In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

In [2]:

```
from analyzer import analyzer
from gates import dff, And
from utils import to_binary, to_lists
```

In [3]:

```
# D flip-flop
d_ff = dff()
d_in  = [0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0]
reset = [0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
clock = [0, 1] * 10
```

In [4]:

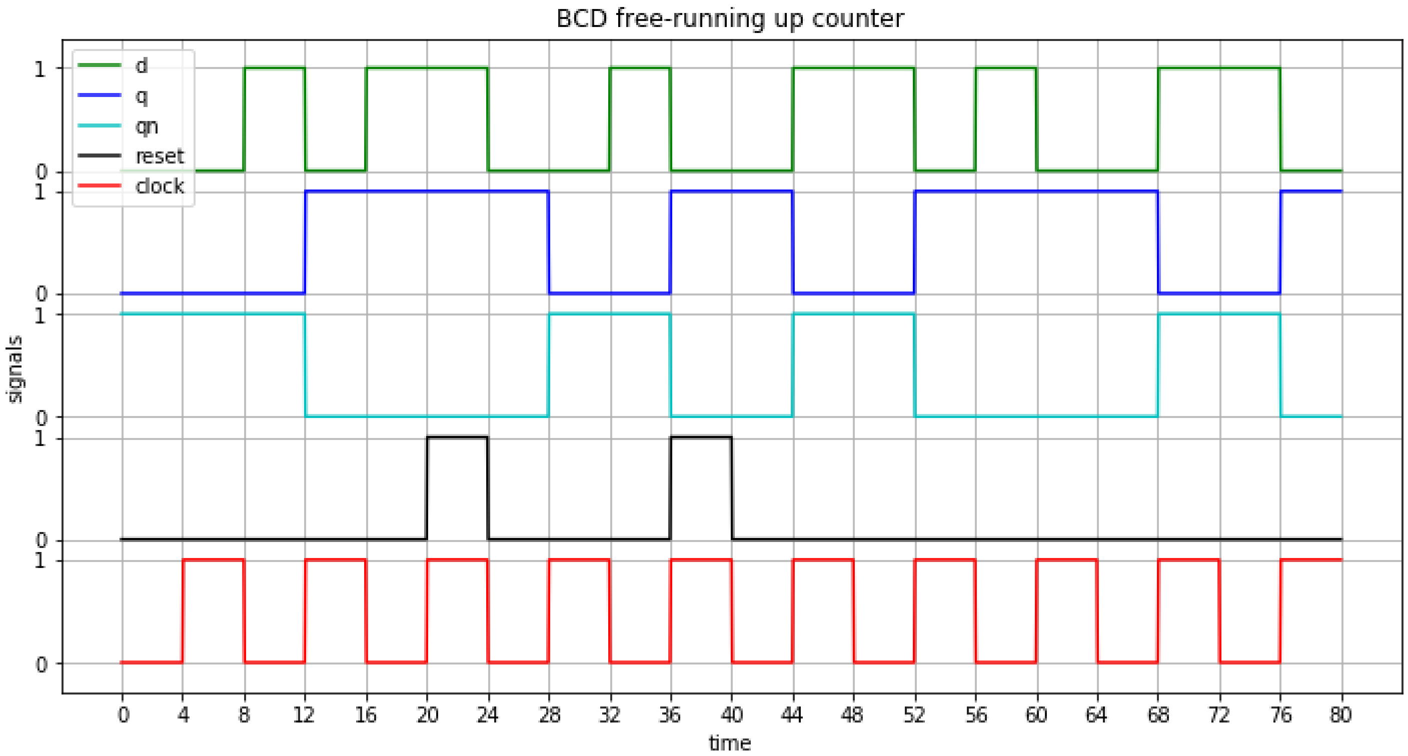
```
ticks = 4 # four time units per half-cycle

# signals of interest
d, q, qn = [], [], []
rst, clk = [], []

def record_data(a, b, c, dd, e):
    clk.append(a)
    rst.append(b)
    d.append(c)
    q.append(dd)
    qn.append(e)

for i in range(len(d_in)):
    for tick in range(ticks):
        d_ff.inputs(clock[i], reset[i], d_in[i])
        q1, q2 = d_ff.outputs()
        record_data(clock[i], reset[i], d_in[i], q1, q2)
```

In [5]:

```
signal = [d, q, qn, rst, clk]
name = ["d", "q", "qn", "reset", "clock"]
color = ["g-", "b-", "c-", "k-", "r-"]
```

In [6]:

```
x = analyzer("BCD free-running up counter", signal, name, color, 12, 6, ticks)
```



In [7]:

