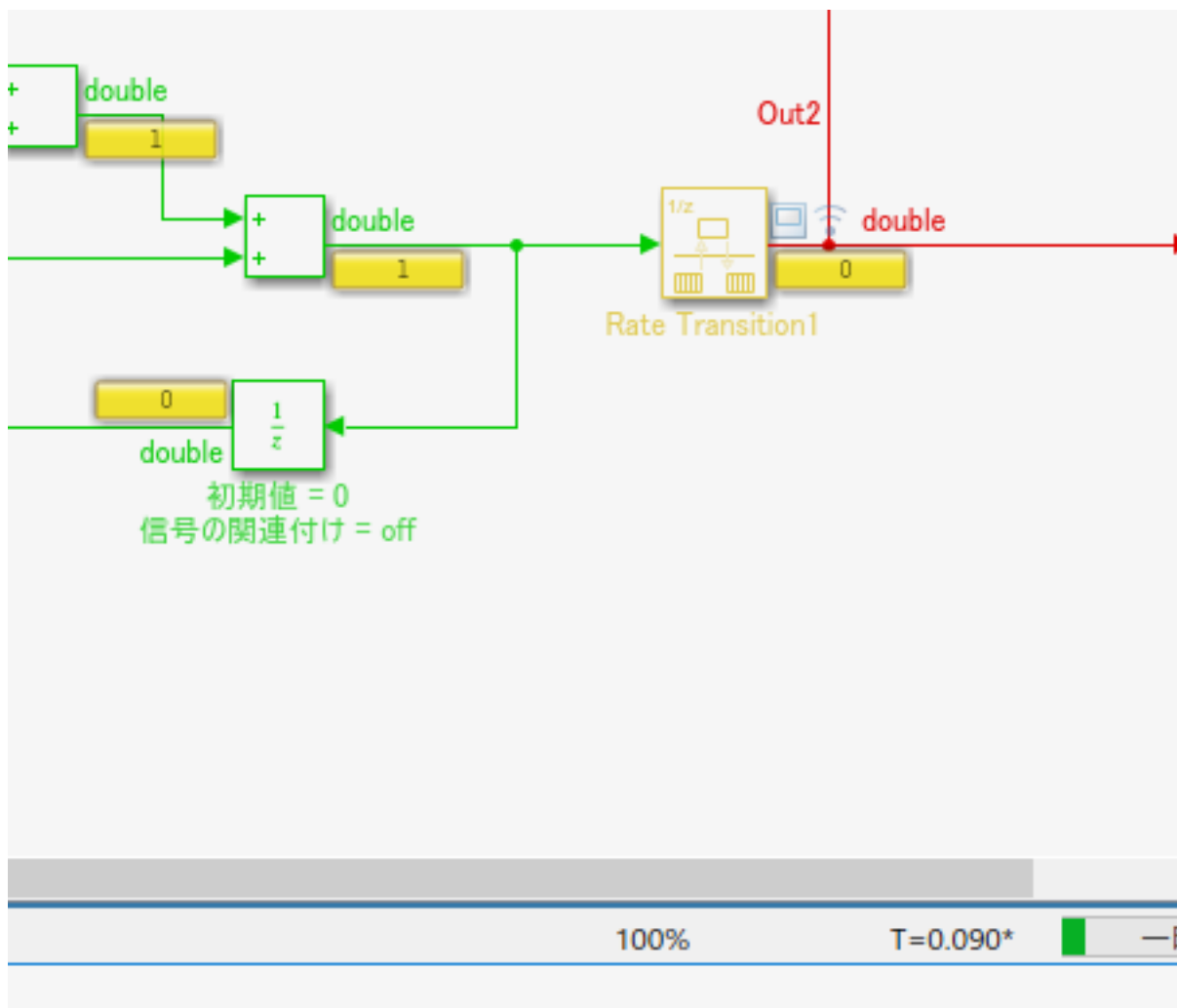
## 結果

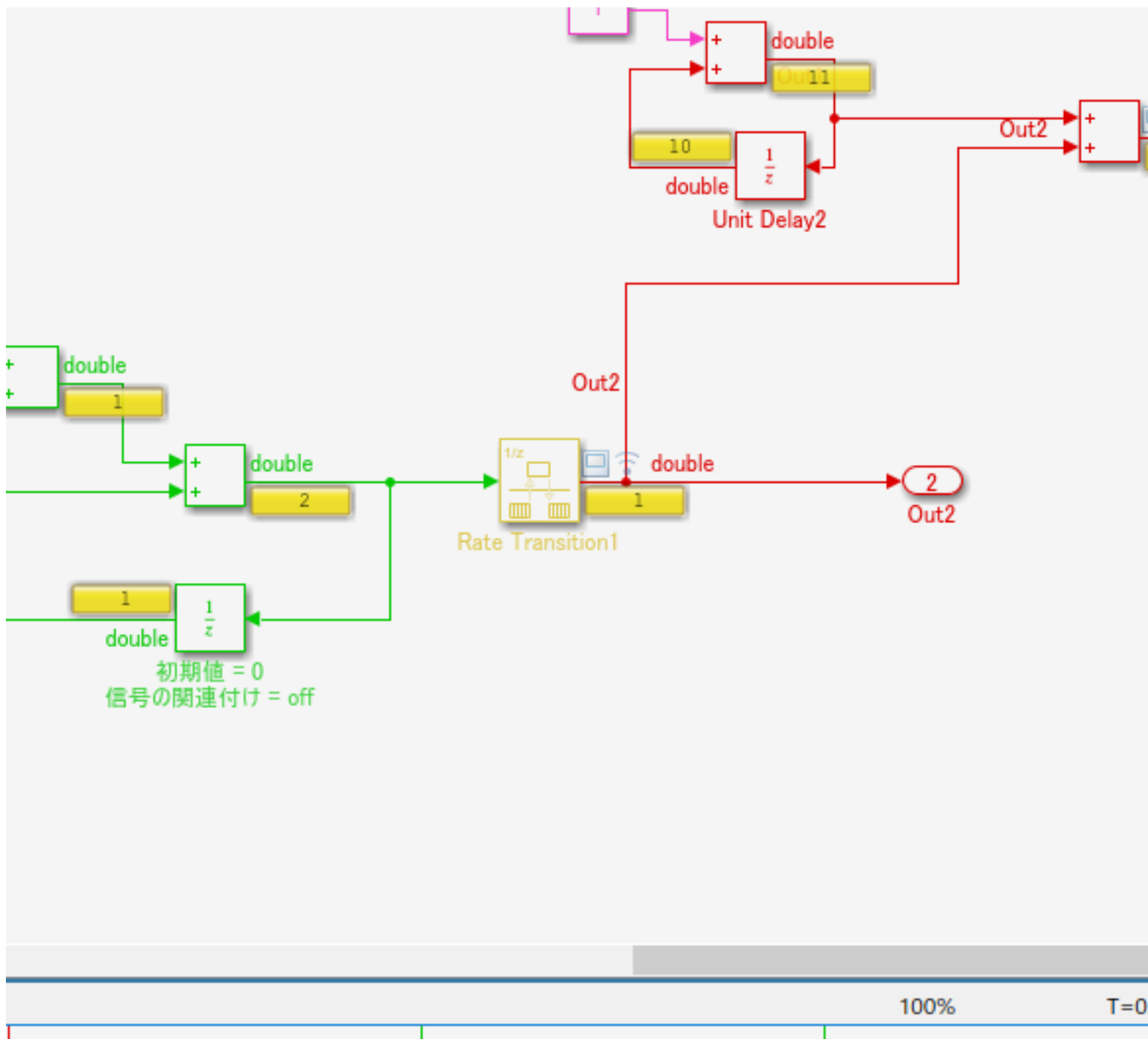
out3の入力値が1周期前後するため、結果に違いが生じる。

```
%% ビューワー表示 自分で設定して使って
Simulink.sdi.view
```

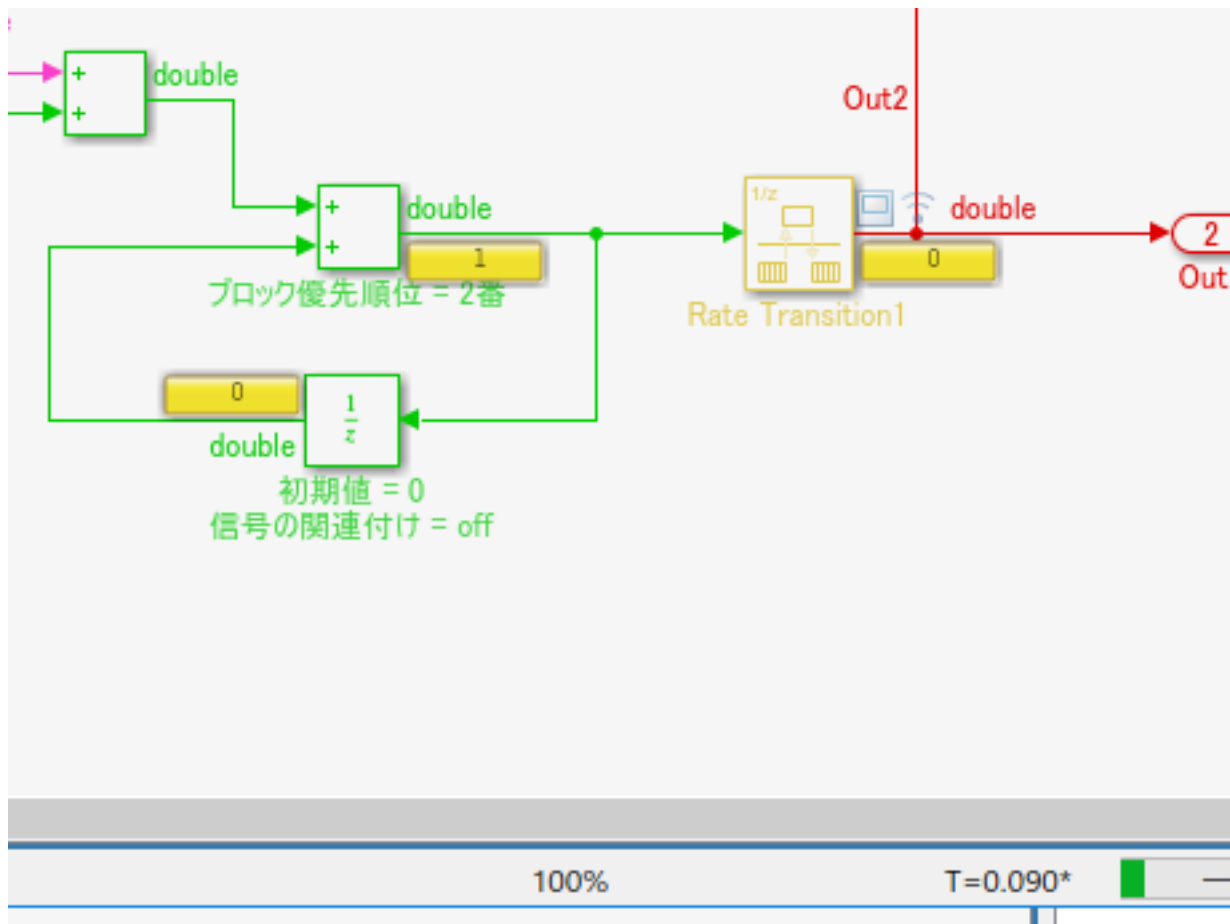
デバッグモードで比較する

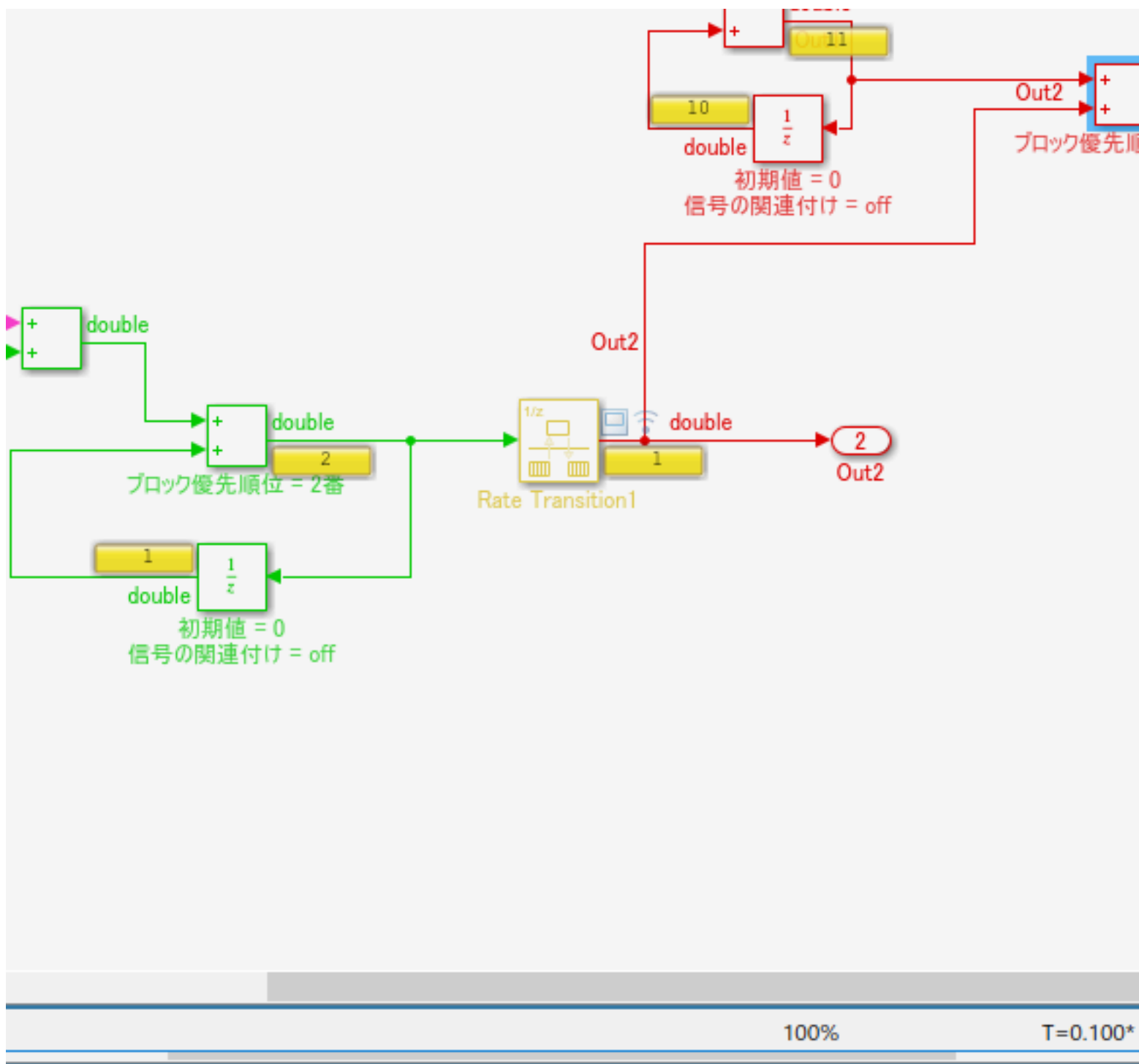
モデル1





モデル 2





実行順序の設定を行ったが、Rate Transitonがタイミングのずれを吸収し、結果には影響を及ぼさないようになっている。

コンフィギュレーションの影響



元々のコンフィギュレーション

周期的なサンプル時間に制約 : 制約なし

各離散レートを個別のタスクとして扱う チェックなし。

🔍 検索

ソルバー

データのインポート/エクス...  
数学とデータ型

▶ 診断

ハードウェア実行

モデル参照

シミュレーション ターゲット

▶ コード生成

▶ カバレッジ

シミュレーション時間

開始時間: 0.0

終了

ソルバーの選択

タイプ: 固定ステップ

ソルバー: 離散

▼ ソルバーの詳細

固定ステップ サイズ (基本サンプル時間): 0.01

タスクとサンプル時間オプション

周期的なサンプル時間の制約: 制約なし

☐ 各離散レートを個別のタスクとして扱う

☐ ターゲット上でタスクの同時実行を許可

☒ データ転送に対するレート変換を自動的に扱う

確定的なデータ転送: 可能な限り

☐ 優先順位の値が高いほどタスクの優先順位が高いこと

レートを個別指定

上の評価同様に、レートが遅い周期の優先順位を上げる。

## ▼ ソルバーの詳細

固定ステップ サイズ (基本サンプル時間): 0.01

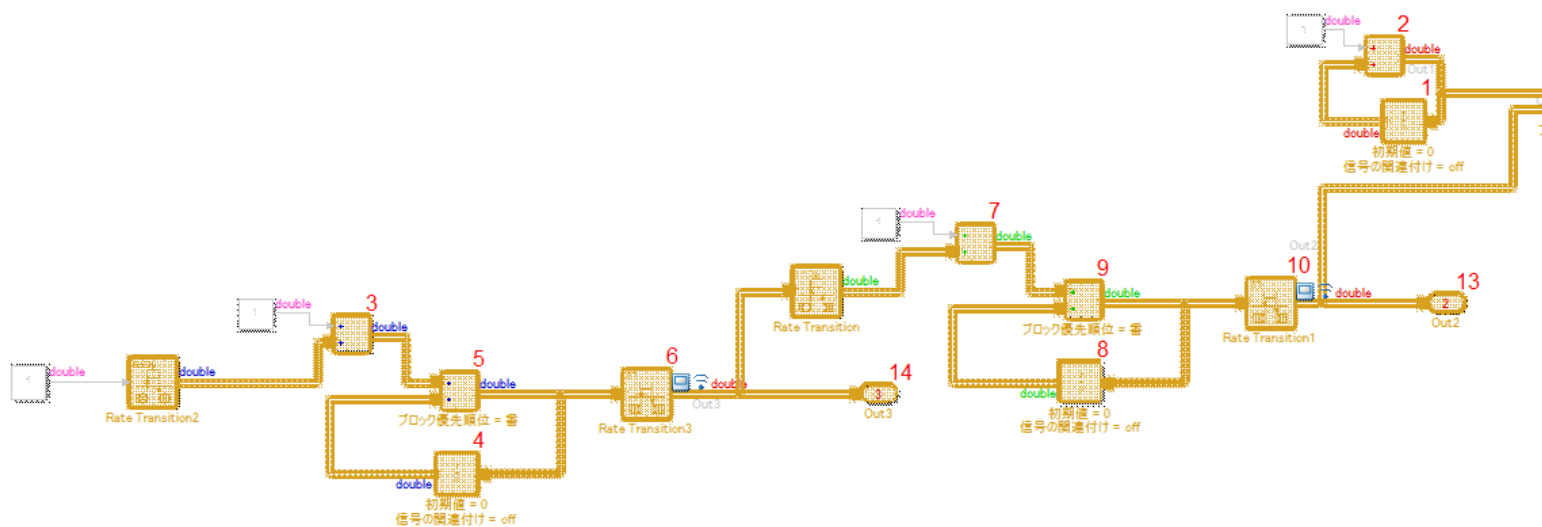
### タスクとサンプル時間オプション

周期的なサンプル時間の制約: サンプル時間を制約する

サンプル時間のプロパティ:  $[[0.01, 0, 3]; [0.1, 0, 2]; [0.2, 0, 1];]$

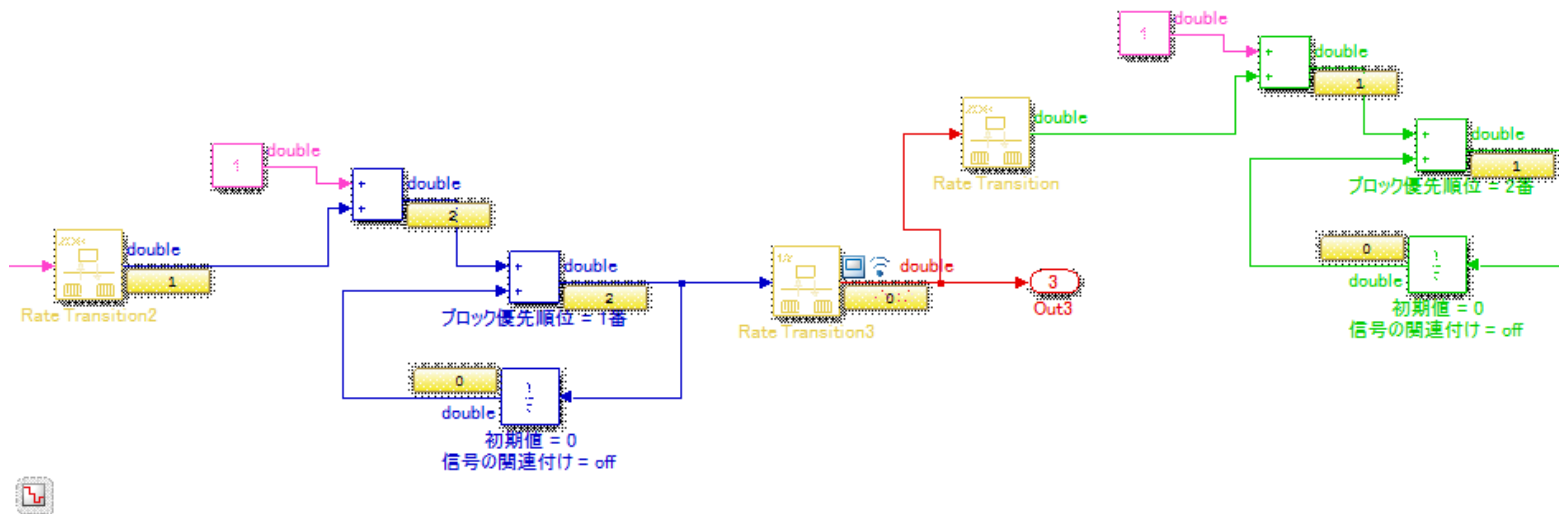
- ☐ 各離散レートを個別のタスクとして扱う
- ☐ データ転送に対するレート変換を自動的に取り扱う
- ☐ 優先順位の値が高いほどタスクの優先順位が高いことを示す

実行順序を表示すると、同上と同じように左側に置いたレートの遅い物の実行順序が早くなった。



```
open_system("timebasedmodel2.slx");
set_param(bdroot, 'SampleTimeColors', 'on')
sim(bdroot);
save('matlog_model2.mat', 'logout_modle2');
q_ans=questdlg({'モデルの表示を確認して先に進んでください。', '先に進みますか'});
if ~strcmp(q_ans, 'Yes')
    return;
end
```

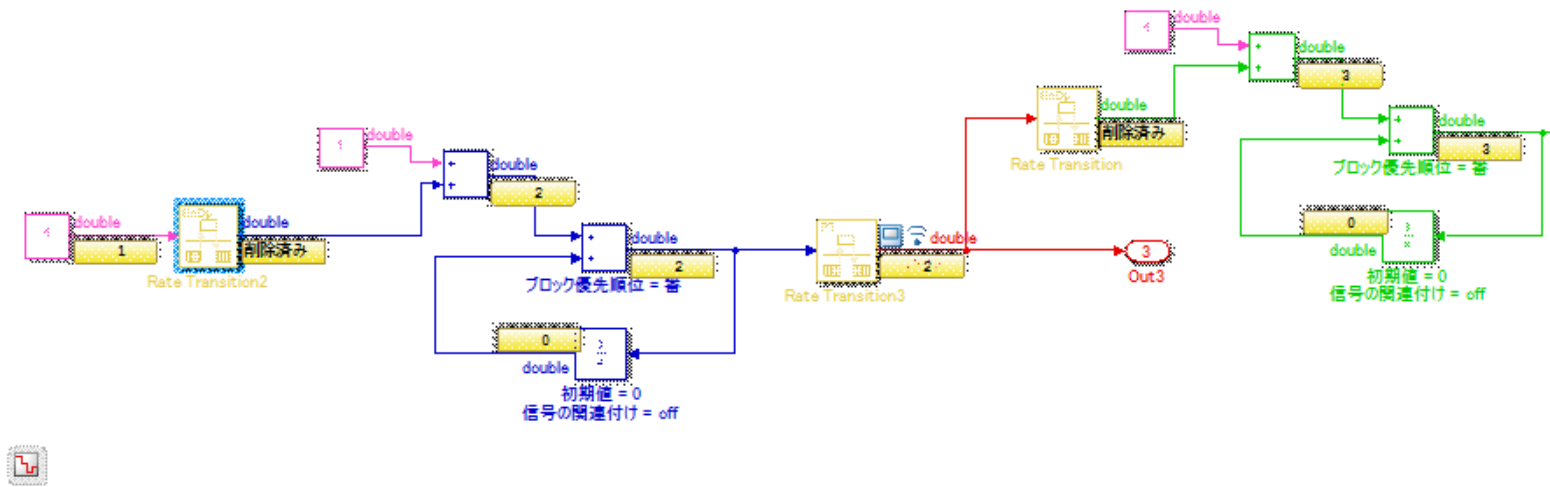
### timebasedmodel1の1回目の実行結果



## timebasedmodel2の実行結果

Rate Tranjitionが仕事をしていません。

データも一部最適化で削除されているのが良くわかる。



## コンフィギュレーションの影響

先ほどのコンフィギュレーションは、実行結果に影響を及ぼしたが、実行順序の表示には影響を及ぼさなかったため、









元の`timebasedmodel0`に対して、各離散レートを個別のタスクとして扱うをチェックする



```
open_system("timebasedmodel3.slx");
set_param(bdroot, 'SampleTimeColors', 'on')
sim("timebasedmodel3.slx");
save('matlog_model3.mat', 'logout_model3');
```

```
q_ans=questdlg({'モデルの表示を確認して先に進んでください。','先に進みますか'})  
if ~strcmp(q_ans,'Yes')  
    return;  
end
```

