# Tahmini Ders İçeriği
# (Tentative Couse Schedule – Syllabus)

**1. Hafta:** Sayı sistemleri, onluk/ikilik taban sayı gösterimleri, mantıksal kapılar, computer system overview, başarım (performance)

**2. Hafta:** 2'lik tabanda işaretli sayılar, mikroişlemci tarihi, benchmarking, başarım,

**3. Hafta:** Başarım, Amdahl yasası, RISC-V development Environment, Verilog HDL ile Birleşik (Combinational) devreler

**4. Hafta:**, Verilog HDL ile sıralı (sequential) mantıksal devre ve sonlu durum makinası tasarımı, timing analysis

**5. Hafta:** Aritmetik devre tasarımları: Toplama, çıkarma, çarpma, bölme, trigonometri, square-root, hyperbolic, exponential, logarithm

**6. Hafta:** Fixed ve Floating-Point sayı gösterimleri

**7. Hafta:** RISC-V buyruk kümesi mimarisi (ISA) ve buyrukların tanıtımı

**8. Hafta:** RISC-V buyruk kümesi mimarisi (ISA) ve buyrukların tanıtımı

| | |
|---|---|
| 20 Mart - 27 Haziran 2023 | II.Yarıyıl (Bahar Yarıyılı) |
| 3 - 30 Temmuz 2023 | II.Yarıyıl (Bahar Yarıyılı) Final/Bütünleme Sınavları |

**9. Hafta:** Tek-çevrim işlemci tasarımı (single-cycle CPU)

**10. Hafta:** Çok-çevrim işlemci tasarımı (multi-cycle CPU)

**11. Hafta:** Boruhatlı işlemci tasarımı (pipelined CPU)

**12. Hafta:** Bellek sistemi ve hiyerarşisi

**13. Hafta:** İleri mimari konuları: Branch prediction, superscalar cpu, out-of-order execution, multi-core systems

**14. Hafta:** Gömülü sistemler, mikrodenetleyiciler, SoCs

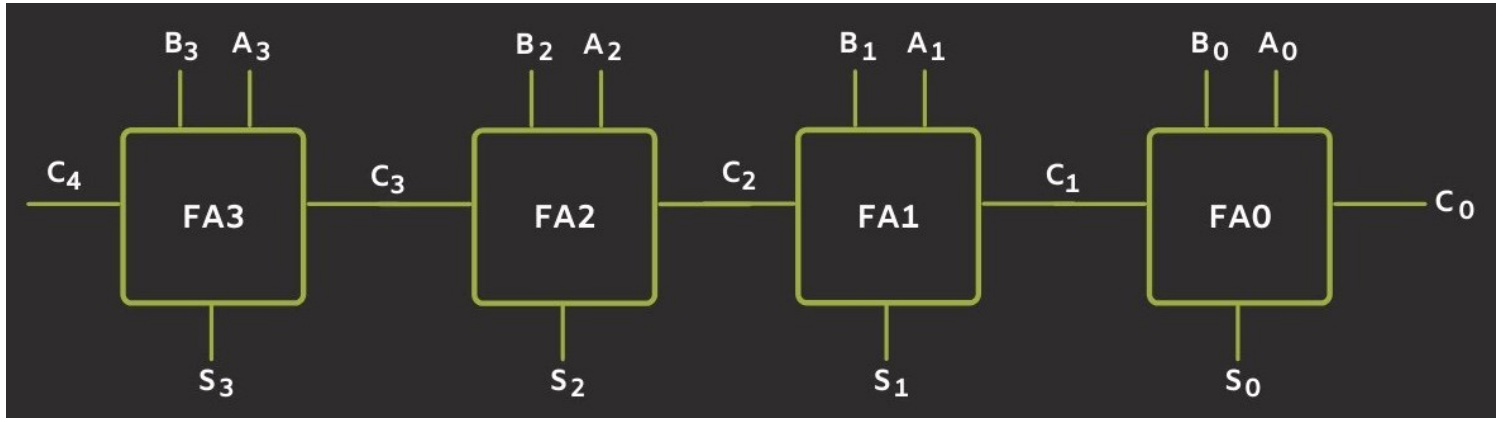
# TOPLAMA ve ÇIKARMA

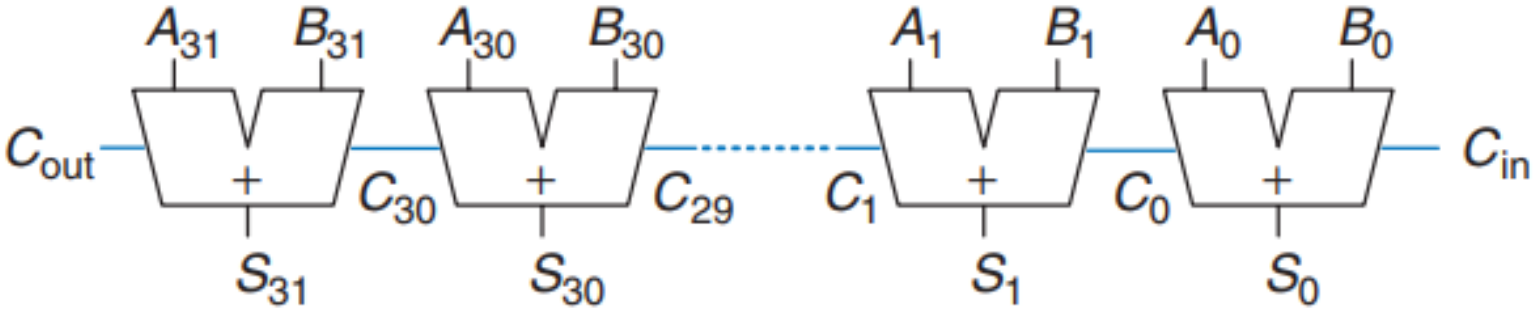
$$
\begin{array}{rl}
 & 00000000\ 00000000\ 00000000\ 00000111_{two} = 7_{ten} \\
+ & 00000000\ 00000000\ 00000000\ 00000110_{two} = 6_{ten} \\
\hline
= & 00000000\ 00000000\ 00000000\ 00001101_{two} = 13_{ten}
\end{array}
$$

$$
\begin{array}{rl}
 & 00000000\ 00000000\ 00000000\ 00000111_{two} = 7_{ten} \\
- & 00000000\ 00000000\ 00000000\ 00000110_{two} = 6_{ten} \\
\hline
= & 00000000\ 00000000\ 00000000\ 00000001_{two} = 1_{ten}
\end{array}
$$

$$
\begin{array}{rl}
 & 00000000\ 00000000\ 00000000\ 00000111_{two} = 7_{ten} \\
+ & 11111111\ 11111111\ 11111111\ 11111010_{two} = -6_{ten} \\
\hline
= & 00000000\ 00000000\ 00000000\ 00000001_{two} = 1_{ten}
\end{array}
$$