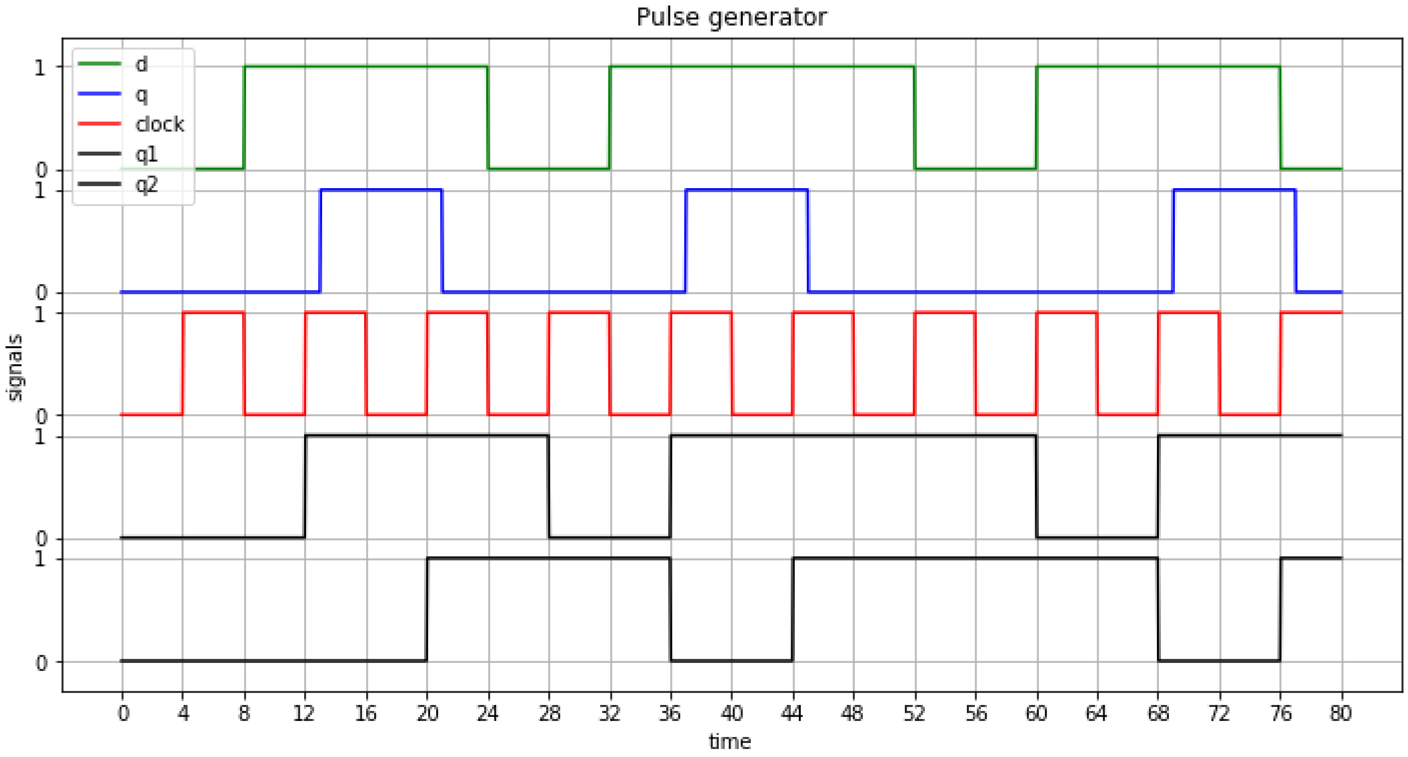
```
# pulse generator
d1 = dff()
d2 = dff()
a = And()
d_in  = [0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0]
clock = [0, 1] * 10
```

In [8]:

```
ticks = 4 # four time units per half-cycle

# signals of interest
d, q, q1, q2 = [], [], [], []
clk = []

def record_data(a, b, c, dx, e):
    clk.append(a)
    d.append(b)
    q.append(c)
    q1.append(dx)
    q2.append(e)

q1x = 0
q2xn = 1
for i in range(len(d_in)):
    for tick in range(ticks):
        d1.inputs(clock[i], 0, d_in[i])
        d2.inputs(clock[i], 0, q1x)
        a.inputs(q1x, q2xn)
        q1x, _ = d1.outputs()
        q2x, q2xn = d2.outputs()
        record_data(clock[i], d_in[i], a.output(), q1x, q2x)
```

In [9]:

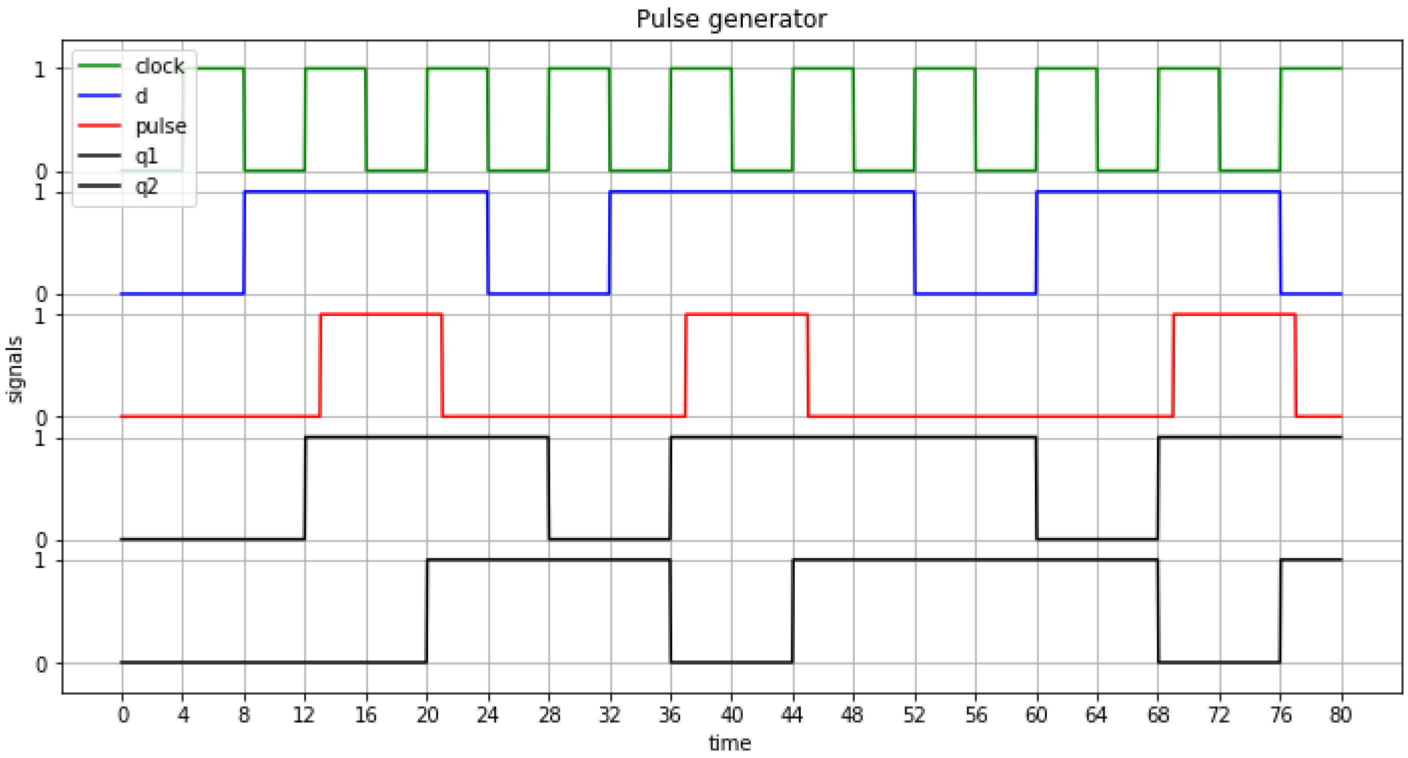
```
signal = [d, q, clk, q1, q2]
name = ["d", "q", "clock", "q1", "q2"]
color = ["g-", "b-", "r-", "k-", "k-"]
x = analyzer("Pulse generator", signal, name, color, 12, 6, ticks)
```



In [10]:

```
# pulse generator

# circuit elements
d1 = dff()
d2 = dff()
a = And()

# input data
d_in  = [0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0]
clock = [0, 1] * 10

ticks = 4 # inputs are mantained four times

data = pd.DataFrame()

# initial states
q1x = 0
q2xn = 1

for i in range(len(d_in)):
    for tick in range(ticks):

        ## logic circuit
        d1.inputs(clock[i], 0, d_in[i])
        d2.inputs(clock[i], 0, q1x)
        a.inputs(q1x, q2xn)

        ## circuit signals
        q1x, _ = d1.outputs()
        q2x, q2xn = d2.outputs()

        ## data recording
        data = data.append(
            {'clock': clock[i],
             'd': d_in[i],
             'pulse': a.output(),
             'q1':q1x,
             'q2': q2x},
            ignore_index=True).astype(int)

data = data[['clock', 'd', 'pulse', 'q1', 'q2']]
```

In [11]:
```
signal, name = to_lists(data)
color = ["g-", "b-", "r-", "k-", "k-"]
x = analyzer("Pulse generator", signal, name, color, 12, 6, ticks)
```



In [ ]: