Tahmini Ders İçeriği
(Tentative Couse Schedule – Syllabus)

TOBB ETÜ
Ekonomi ve Teknoloji Üniversitesi

1. Hafta: Sayısal Sinyaller/Sistemler, İkilik Tabanda Sayılar, Taban Aritmetiği, İşaretli/Eksi Sayıların Gösterimi, Sayısal Tasarım Tarihçesi

2. Hafta: İkili Mantık Aritmetiği ve Kapıları, Bool Cebiri Teorisi ve Tanımları, Bool Fonksiyonları, Kapı-Seviyesinde Yalınlaştırma, Karnough Haritası, Önemsenmeyen Durumlar, NAND, NOR, XOR

3-4. Hafta: FPGA, Birleşik (Combinational) Devreler, Aritmetik Modüller, Decoder, Encoder, Mux, Verilog HDL

5-6. Hafta: **(10-14, 17-21 Ekim)** Ardışık (Sequential) Devreler, Mandal (Latch), Flip-Flop, Zamanlama (Timing)

Lab Sınavı (265/264L) **(19 Ekim)**

Proje Duyurusu

7. Hafta: **(24-28 Ekim)** Durum Makinaları, Örnek Tasarımlar

8. Hafta: **(31 Ekim – 4 Kasım)** Yazmaçlar (Registers), Sayaçlar (Counters)

Ara Sınav (265/264) **(15 Kasım)**

9. Hafta: **(7-11 Kasım)** Bellekler, FPGA'da Block RAM, OpenRAM

10. Hafta: **(14-18 Kasım)** RTL (Register Transfer Level) ASMD (Algorithmic State Machine and Datapath) Tasarımları

11-12. Hafta: **(21-25 Kasım, 28 Kasım – 2 Aralık)** Boru hattı, FPGA ve ASIC Tasarım Akışları

Final **(7 Aralık)** – Proje Teslimleri **(18 Aralık)**

Birleşik (Combinational) Devreler

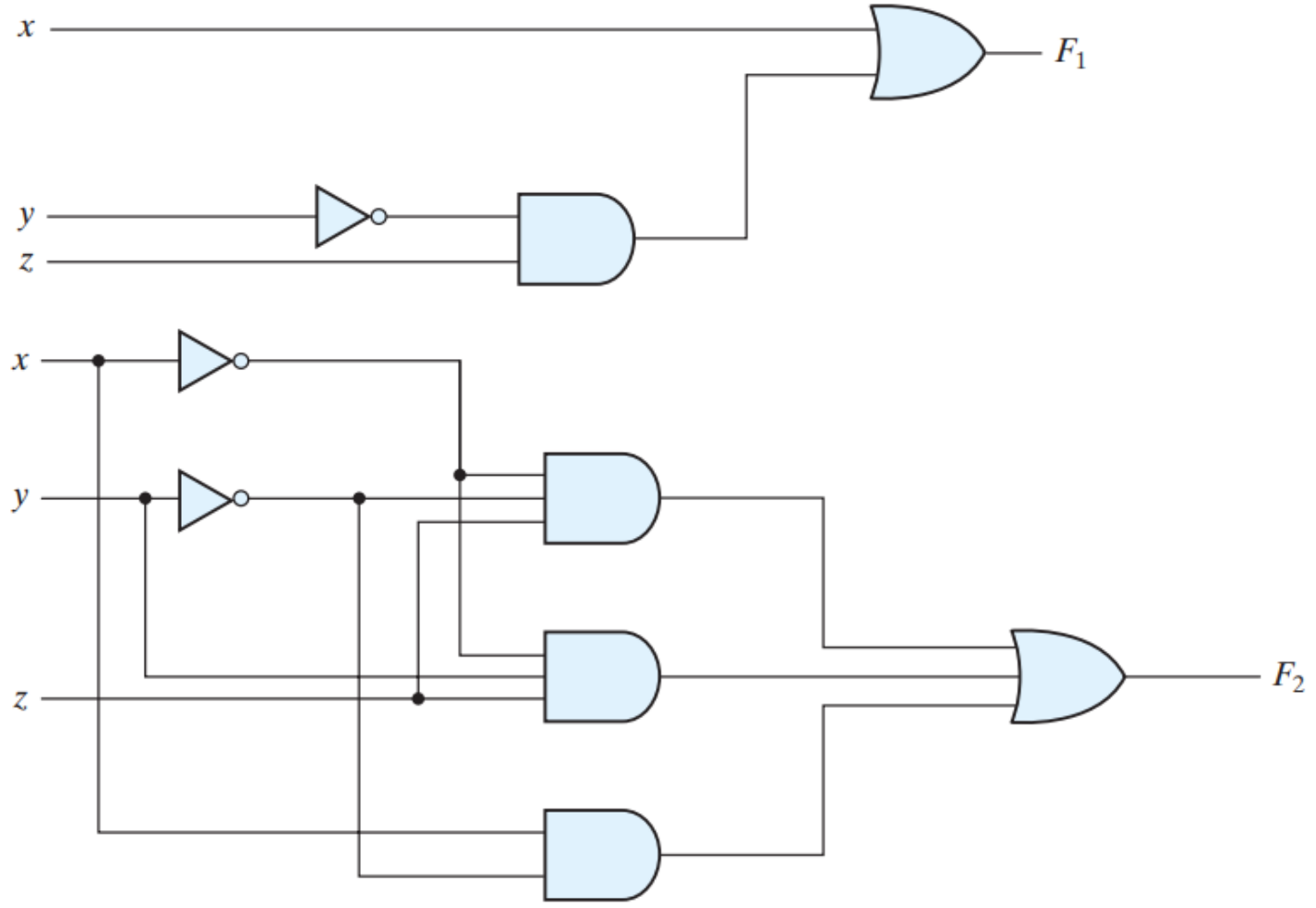
Çıkış sinyali, yalnızca o anki giriş sinyallerinin değerine bağlı olan devreler

Geçmiş giriş sinyal değerlerinin çıkış sinyaline etkisi yok

F1 = x + y'z

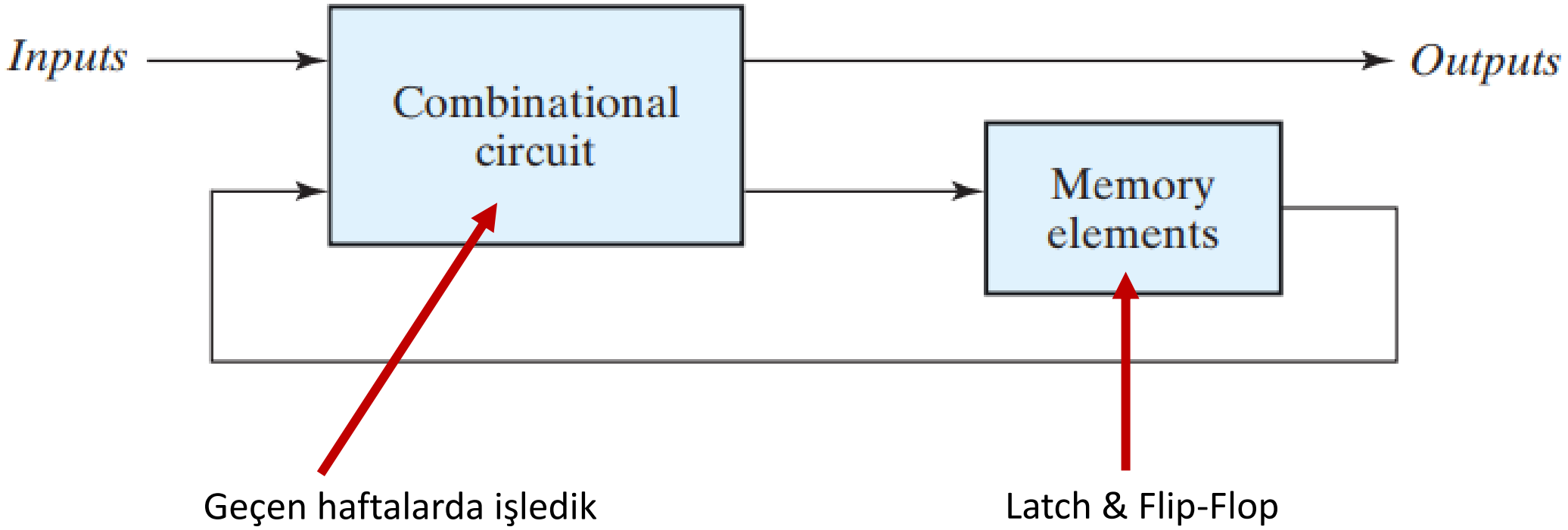
F2 = x'y'z + x'yz + xy'

F1 & F2				
Giriş			Çıkış	
x	y	z	F1	F2
0	0	0	0	0
0	0	1	1	1
0	1	0	0	0
0	1	1	0	1
1	0	0	1	1
1	0	1	1	1
1	1	0	1	0
1	1	1	1	0

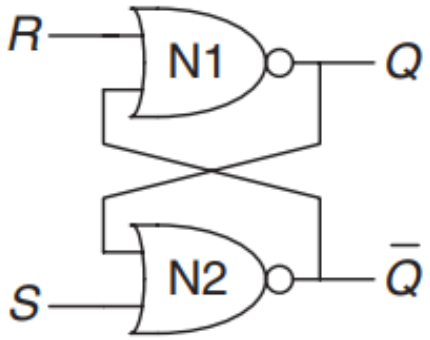


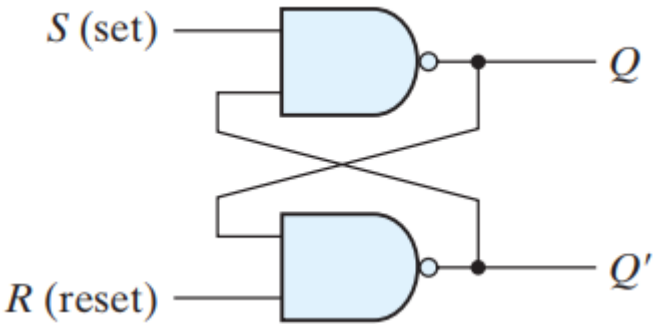
Sıralı (Sequential) Devreler

Çıkış sinyali, giriş sinyallerinin hem o anki hem de geçmiş değerine bağlı olan devreler. İçerisinde bellek (memory) birimi bulundururlar. Devre içerisindeki belleğin o anki değeri, devrenin durumu (state) olarak adlandırılır.



SR (Set/Reset) Mandal (Latch)





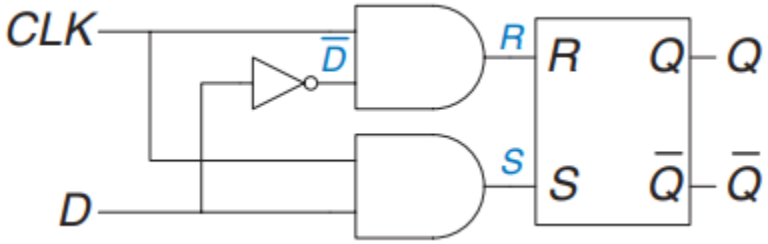
Durum1: R=1 & S=0 → Q = 0, Q_bar = 1
Durum2: R=0 & S=1 → Q_bar = 0, Q = 1
Durum3: R=1 & S=1 → Q = 0, Q_bar = 0
Durum4: R=0 & S=0 → Q = Q_prev, Q_bar = Q_bar_prev

Case	S	R	Q	$\bar{Q}$
IV	0	0	Q_{prev}	$\bar{Q}_{prev}$
I	0	1	0	1
II	1	0	1	0
III	1	1	0	0

Problem: S ve R aynı anda 1 olduğunda istenmeyen durum oluşuyor
Çözüm: Tasarım öyle düzenlenmeli ki S ve R aynı anda hiç ‘1’ olmasın

Bunu sağlamak mümkün olmayabilir ya da tasarımı zorlaştırabilir
Çözüm: Mandal tipini değiştirmek → D-Latch

D Mandal (Latch)



CLK	D	$\overline{D}$	S	R	Q	$\overline{Q}$
0	X	$\overline{X}$	0	0	Q_{prev}	$\overline{Q}_{prev}$
1	0	1	0	1	0	1
1	1	0	1	0	1	0

D mandal sayesinde S ve R aynı anda ‘1’ olamıyor

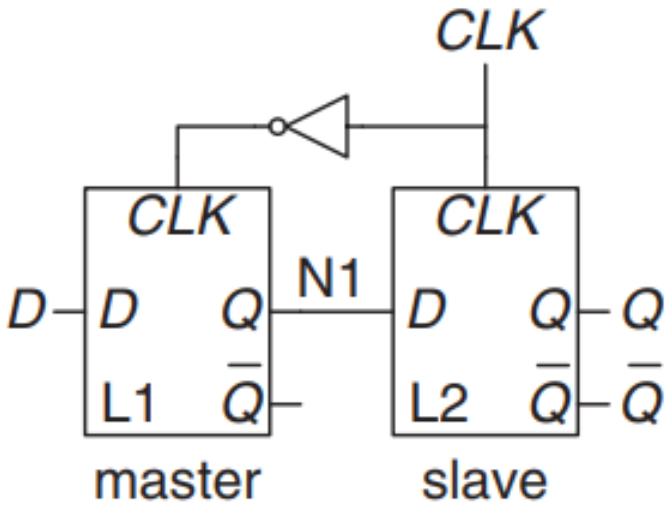
CLK sinyali ‘1’ olduğu zaman Q değeri D’ye eşit oluyor. CLK sinyali ‘0’ olduğu zaman S ve R ‘0’ oluyor ve Q değeri önceki durumunu, yani değerini koruyor

CLK sinyali ‘1’ oldukça Q sinyaline D değeri atanıyor. Atamaların sadece belirli “an”larda yapılması zamanlama analizi ve sistemin senkronize olması açısından daha kolay olacaktır.

Belirli “an”larda değer ataması yapılması → Flip-Flop

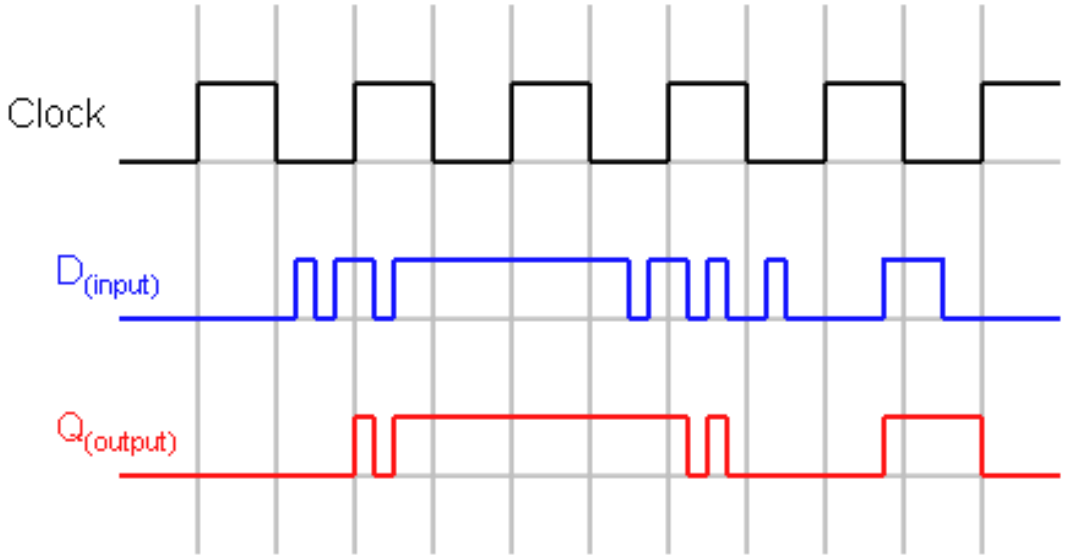
Latch : Level-sensitive
Flip-Flop : Edge-sensitive

D Flip-Flop

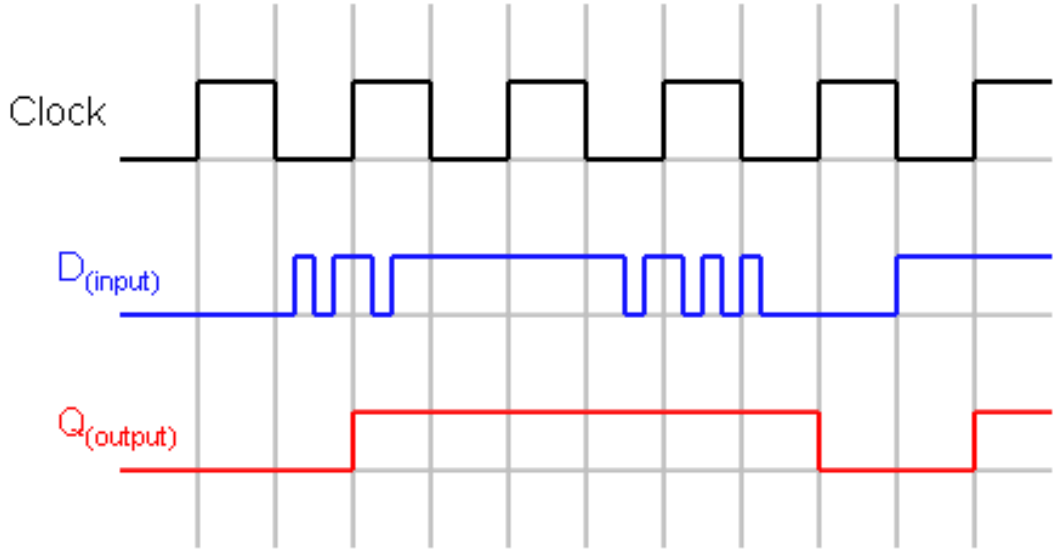


In other words, a D flip-flop copies D to Q on the rising edge of the clock and remembers its state at all other times. Reread this definition until you have it memorized; one of the most common problems for beginning digital designers is to forget what a flip-flop does.

D Latch



D Flip-Flop



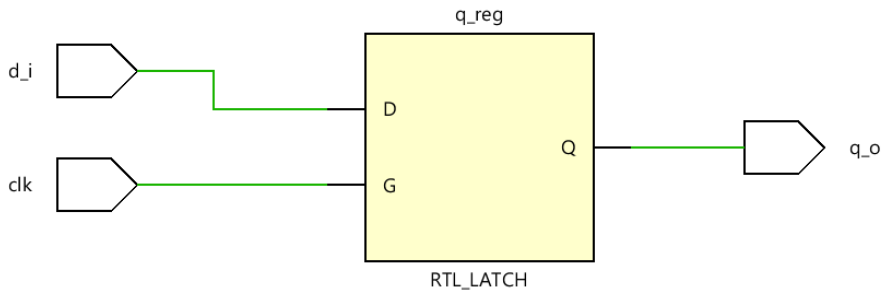
D Latch - Verilog

TOBB ETÜ
Ekonomi ve Teknoloji Üniversitesi

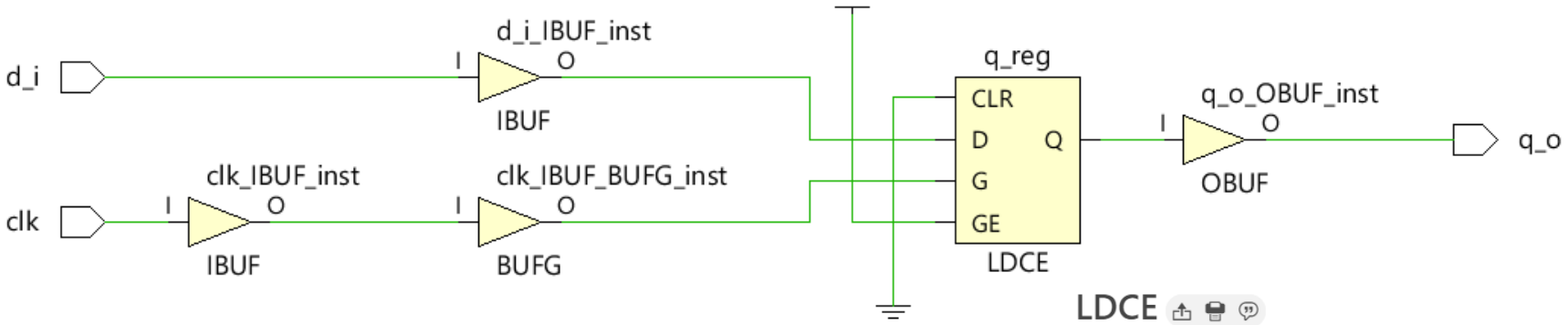
```
module d_latch
(
input d_i,
input clk,
output q_o
);

reg q;

always@(clk,d_i) begin
    if (clk == 1'b1) begin
        q <= d_i;
    end
end

assign q_o = q;

endmodule
```

Non-Blocking Assignment

Vivado Elaboration



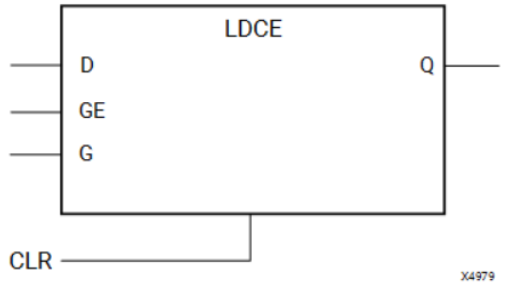
Vivado Synthesis



LDCE

Primitive: Transparent Latch with Clock Enable and Asynchronous Clear

- PRIMITIVE_GROUP: REGISTER
- PRIMITIVE_SUBGROUP: LATCH
- Families: UltraScale, UltraScale+



Gate Input

Synthesis (2 warnings)

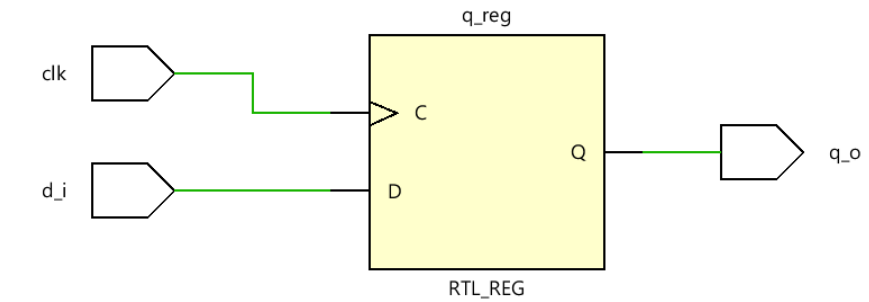
[Synth 8-327] inferring latch for variable 'q_reg' [d_latch.v:34]

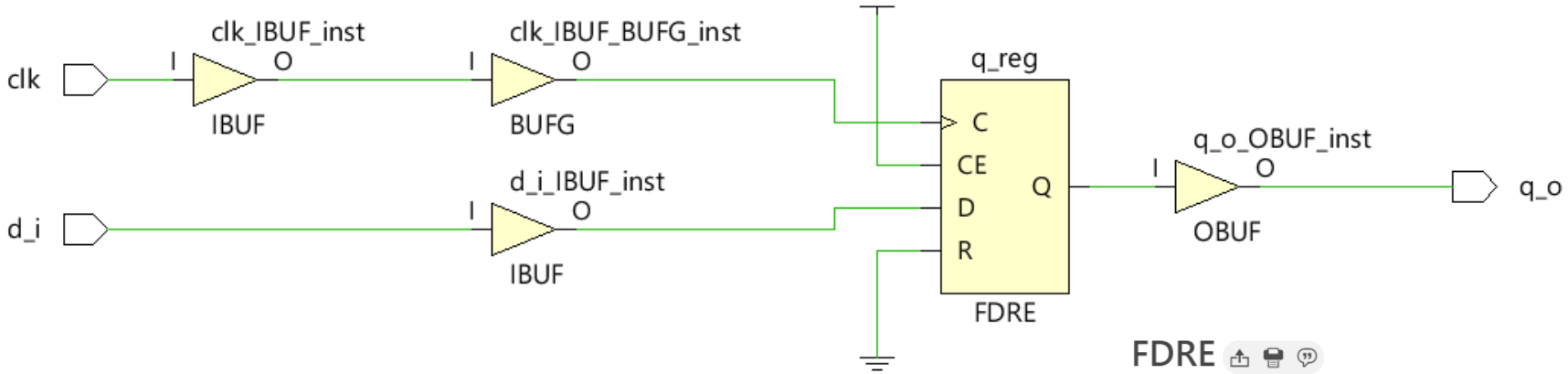
D FF - Verilog

TOBB ETÜ
Ekonomi ve Teknoloji Üniversitesi

```
module d_ff
(
input d_i,
input clk,
output q_o
);

reg q;

always@(posedge clk) begin
    q <= d_i;
end

assign q_o = q;

endmodule
```

Non-Blocking Assignment



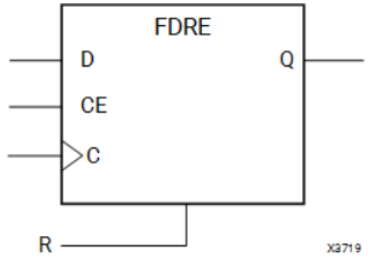


Resource	Estimation	Available	Utilization %
FF	1	41600	0.01
IO	3	106	2.83
BUFG	1	32	3.13

FDRE

Primitive: D Flip-Flop with Clock Enable and Synchronous Reset

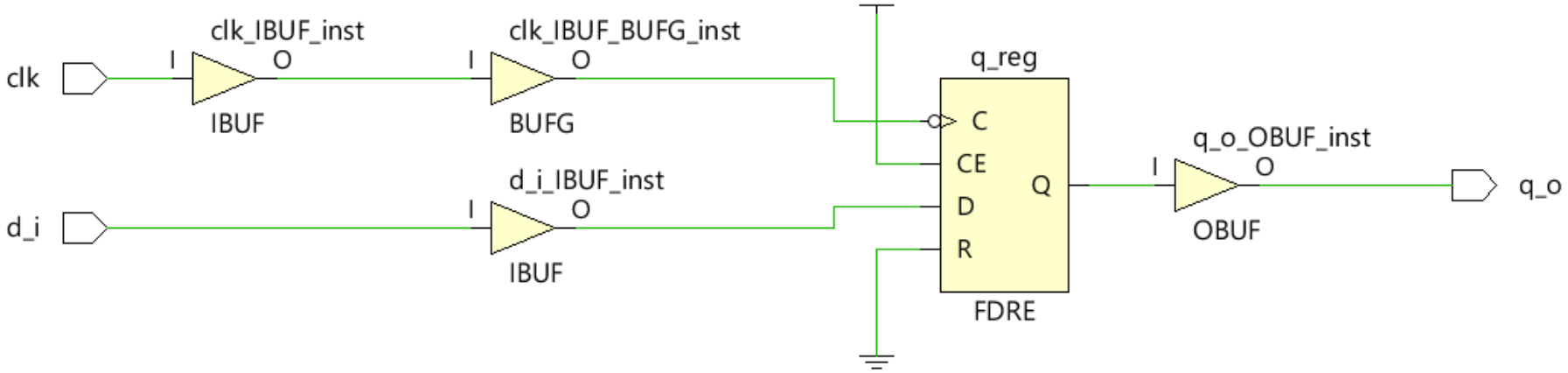
- PRIMITIVE_GROUP: REGISTER
- PRIMITIVE_SUBGROUP: SDR
- Families: UltraScale, UltraScale+



D FF - Verilog

TOBB ETÜ
Ekonomi ve Teknoloji Üniversitesi

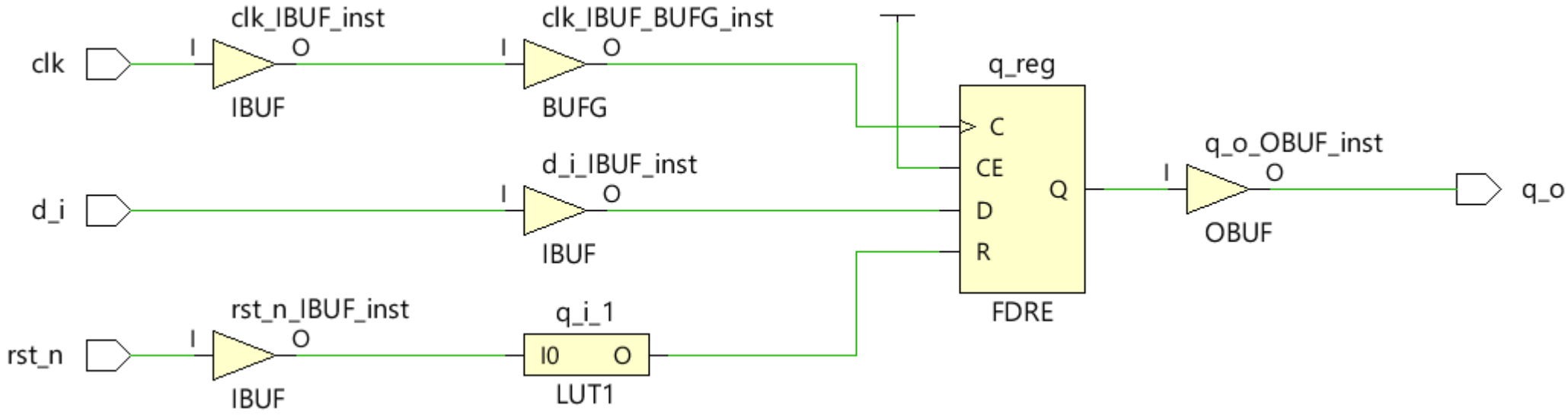
```
module d_ff
(
input d_i,
input clk,
output q_o
);

reg q;

always@(negedge clk) begin
    q <= d_i;
end

assign q_o = q;

endmodule
```



D FF | Active-Low Reset - Verilog

TOBB ETÜ
Ekonomi ve Teknoloji Üniversitesi

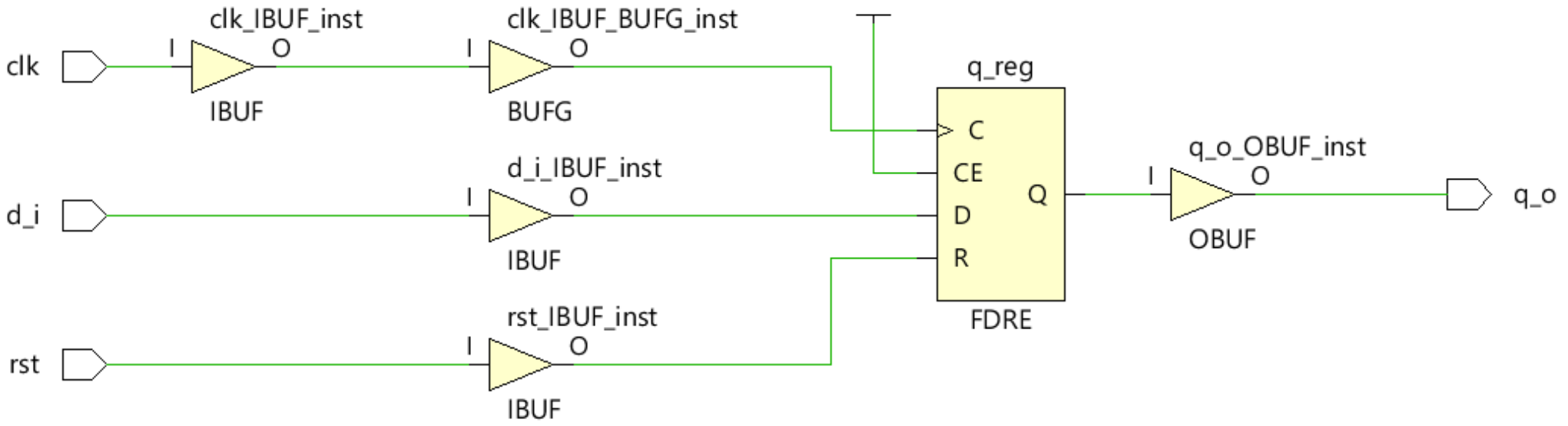
```verilog
module d_ff_rst
(
input d_i,
input rst_n,    // Active-low reset
input clk,
output q_o
);

reg q;

always@(posedge clk) begin
    if (rst_n == 1'b0) begin
        q <= 1'b0;
    end
    else begin
        q <= d_i;
    end
end

assign q_o = q;

endmodule
```



D FF | Active-High Reset - Verilog

TOBB ETÜ
Ekonomi ve Teknoloji Üniversitesi

```verilog
module d_ff_rst
(
input d_i,
input rst,    // Active-high reset
input clk,
output q_o
);

reg q;

always@(posedge clk) begin
    if (rst == 1'b1) begin
        q <= 1'b0;
    end
    else begin
        q <= d_i;
    end
end

assign q_o = q;

endmodule
```



D FF | Asenkron Reset - Verilog

TOBB ETÜ
Ekonomi ve Teknoloji Üniversitesi

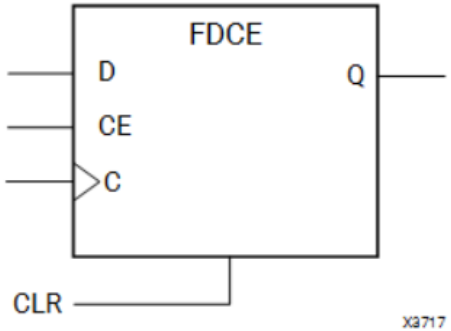
```verilog
module d_ff_rst
(
input d_i,
input rst_n,    // Active-low reset
input clk,
output q_o
);

reg q;

// always@(posedge clk) begin // Synchronous reset
always@(posedge clk, negedge rst_n) begin   // Asynchronous reset
    if (rst_n == 1'b1) begin
        q <= 1'b0;
    end
    else begin
        q <= d_i;
    end
end

assign q_o = q;

endmodule
```

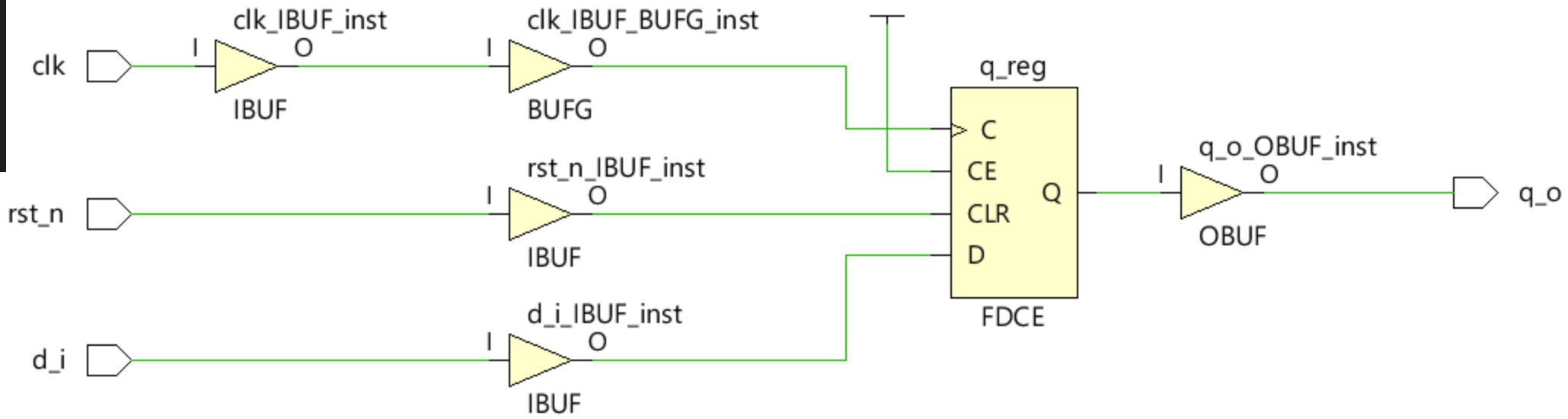
0 olacak ☺

FDCE

Primitive: D Flip-Flop with Clock Enable and Asynchronous Clear

- PRIMITIVE_GROUP: REGISTER
- PRIMITIVE_SUBGROUP: SDR
- Families: UltraScale, UltraScale+





Reset Gerekli mi? Senkron vs Asenkron Active-High vs Active-Low

TOBB ETÜ
Ekonomi ve Teknoloji Üniversitesi

UltraFast Design Methodology Guide (UG 949)

When and Where to Use a Reset

FPGA devices have dedicated global set/reset signals (GSR). These signals initialize all registers to the initial value specified state in the HDL code at the end of device configuration.

If an initial state is not specified, it defaults to a logic zero. Accordingly, every register is at a known state at the end of configuration, regardless of the reset topology specified in the HDL code. It is not necessary to code a global reset for the sole purpose of initializing the device on power-up.

Synchronous Reset vs. Asynchronous Reset

If a reset is needed, Xilinx recommends code synchronous resets. Synchronous resets have many advantages over asynchronous resets.

Control Signal Polarity (Active-High vs. Active-Low)

For high-fanout control signals like clock enables or resets, it is best to use active high in the entire design. If a block operates with active low resets or clock enables, inverters get added to the design and there is an associated timing penalty. It can restrict synthesis options to flat or rebuilt to optimize the inverters or require the implementation of a custom solution.

The Slice and internal logic of the Xilinx FPGA clock enables and resets are inherently active-High. Describing active-Low resets or clock enables may result in additional LUTs used as simple inverters for those routes.

Reset Gerekli mi? Senkron vs Asenkron
Active-High vs Active-Low

IC (Integrated Circuit) tasarımlarında FF'lar, Xilinx'in SRAM tabanlı Flip-Flop'ları gibi güç verilince kendi kendini başlangıç değeri alamıyor

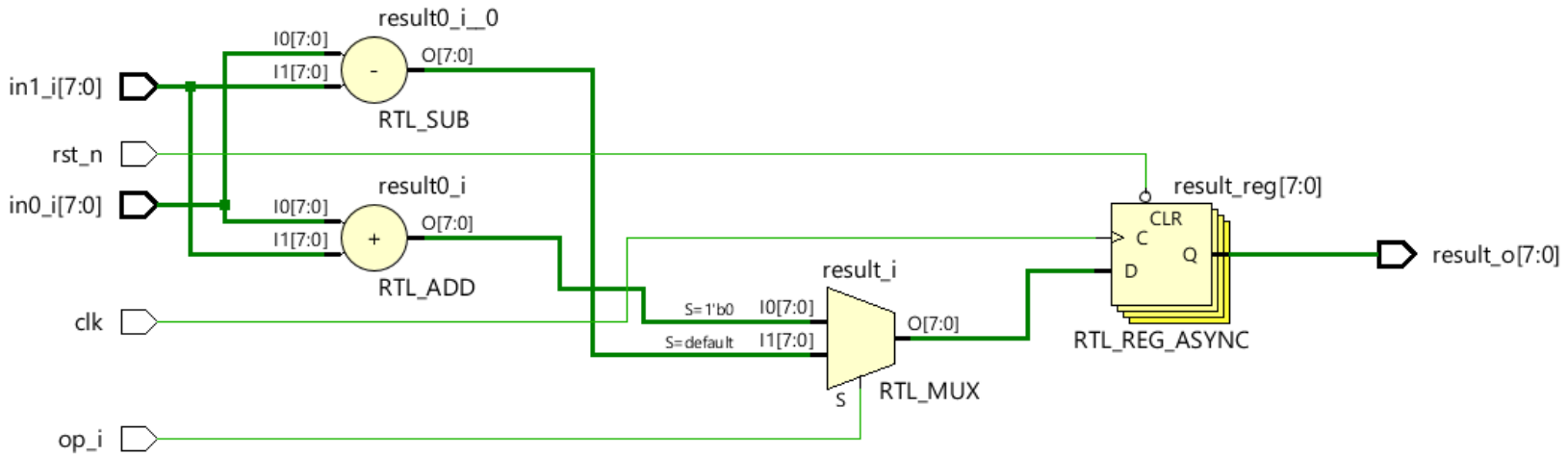
Pek çok flash tabanlı FPGA ailesinde de (Örn. Microchip SmartFusion) FF'lar başlangıç değeri alamıyor

Dolayısıyla reset sinyali ile FF'ları güç verilince resetlemek IC ve flash tabanlı FPGA'lar için gerekli. Yazılacak olan bir Verilog ya da VHDL kodunun sadece Xilinx FPGA'lar için uyumlu değil de IC ya da flash tabanlı FPGA'lar için de uyumlu olması isteniyorsa reset eklemek mantıklı

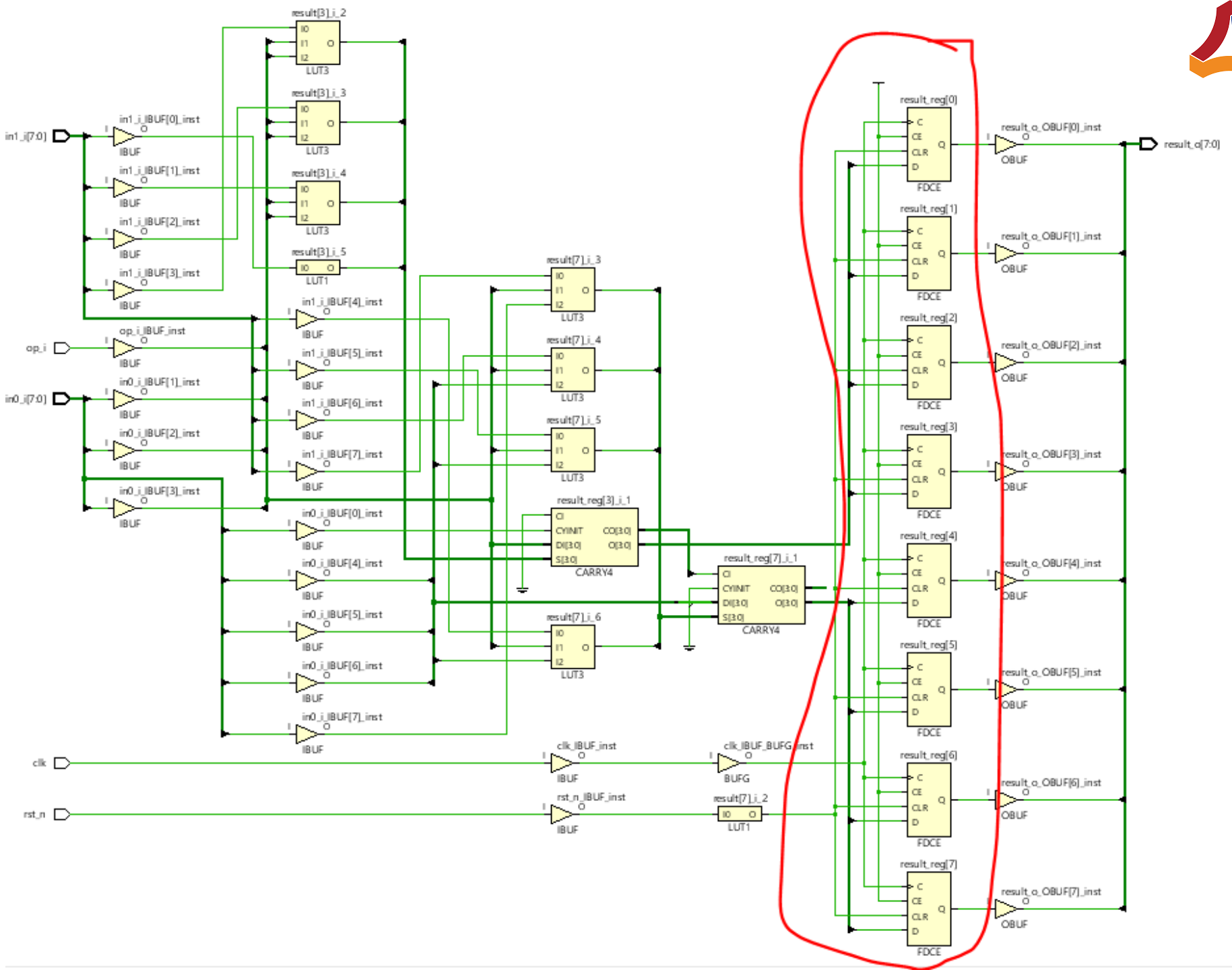
8-bit Add/Sub Modülünü Senkronize Etmek

TOBB ETÜ
Ekonomi ve Teknoloji Üniversitesi

```
module add_sub_8bit_sync
(
input [7:0] in0_i,
input [7:0] in1_i,
input op_i,
input clk,
input rst_n,
output [7:0] result_o
);

reg [7:0] result;

always @(posedge clk, negedge rst_n) begin
    if (rst_n == 1'b0) begin
        result <= 8'b00000000;
    end
    else begin
        if (op_i == 1'b0) begin
            result <= in0_i + in1_i;
        end
        else begin
            result <= in0_i - in1_i;
        end
    end

end

assign result_o = result;

endmodule
```



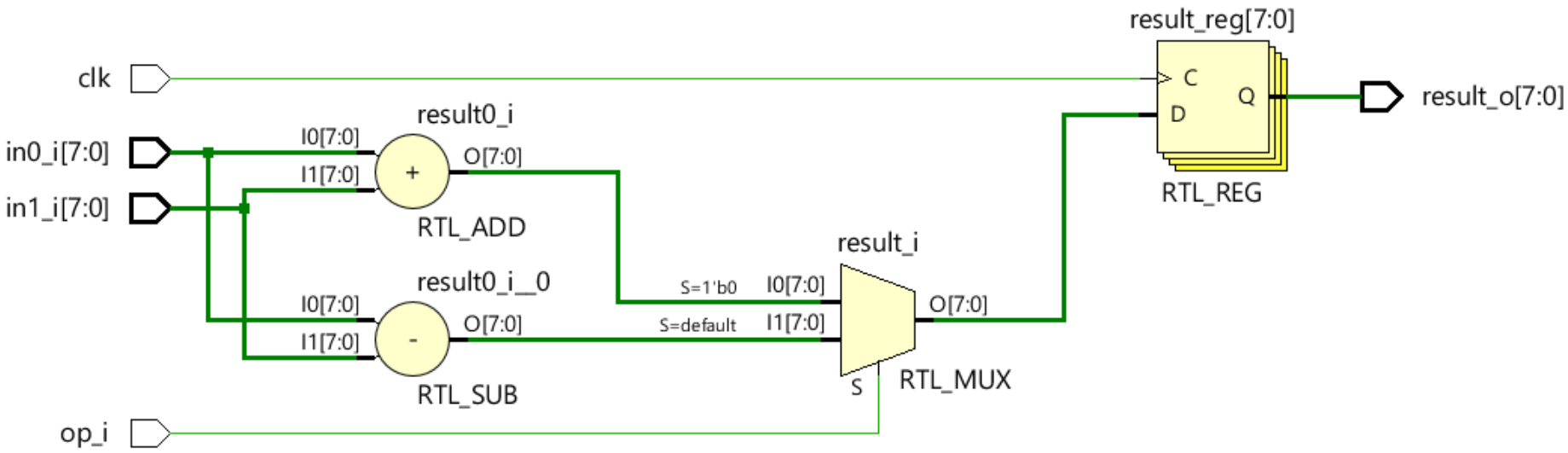
Resource	Estimation	Available	Utilization %
LUT	9	20800	0.04
FF	8	41600	0.02
IO	27	106	25.47
BUFG	1	32	3.13


in1_i[7:0]
op_i
in0_i[7:0]
clk
rst_n
result_o[7:0]
IBUF
OBUF
LUT3
LUT1
CARRY4
BUFG
FDCE

reg başlangıç değeri (initial value)

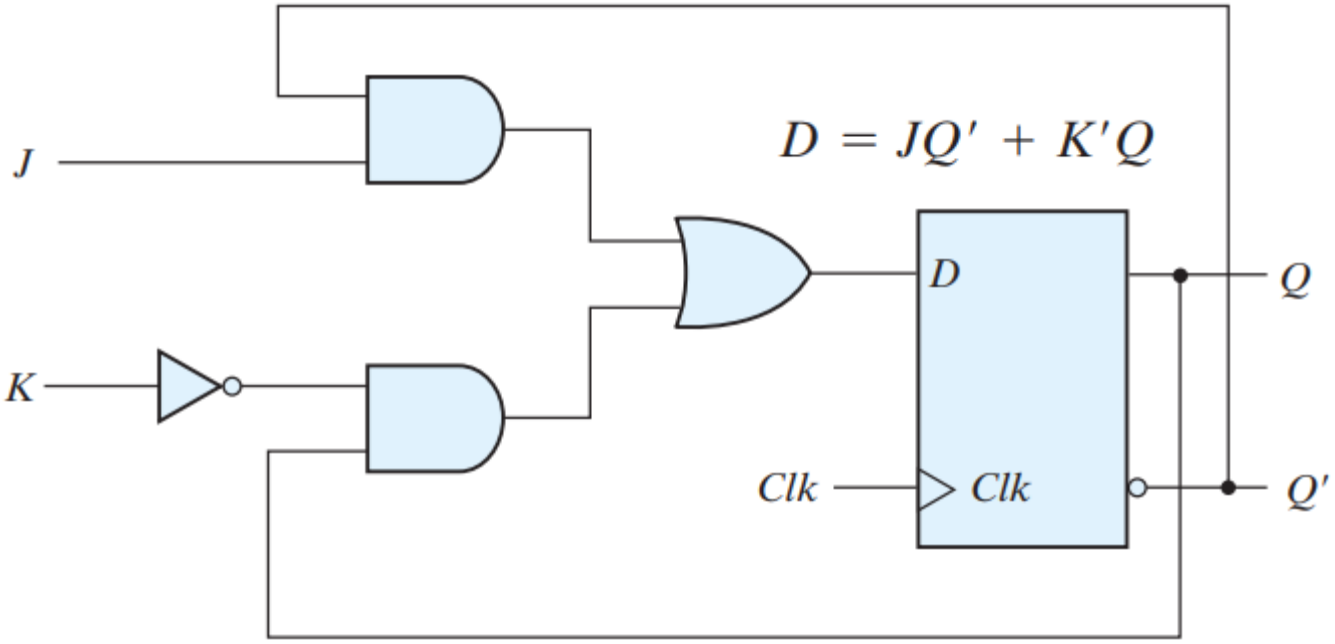
TOBB ETÜ
Ekonomi ve Teknoloji Üniversitesi

```
module add_sub_8bit_initval
(
input [7:0] in0_i,
input [7:0] in1_i,
input op_i,
input clk,
output [7:0] result_o
);

// only works on SRAM based FPGAs
reg [7:0] result = 8'b00000000;

always @(posedge clk) begin

    if (op_i == 1'b0) begin
        result <= in0_i + in1_i;
    end
    else begin
        result <= in0_i - in1_i;
    end

end

assign result_o = result;

endmodule
```



Resource	Estimation	Available	Utilization %
LUT	8	20800	0.04
FF	8	41600	0.02
IO	26	106	24.53
BUFG	1	32	3.13

Diğer FF Türleri: JK ve T



(a) Circuit diagram

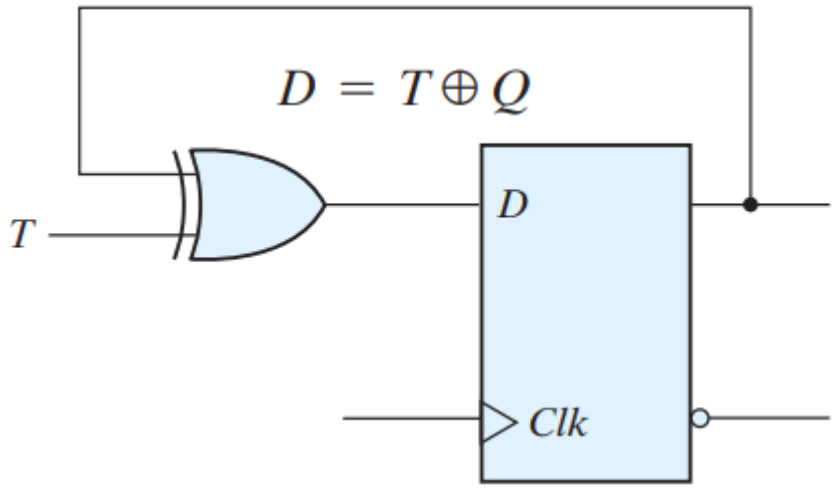


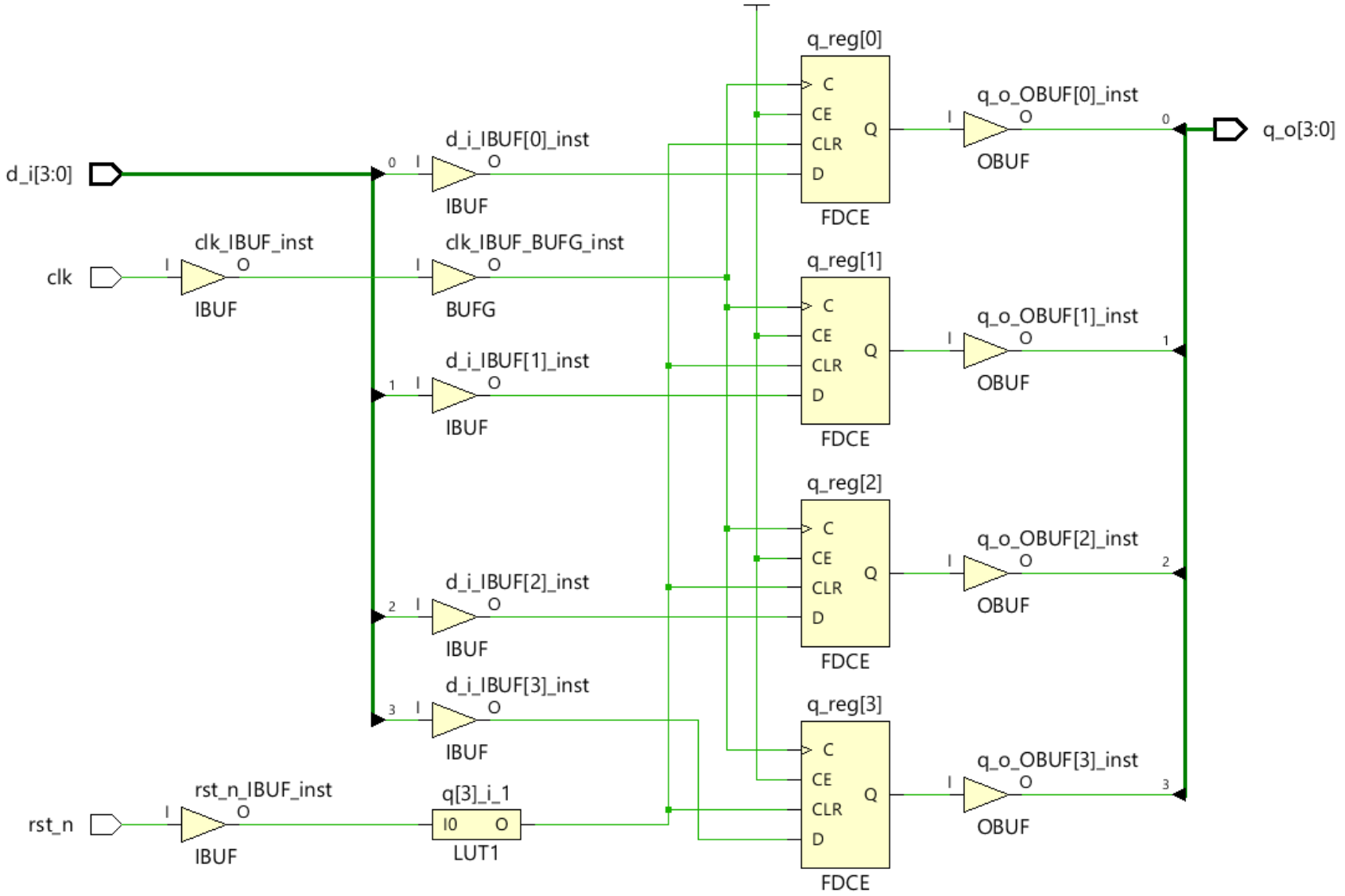
FIGURE 5.12
JK flip-flop

Yazmaç (Register)

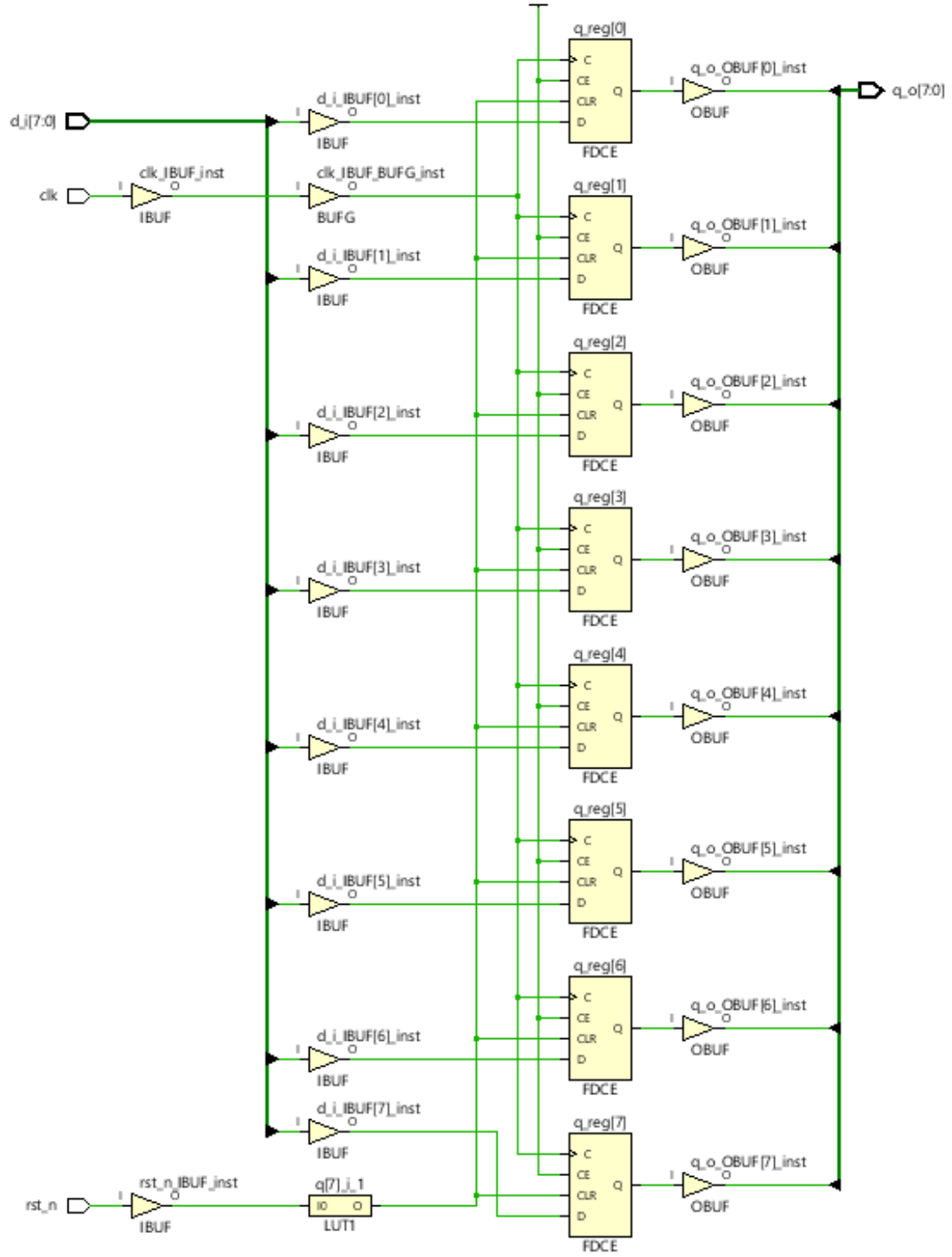
TOBB ETÜ
Ekonomi ve Teknoloji Üniversitesi

Gruplanmış FF'lardan oluşan ve aynı saat sinyali ile beslenen sayısal devredir

```
module reg4
(
input [3:0] d_i,
input rst_n,
input clk,
output [3:0] q_o
);

reg [3:0] q;

always @(posedge clk or negedge rst_n) begin
    if (rst_n == 1'b0) begin
        q <= 4'b0000;
    end
    else begin
        q <= d_i;
    end
end

assign q_o = q;

endmodule
```



Yazmaç (Register)



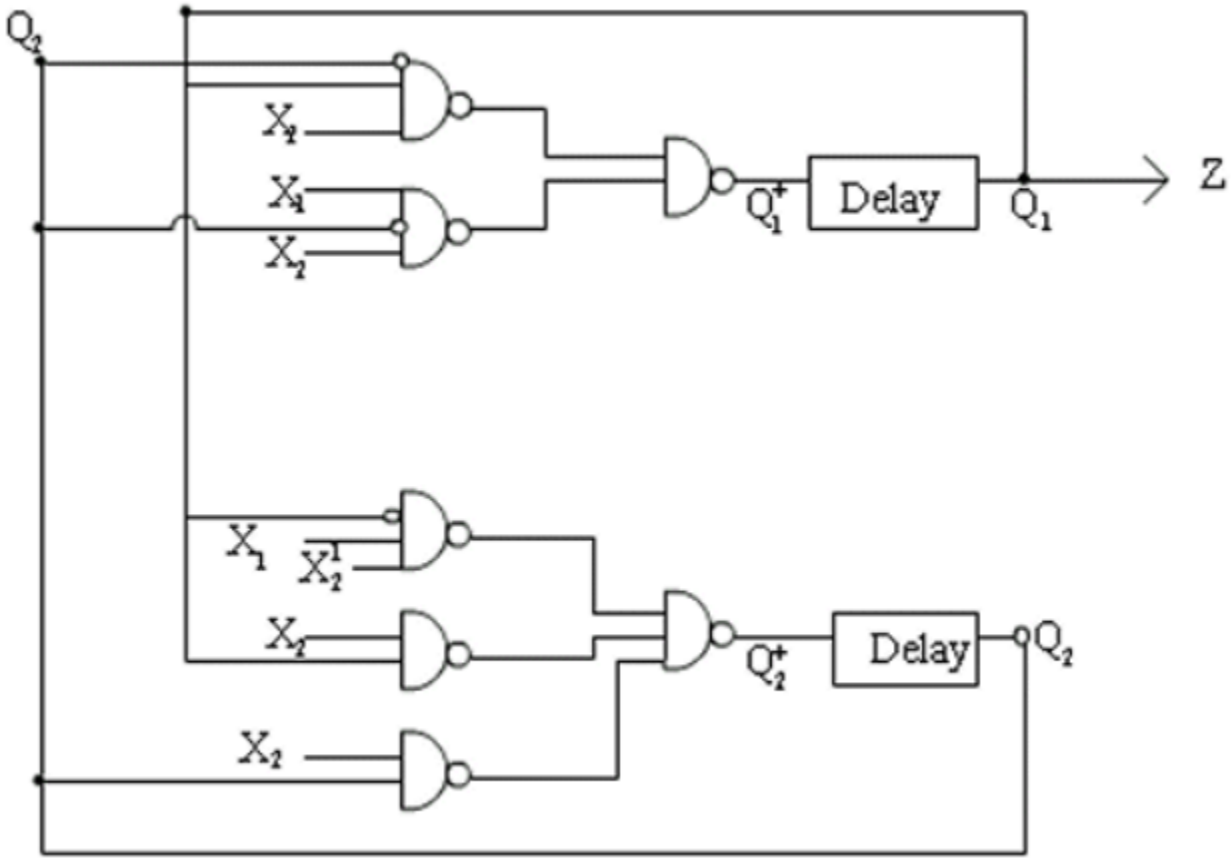

```
module reg8
(
input [7:0] d_i,
input rst_n,
input clk,
output [7:0] q_o
);

reg [7:0] q;

always @(posedge clk or negedge rst_n) begin
    if (rst_n == 1'b0) begin
        q <= 8'b00000000;
    end
    else begin
        q <= d_i;
    end
end

assign q_o = q;

endmodule
```

Yazmaç (Register) – Verilog “parameter” Keyword


```
module reg_param
#(
parameter N=32
)
(
input [N-1:0] d_i,
input rst_n,
input clk,
output [N-1:0] q_o
);

reg [N-1:0] q;

always @(posedge clk or negedge rst_n) begin
    if (rst_n == 1'b0) begin
        q <= {N{1'b0}};
    end
    else begin
        q <= d_i;
    end
end

assign q_o = q;

endmodule
```

Resource	Estimation	Available	Utilization %
LUT	1	20800	0.01
FF	32	41600	0.08
IO	66	106	62.26
BUFG	1	32	3.13

ASENKRON vs SENKRON SIRALI DEVRELER (ASYNCHRONOUS vs SYNCHRONOUS SEQUENTIAL CIRCUITS)



Asenkron devrelerde bellek (memory) bulunmaz ve bir saat sinyali ile bu bellek birimleri senkronize edilmez

Asenkron devrelerde saat olmadığı için kapı seviyesinde gecikmelere dayalı ve geribesleme (feedback) yapıları ile durumlar (states) meydana getirir

Teorik olarak asenkron sıralı devrelerin senkron devrelere göre daha hızlı ve daha az güç tüketen olması beklenir

Fakat asenkron devrelerin zamanlama (timing) analizi senkron devrelere göre çok daha zordur

Günümüzde **akademik alanda** asenkron devreler çalışılmaya devam etmektedir

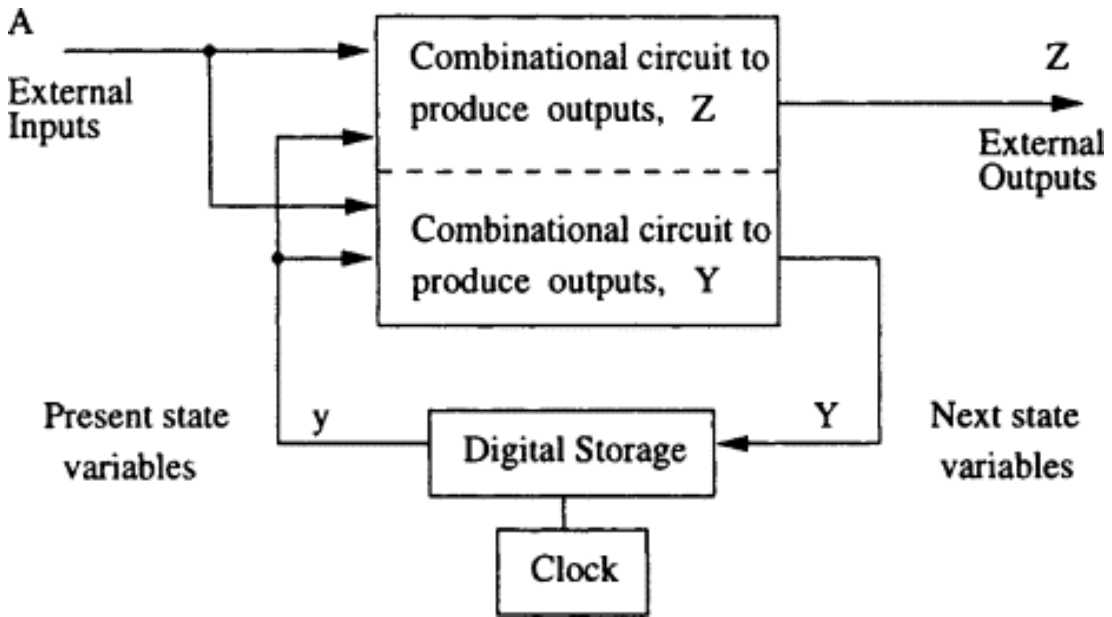
SENKRON SIRALI DEVRELER (SYNCHRONOUS SEQUENTIAL CIRCUITS)

Senkron sıralı devreler birleşik (combinational) ve sıralı (sequential) bloklardan oluşur

Birleşik devre kısmı giriş sinyallerinin değişimine anında tepki veren ve sonuç sinyalleri hesaplanan devrelerdir

Sıralı devre kısmı ise bir saat sinyali (clock signal) ile senkronize olmuş genellikle FF'lardan oluşan ve devrenin o anki durumunu (state) ifade eden bloktur

FF'lar, saat vuruşunun yükselen ya da alçalan kenarında giriş sinyal değerini çıkış sinyaline iletirler



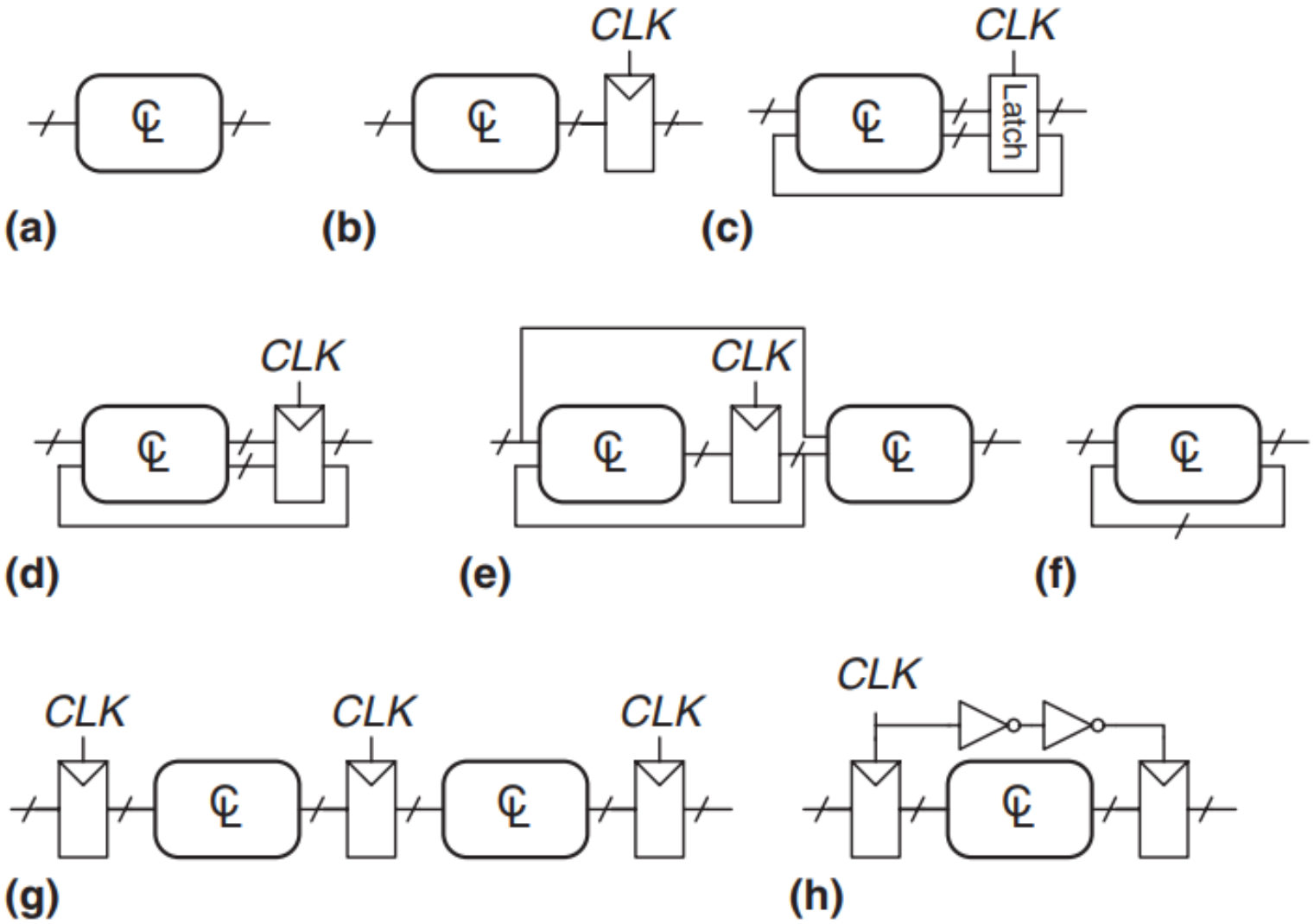
SENKRON SIRALI DEVRELER (SYNCHRONOUS SEQUENTIAL CIRCUITS)

TOBB ETÜ
Ekonomi ve Teknoloji Üniversitesi

The rules of *synchronous sequential circuit composition* teach us that a circuit is a synchronous sequential circuit if it consists of interconnected circuit elements, such that

- Every circuit element is either a register or a combinational circuit
- At least one circuit element is a register
- All registers receive the same clock signal
- Every cyclic path contains at least one register

HANGİLERİ SENKRON SIRALI DEVREDİR?



SONLU DURUM MAKİNELERİ (FINITE STATE MACHINES)

TOBB ETÜ
Ekonomi ve Teknoloji Üniversitesi

Neden “sonlu durum” → Çünkü n adet FF ile 2^n kadar durum meydana getirilebiliyor

Durum tablosu oluşturma → Giriş sinyalleri ve şu anki duruma (present state) göre sonraki durum (next state) ve çıkış sinyallerini hesaplamak

Her bir saat vuruşunda, FF’lar sonraki durum (next state) değerini alırlar ve buna göre de çıkış sinyalleri güncellenir

2 tür sonlu durum makinesi vardır: Moore – Mealy

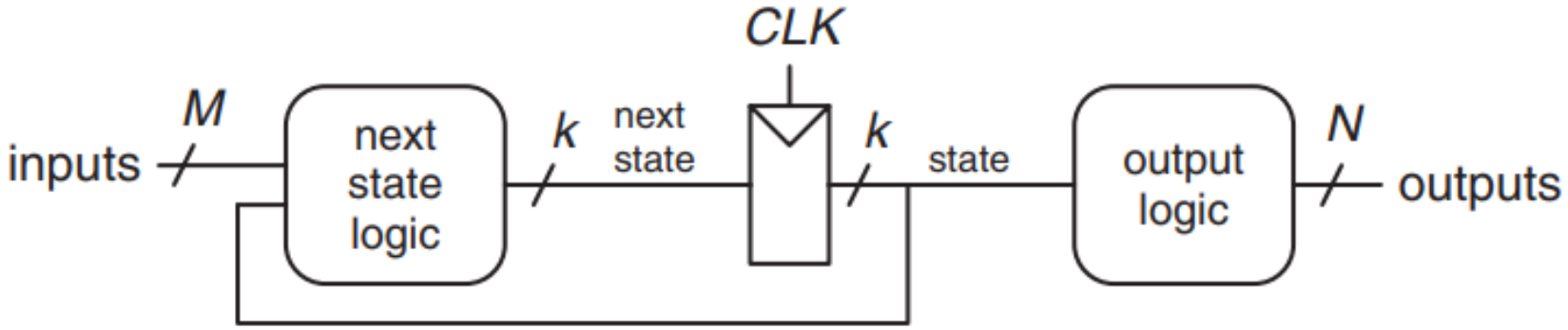
Moore makinelerinde, çıkış sinyalleri sadece devrenin o anki durumuna bağlıdır

Mealy makinelerinde, çıkış sinyalleri hem devrenin o anki durumuna, hem de giriş sinyallerine bağlıdır

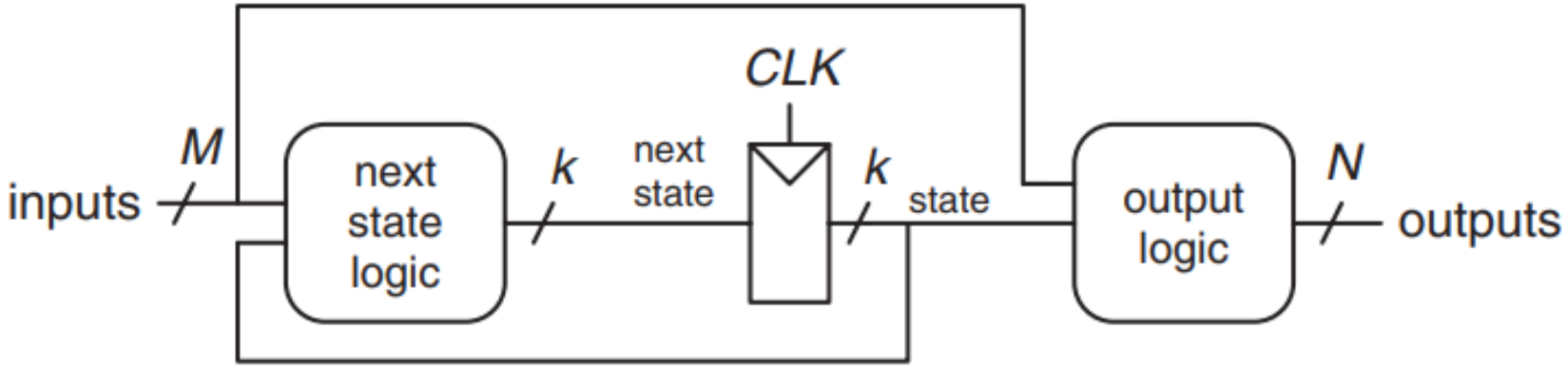
Moore ve Mealy bu yöntemleri 1955/56 yıllarında ortaya atmışlardır

Edward F. Moore (1925-2003) → Bell Lab, University of Wisconsin
George H. Mealy (1927-2010) → Bell Lab, Harvard University

SONLU DURUM MAKİNELERİ



(a)



(b)