# RIPPLE-CARRY ADDER



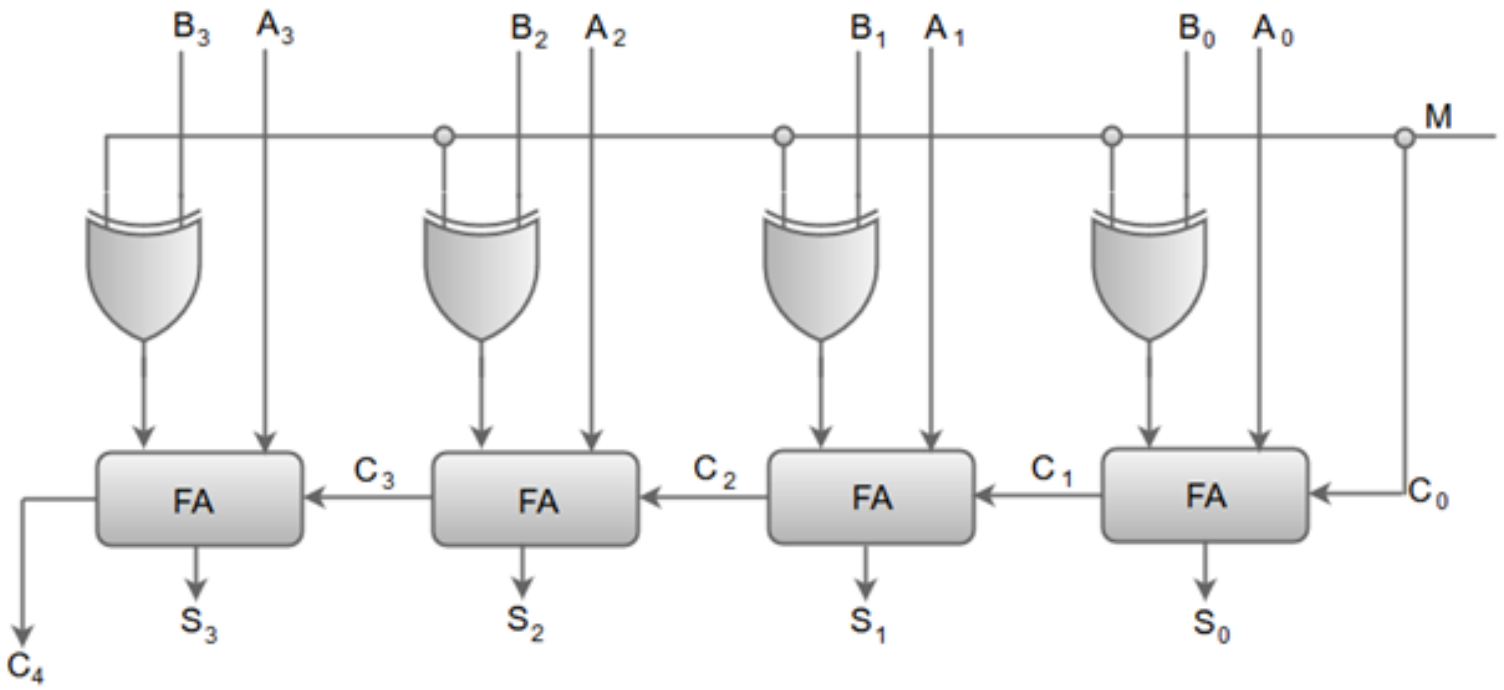




```
module ripple_carry_adder
#(parameter N = 4)
(
input [N-1:0] a_i,
input [N-1:0] b_i,
input cin_i,
output [N-1:0] s_o,
output cout_o
);

wire [N:0] carry;

genvar i;
generate
for (i=0; i<N; i=i+1) begin
    full_adder full_adder_inst
    (
        .a_i(a_i[i]),
        .b_i(b_i[i]),
        .cin_i(carry[i]),
        .s_o(s_o[i]),
        .cout_o(carry[i+1])
    );
end
endgenerate

assign carry[0] = cin_i;
assign cout_o   = carry[N];

endmodule
```

# ADDER/SUBTRACTOR CIRCUIT

2’ye tümleyen alma:
Sayının 1’e tümleyenini al (tüm bitleri tersle, 0’sa 1, 1’se 0 yap)
1 ekle

**4 bit adder-subtractor:**




```
module add_sub
#(parameter N = 32)
(
input [N-1:0] a_i,
input [N-1:0] b_i,
input op_i,              // add if 0, sub if 1
output [N-1:0] s_o,
output cout_o
);

wire [N:0] carry;
wire [N-1:0] b_reg;

genvar i;
generate
for (i=0; i<N; i=i+1) begin
    assign b_reg[i] = b_i[i] ^ op_i;
    full_adder full_adder_inst
    (
        .a_i(a_i[i]),
        .b_i(b_reg[i]),
        .cin_i(carry[i]),
        .s_o(s_o[i]),
        .cout_o(carry[i+1])
    );
end
endgenerate

assign carry[0] = op_i;
assign cout_o   = carry[N];

endmodule
```

# CARRY-LOOKAHEAD ADDER

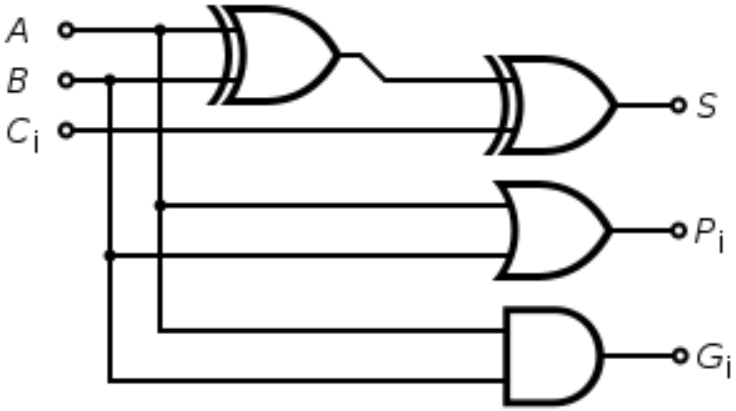
| $A$ | $B$ | $C_i$ | $C_o$ | Type of Carry |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | None |
| 0 | 0 | 1 | 0 | None |
| 0 | 1 | 0 | 0 | None |
| 0 | 1 | 1 | 1 | Propagate |
| 1 | 0 | 0 | 0 | None |
| 1 | 0 | 1 | 1 | Propagate |
| 1 | 1 | 0 | 1 | Generate |
| 1 | 1 | 1 | 1 | Generate/Propagate |

Cin değerinden bağımsız olarak 2 adet sinyal tanımla:

1) Propagate (yaymak, çoğaltmak, propaganda): A veya B'den herhangi biri, ya da ikisi de '1' ise, gelecek olan Cin değerini Cout'a propagate ediyor, yani Cin '1' ise Cout da '1' oluyor

2) Generate (oluşturmak): A ve B '1' ise, Cin değerinden bağımsız olarak Cout '1' olur, Cout oluşturulmuş olunur.

Soru: Ripple-Carry toplayıcıda her bir basamaktaki S ve Cout çıkış sinyalleri bir önceki aşamanın Cout çıkış sinyaline bağlıydı. Carry-Lookahead'de, her bir aşamayı bir önceki aşamadan bağımsız hale getirebilir miyiz?




$$G_i = A_i \,\&\, B_i$$

$$P_i = A_i \wedge B_i = A_i \mid B_i$$

$$S_i = P_i \wedge C_i = (A_i \wedge B_i) \wedge C_i$$

$$C_{i+1} = G_i + P_i C_i$$

Nasıl eşit olur? Çünkü $C_{i+1}$ hesabında $G_i$ parametresi 1 olur

# CARRY-LOOKAHEAD ADDER

Cin değerinden bağımsız olarak 2 adet sinyal tanımla:

1) Propagate (yaymak, çoğaltmak, propaganda): A veya B'den herhangi biri, ya da ikisi de '1' ise, gelecek olan Cin değerini Cout'a propagate ediyor, yani Cin '1' ise Cout da '1' oluyor

2) Generate (oluşturmak): A ve B '1' ise, Cin değerinden bağımsız olarak Cout '1' olur, Cout oluşturulmuş olunur.

Soru: Ripple-Carry toplayıcıda her bir basamaktaki S ve Cout çıkış sinyalleri bir önceki aşamanın Cout çıkış sinyaline bağlıydı. Carry-Lookahead'de, her bir aşamayı bir önceki alamadan bağımsız hale getirebilir miyiz?

$$G_i = A_i \,\&\, B_i$$

$$P_i = A_i \,\hat{}\, B_i = A_i \mid B_i$$

$$S_i = P_i \,\hat{}\, C_i = (A_i \,\hat{}\, B_i) \,\hat{}\, C_i$$

$$C_{i+1} = G_i + P_i C_i$$

$$C_1 = G_0 + P_0 \cdot C_0 \qquad C_1 = G_0 + P_0 \cdot C_0$$

$$C_2 = G_1 + P_1 \cdot C_1 \qquad C_2 = G_1 + G_0 \cdot P_1 + C_0 \cdot P_0 \cdot P_1$$

$$C_3 = G_2 + P_2 \cdot C_2 \qquad C_3 = G_2 + G_1 \cdot P_2 + G_0 \cdot P_1 \cdot P_2 + C_0 \cdot P_0 \cdot P_1 \cdot P_2$$

$$C_4 = G_3 + P_3 \cdot C_3 \qquad C_4 = G_3 + G_2 \cdot P_3 + G_1 \cdot P_2 \cdot P_3 + G_0 \cdot P_1 \cdot P_2 \cdot P_3 + C_0 \cdot P_0 \cdot P_1 \cdot P_2 \cdot P_3$$

# CARRY-LOOKAHEAD ADDER

$G_i = A_i \,\&\, B_i$

$P_i = A_i \,\wedge\, B_i = A_i \mid B_i$

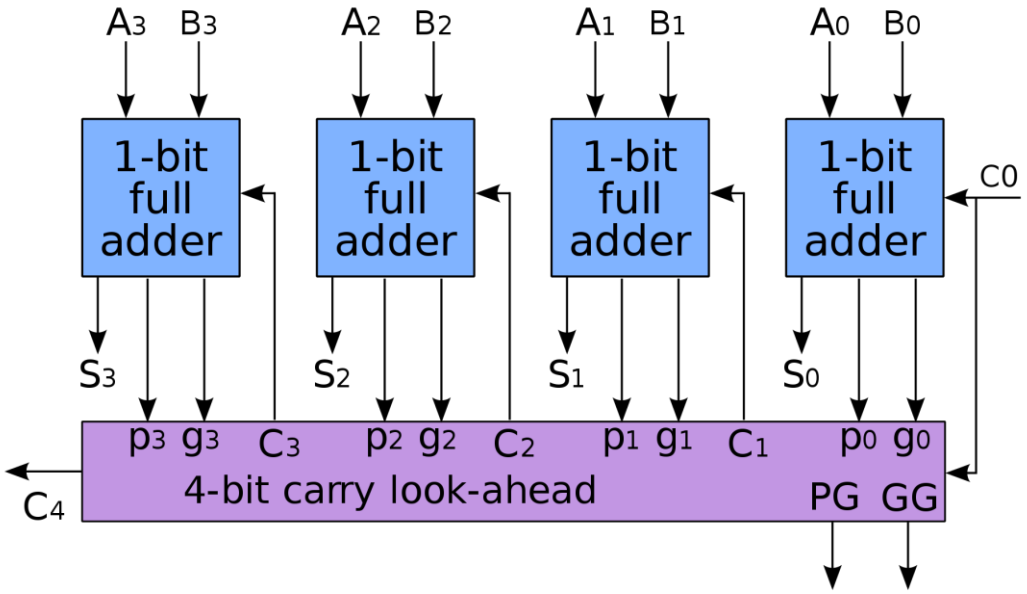
$S_i = P_i \,\wedge\, C_i = (A_i \,\wedge\, B_i) \,\wedge\, C_i$

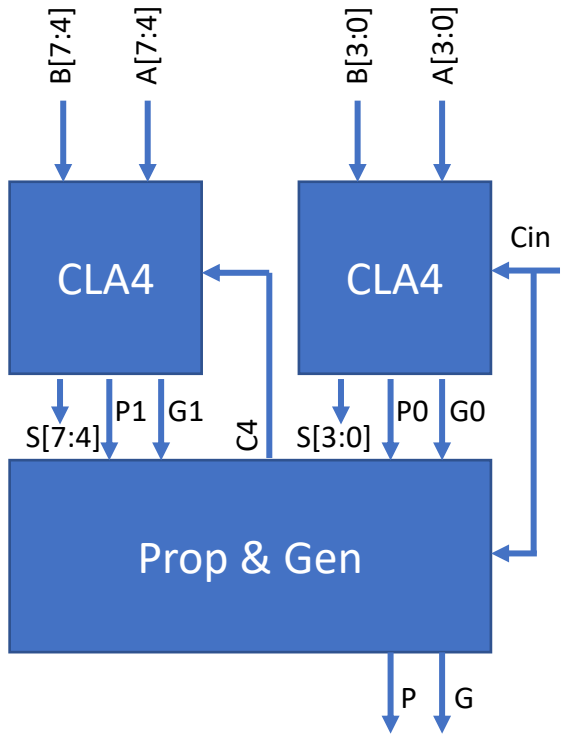
$C_{i+1} = G_i + P_i C_i$

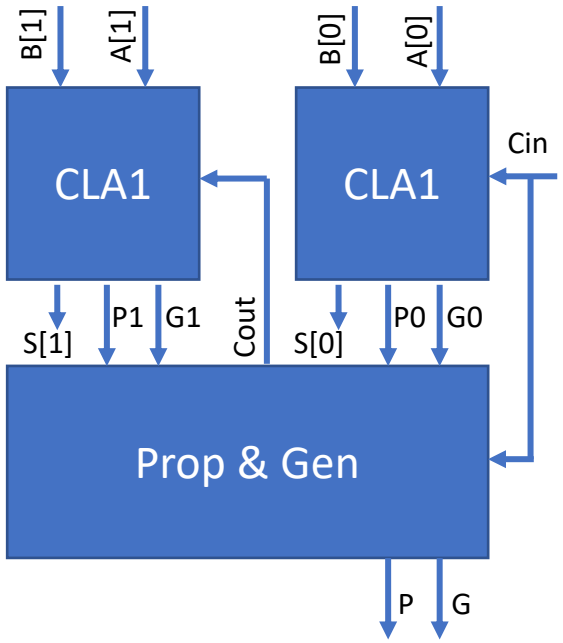
$$C_1 = G_0 + P_0 \cdot C_0 \qquad C_1 = G_0 + P_0 \cdot C_0$$
$$C_2 = G_1 + P_1 \cdot C_1 \qquad C_2 = G_1 + G_0 \cdot P_1 + C_0 \cdot P_0 \cdot P_1$$
$$C_3 = G_2 + P_2 \cdot C_2 \qquad C_3 = G_2 + G_1 \cdot P_2 + G_0 \cdot P_1 \cdot P_2 + C_0 \cdot P_0 \cdot P_1 \cdot P_2$$
$$C_4 = G_3 + P_3 \cdot C_3 \qquad C_4 = G_3 + G_2 \cdot P_3 + G_1 \cdot P_2 \cdot P_3 + G_0 \cdot P_1 \cdot P_2 \cdot P_3 + C_0 \cdot P_0 \cdot P_1 \cdot P_2 \cdot P_3$$

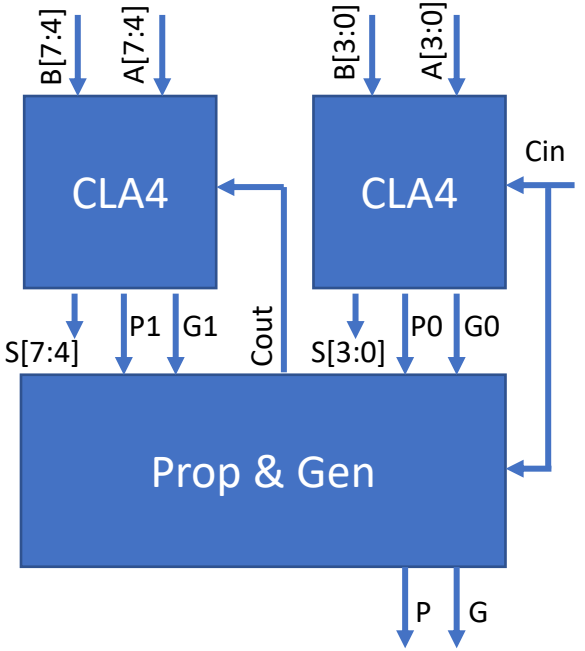
$$PG = P_0 \cdot P_1 \cdot P_2 \cdot P_3$$
$$GG = G_3 + G_2 \cdot P_3 + G_1 \cdot P_3 \cdot P_2 + G_0 \cdot P_3 \cdot P_2 \cdot P_1$$
$$CG = GG + PG \cdot C_{in}$$

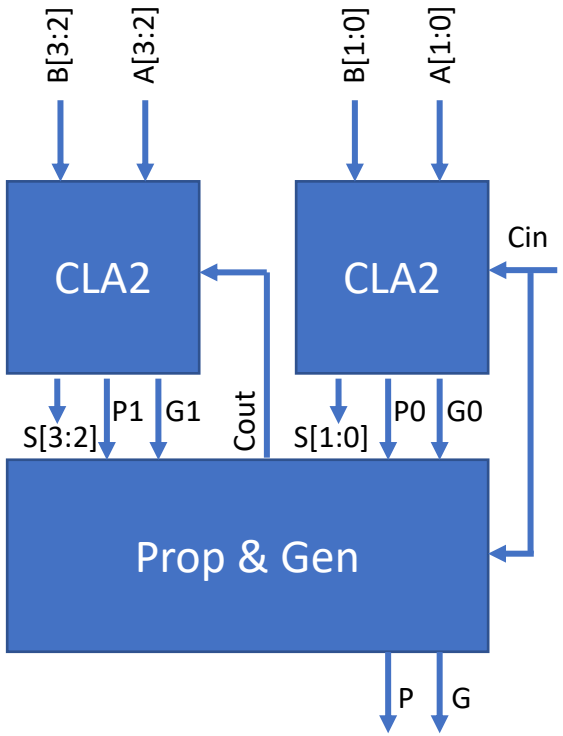
PG: Propagate group
GG: Generate group
CG: Carry group → Same as C4

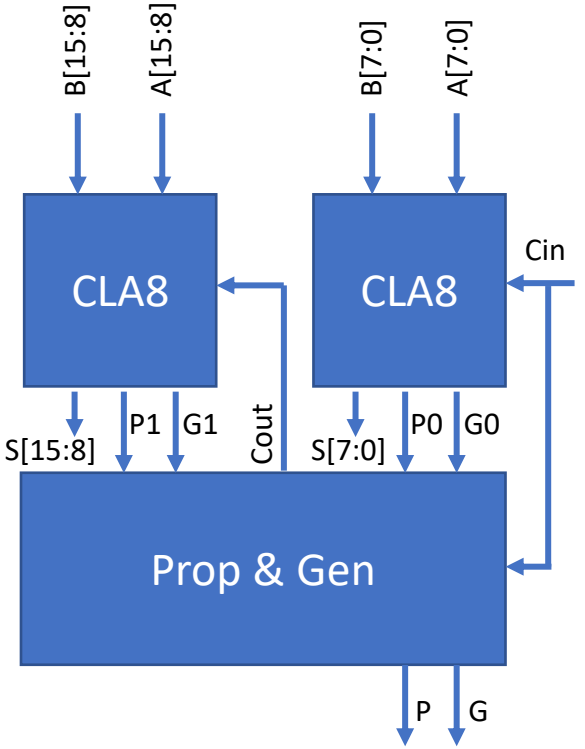


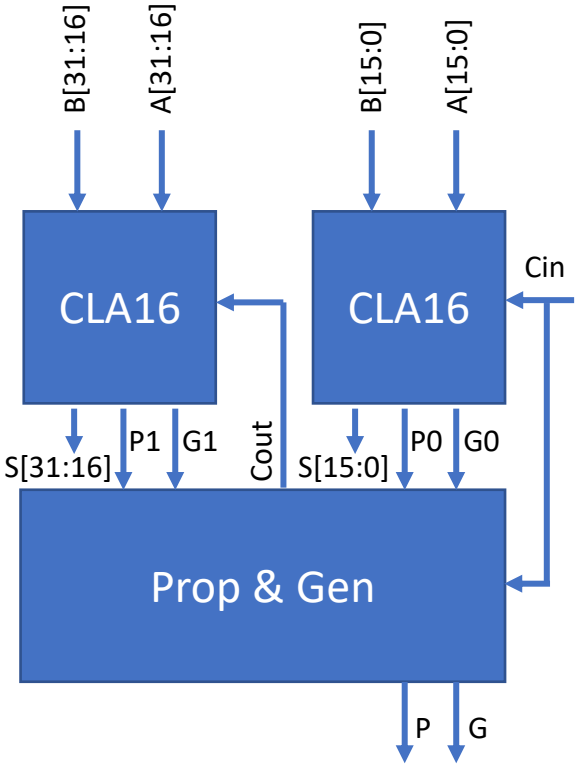




B[1]
A[1]
B[0]
A[0]
CLA1
CLA1
Cin
S[1]
P1
G1
Cout
S[0]
P0
G0
Prop & Gen
P
G


B[7:4]
A[7:4]
B[3:0]
A[3:0]
CLA4
CLA4
Cin
S[7:4]
P1
G1
Cout
S[3:0]
P0
G0
Prop & Gen
P
G


B[3:2]
A[3:2]
B[1:0]
A[1:0]
CLA2
CLA2
Cin
S[3:2]
P1
G1
Cout
S[1:0]
P0
G0
Prop & Gen
P
G


B[15:8]
A[15:8]
B[7:0]
A[7:0]
CLA8
CLA8
Cin
S[15:8]
P1
G1
Cout
S[7:0]
P0
G0
Prop & Gen
P
G

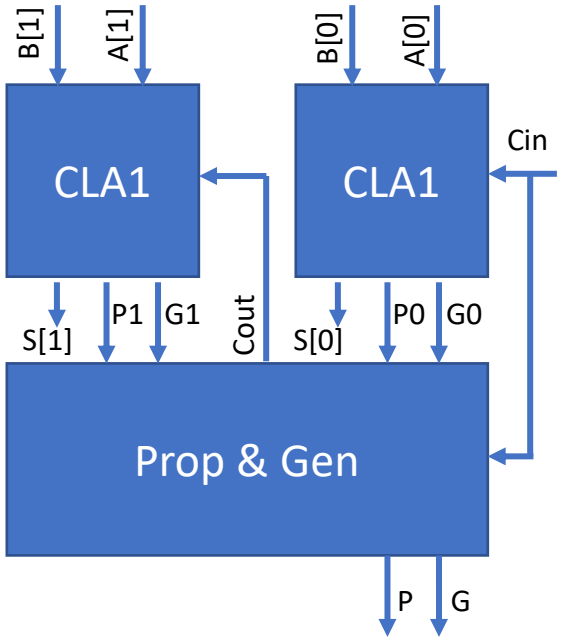

B[31:16]
A[31:16]
B[15:0]
A[15:0]
CLA16
CLA16
Cin
S[31:16]
P1
G1
Cout
S[15:0]
P0
G0
Prop & Gen
P
G

# CARRY-LOOKAHEAD ADDER

$$G_i = A_i \,\&\, B_i$$
$$P_i = A_i \,\hat{}\, B_i = A_i \mid B_i$$
$$S_i = P_i \,\hat{}\, C_i = (A_i \,\hat{}\, B_i) \,\hat{}\, C_i$$
$$C_{i+1} = G_i + P_i C_i$$

$$PG = P_0 \cdot P_1 \cdot P_2 \cdot P_3$$
$$GG = G_3 + G_2 \cdot P_3 + G_1 \cdot P_3 \cdot P_2 + G_0 \cdot P_3 \cdot P_2 \cdot P_1$$

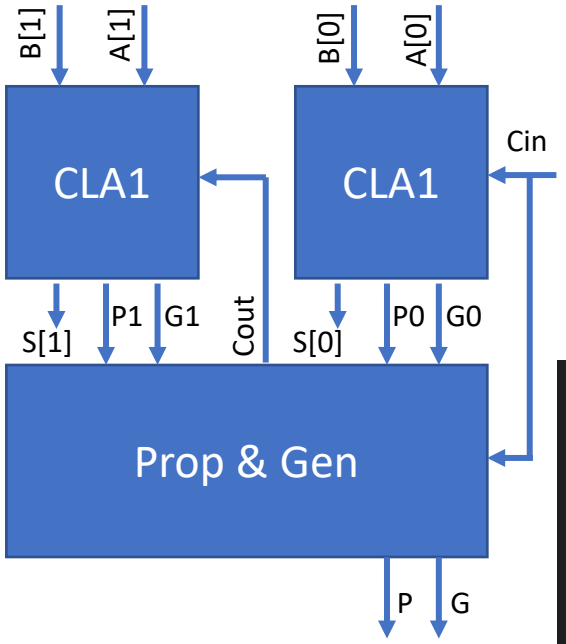
$$PG = P_1 \,\&\, P_0$$
$$GG = G_1 \mid G_0 \,\&\, P_1$$
$$Cout = G_0 \mid P_0 \,\&\, Cin$$




```
module cla1
(
input a_i,
input b_i,
input cin_i,
output p_o,
output g_o,
output s_o
);

assign p_o = a_i ^ b_i;
assign g_o = a_i & b_i;
assign s_o = a_i ^ b_i ^ cin_i;

endmodule
```

```
module prop_gen
(
input cin_i,
input [1:0] p_i,
input [1:0] g_i,
output cout_o,
output p_o,
output g_o
);

assign cout_o = g_i[0] | p_i[0] & cin_i;
assign p_o = &p_i;
assign g_o = g_i[1] | g_i[0] & p_i[1];

endmodule
```

# CARRY-LOOKAHEAD ADDER

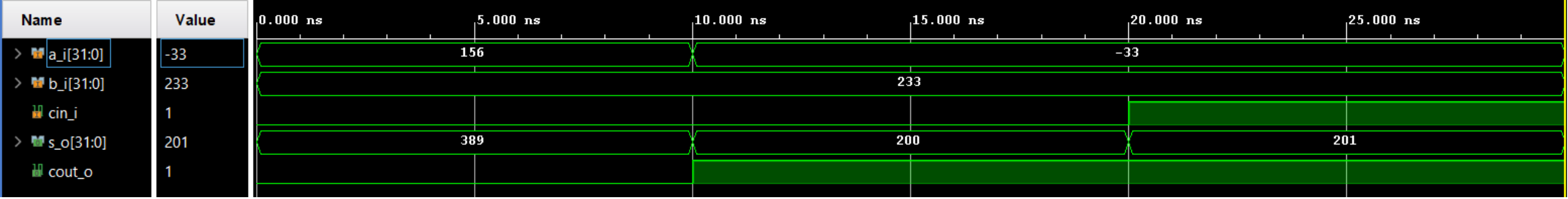



```
module cla2
(
input [1:0] a_i,
input [1:0] b_i,
input cin_i,
output p_o,
output g_o,
output [1:0] s_o
);

wire [1:0] p,g;
wire cout;

cla1 cla1_1
(
.a_i    (a_i[0]),
.b_i    (b_i[0]),
.cin_i  (cin_i),
.p_o    (p[0]),
.g_o    (g[0]),
.s_o    (s_o[0])
);
```

```
cla1 cla1_2
(
.a_i    (a_i[1]),
.b_i    (b_i[1]),
.cin_i  (cout),
.p_o    (p[1]),
.g_o    (g[1]),
.s_o    (s_o[1])
);

prop_gen pg
(
.cin_i  (cin_i),
.p_i    (p),
.g_i    (g),
.cout_o (cout),
.p_o    (p_o),
.g_o    (g_o)
);

endmodule
```

```
module cla32
(
input [31:0] a_i,
input [31:0] b_i,
input cin_i,
output cout_o,
output [31:0] s_o
);

wire [1:0] p,g;
wire cout;

assign cout_o = g[1] | p[1] & cout;

cla16 cla16_1
(
.a_i    (a_i[15:0]),
.b_i    (b_i[15:0]),
.cin_i  (cin_i),
.p_o    (p[0]),
.g_o    (g[0]),
.s_o    (s_o[15:0])
);
```

```
cla16 cla16_2
(
.a_i    (a_i[31:16]),
.b_i    (b_i[31:16]),
.cin_i  (cout),
.p_o    (p[1]),
.g_o    (g[1]),
.s_o    (s_o[31:16])
);

prop_gen pg
(
.cin_i  (cin_i),
.p_i    (p),
.g_i    (g),
.cout_o (cout),
.p_o    (),
.g_o    ()
);

endmodule
```




```
module tb_ripple_carry_adder;

reg [31:0] a_i;
reg [31:0] b_i;
reg cin_i;
wire cout_o;
wire [31:0] s_o;

ripple_carry_adder
#(.N(32))
DUT
(
.a_i    (a_i   ),
.b_i    (b_i   ),
.cin_i  (cin_i ),
.cout_o (cout_o),
.s_o    (s_o   )
);
```

```
module tb_cla32;

reg [31:0] a_i;
reg [31:0] b_i;
reg cin_i;
wire cout_o;
wire [31:0] s_o;

cla32 DUT
(
.a_i    (a_i   ),
.b_i    (b_i   ),
.cin_i  (cin_i ),
.cout_o (cout_o),
.s_o    (s_o   )
);
```

```
initial begin
a_i = 156;
b_i = 233;
cin_i = 0;
#10;

a_i = -33;
b_i = 233;
cin_i = 0;
#10;

a_i = -33;
b_i = 233;
cin_i = 1;
#10;


$finish;
end

endmodule
```

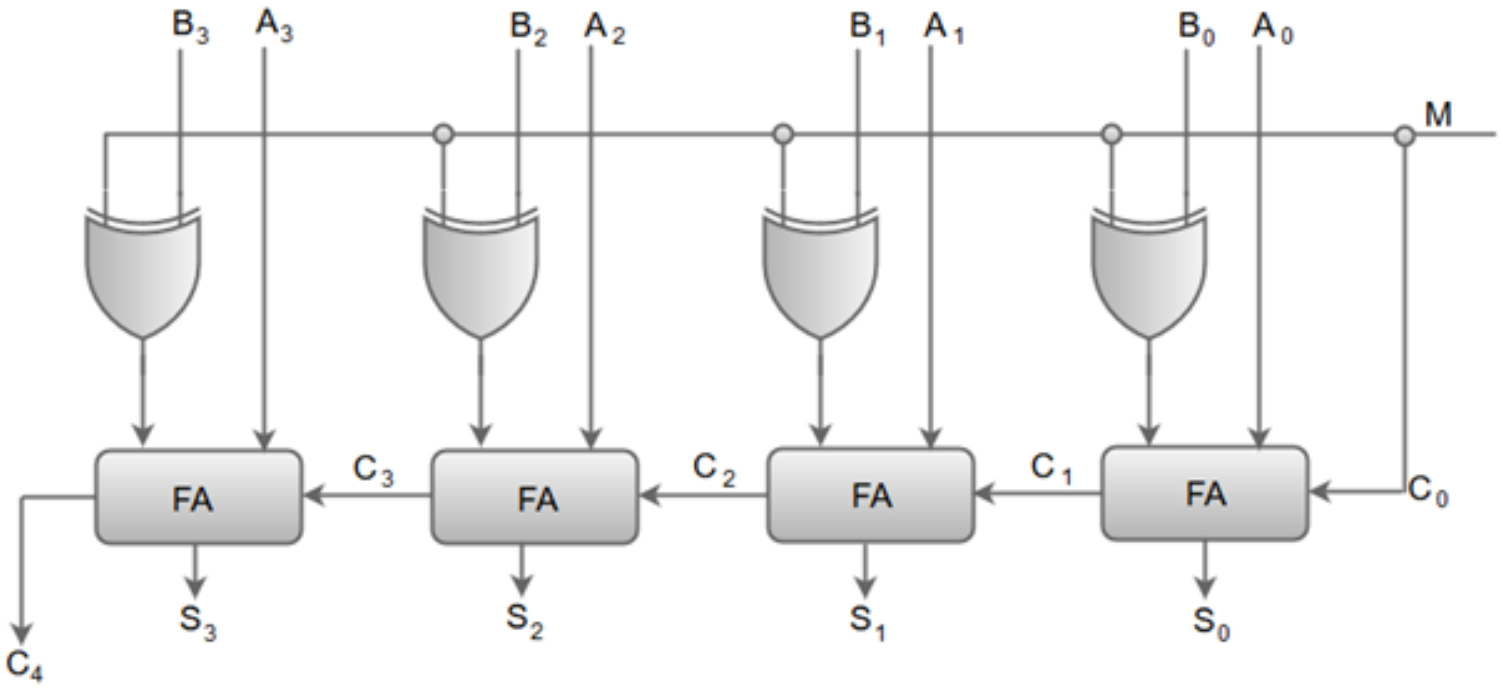
YONGATEK Microelectronics

ERCİYES ÜNİVERSİTESİ 1978

# Verilog Headers

```
`ifndef DEFINES_SV
`define DEFINES_SV

`define USE_CARRY_LOOK_AHEAD

`endif
```

**4 bit adder-subtractor:**




```
module add_sub
#(parameter N = 32)
(
input [N-1:0] a_i,
input [N-1:0] b_i,
input op_i,              // add if 0, sub if 1
output [N-1:0] s_o,
output cout_o
);

wire [N:0] carry;
wire [N-1:0] b_reg;

genvar i;
generate
for (i=0; i<N; i=i+1) begin
    assign b_reg[i] = b_i[i] ^ op_i;
    full_adder full_adder_inst
    (
        .a_i(a_i[i]),
        .b_i(b_reg[i]),
        .cin_i(carry[i]),
        .s_o(s_o[i]),
        .cout_o(carry[i+1])
    );
end
endgenerate

assign carry[0] = op_i;
assign cout_o   = carry[N];

endmodule
```

# Verilog IFDEF/IFNDEF Compiler Directives

```
`include "defines.vh"

module add_sub
#(parameter N = 32)
(
input [N-1:0] a_i,
input [N-1:0] b_i,
input op_i,              // add if 0, sub if 1
output [N-1:0] s_o,
output cout_o
);

wire [N-1:0] b_reg;
genvar i;

generate
    for (i=0; i<N; i=i+1) begin
        assign b_reg[i] = b_i[i] ^ op_i;
    end
endgenerate
```

```
`ifndef USE_CARRY_LOOK_AHEAD

wire [N:0] carry;

generate
for (i=0; i<N; i=i+1) begin
    full_adder full_adder_inst
    (
        .a_i(a_i[i]),
        .b_i(b_reg[i]),
        .cin_i(carry[i]),
        .s_o(s_o[i]),
        .cout_o(carry[i+1])
    );
end
endgenerate

assign carry[0] = op_i;
assign cout_o   = carry[N];

`else
```