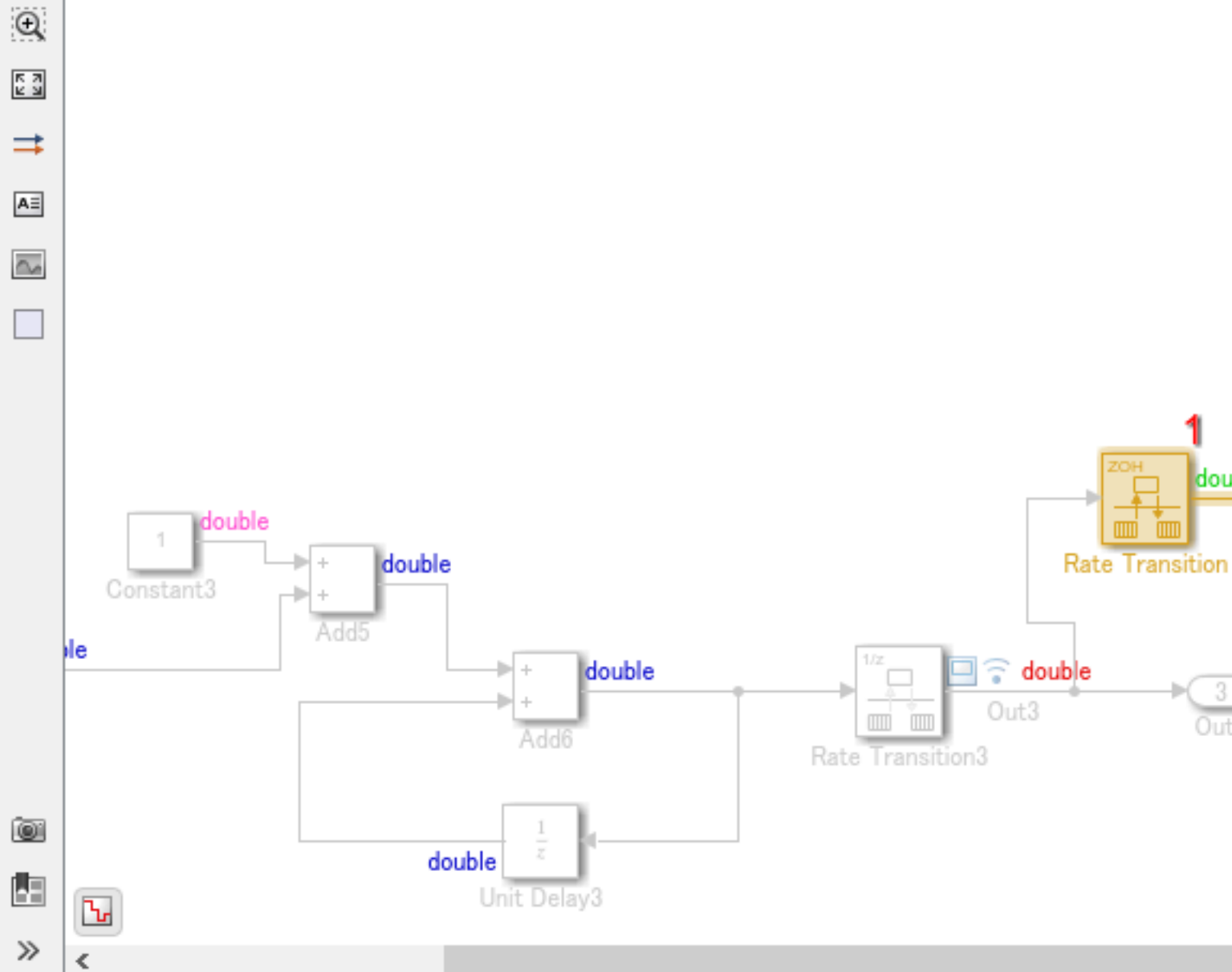
実行順序の表示

シミュレーション	デバッグ	モデル化	書式設定	アプリ
 パフォーマンス アドバイザー パフォーマンス	 診断 情報の オーバーレイ 診断	信号のトレース  コメントアウト  出力値  ツール	一時停止時間 (秒) <input type="text"/> ブレークポイントの追加  ブレークポイントリスト  ブレークポイント	 モデルの 更新 コンパイル

ツール(T) | SimNavi(N)

timebasedmodel3

timebasedmodel3



タスクの選択を切り替えると、表示が変わっていく。

タスク間の実行順序は示せないが、強引なレート設定を行わない限りは、タスク間の表示に意味が無い

(Rate **Tranjition**が更新タイミングを調整するので、タスク間の実行タイミングは前後しても結果に影響が出にくい)