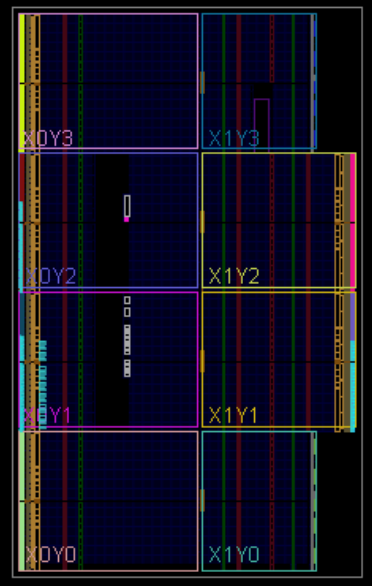
```
`else

cla32 cla32_inst
(
.a_i    (a_i),
.b_i    (b_reg),
.cin_i  (op_i),
.cout_o (cout_o),
.s_o    (s_o)
);

`endif

endmodule
```

SIGN - xc7a100tcsg324-1

ripple_carry_adder
Nets (211)
Leaf Cells (146)

| Name | Slice LUTs (63400) | Slice (15850) | LUT as Logic (63400) | Bonded IOB (210) |
|---|---|---|---|---|
| ripple_carry_adder | 32 | 16 | 32 | 98 |

Project Summary | Device




Source File Properties
ripple_carry_adder.v
General | Properties

Tcl Console | Messages | Log | Reports | Design Runs | DRC | Power | Timing

Unconstrained Paths - NONE - NONE - Setup

- Check Timing (0)
- Intra-Clock Paths
- Inter-Clock Paths
- Other Path Groups
- User Ignored Paths
- Unconstrained Paths
  - NONE to NONE
    - Setup (10)
    - Hold (10)

| Name | Slack | Levels | High Fanout | From | To | Total Delay | Logic Delay | Net Delay | Requirement | Source Clock | Destination Clock | Exception | Clo |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Path 1 | ∞ | 18 | 3 | a_i[0] | cout_o | 28.045 | 8.998 | 19.047 | ∞ | input port clock | | | |
| Path 2 | ∞ | 18 | 3 | a_i[0] | s_o[31] | 27.650 | 8.749 | 18.901 | ∞ | input port clock | | | |
| Path 3 | ∞ | 18 | 3 | a_i[0] | s_o[30] | 26.967 | 8.753 | 18.215 | ∞ | input port clock | | | |
| Path 4 | ∞ | 17 | 3 | a_i[0] | s_o[29] | 26.266 | 8.385 | 17.881 | ∞ | input port clock | | | |
| Path 5 | ∞ | 17 | 3 | a_i[0] | s_o[28] | 25.581 | 8.386 | 17.195 | ∞ | input port clock | | | |
| Path 6 | ∞ | 16 | 3 | a_i[0] | s_o[27] | 24.891 | 8.029 | 16.861 | ∞ | input port clock | | | |
| Path 7 | ∞ | 16 | 3 | a_i[0] | s_o[26] | 24.441 | 8.036 | 16.405 | ∞ | input port clock | | | |

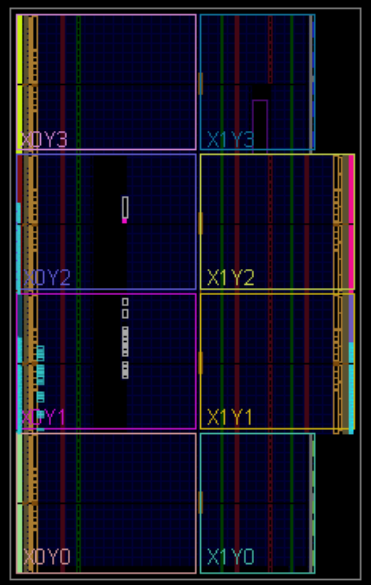
Timing Summary - impl_1 (saved) | Timing Summary - timing_1

SIGN - xc7a100tcsg324-1

N cla32
- Nets (254)
- Leaf Cells (189)

| Name | Slice LUTs (63400) | Slice (15850) | LUT as Logic (63400) | Bonded IOB (210) |
|---|---|---|---|---|
| N cla32 | 61 | 20 | 61 | 98 |

Project Summary | Device




**Source File Properties**

cla32.v

Enabled

General | Properties

Tcl Console | Messages | Log | Reports | Design Runs | Power | DRC | Timing

**Unconstrained Paths - NONE - NONE - Setup**

- Check Timing (0)
- Intra-Clock Paths
- Inter-Clock Paths
- Other Path Groups
- User Ignored Paths
- Unconstrained Paths
  - NONE to NONE
    - Setup (10)
    - Hold (10)

| Name | Slack | Levels | High Fanout | From | To | Total Delay | Logic Delay | Net Delay | Requirement | Source Clock | Destination Clock | Exception | Clo |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Path 1 | ∞ | 7 | 8 | b_i[0] | s_o[28] | 14.894 | 4.662 | 10.232 | ∞ | input port clock | | | |
| Path 2 | ∞ | 7 | 8 | b_i[0] | s_o[30] | 14.796 | 4.675 | 10.121 | ∞ | input port clock | | | |
| Path 3 | ∞ | 7 | 5 | b_i[0] | s_o[12] | 14.752 | 5.170 | 9.582 | ∞ | input port clock | | | |
| Path 4 | ∞ | 7 | 8 | b_i[0] | s_o[31] | 14.671 | 4.671 | 9.999 | ∞ | input port clock | | | |
| Path 5 | ∞ | 7 | 5 | b_i[0] | s_o[13] | 14.515 | 4.937 | 9.578 | ∞ | input port clock | | | |
| Path 6 | ∞ | 7 | 5 | b_i[0] | s_o[15] | 14.412 | 4.937 | 9.475 | ∞ | input port clock | | | |
| Path 7 | ∞ | 7 | 8 | b_i[0] | s_o[29] | 14.409 | 4.661 | 9.748 | ∞ | input port clock | | | |

# Farklı Toplayıcı Algoritmaları

* Brent-Kung Adder

* Kogge-Stone Adder

* Ling Adder

* Carry-skip Adder

...

# Çarpma (Multiplication)

| 31 25 | 24 20 | 19 15 | 14 12 | 11 7 | 6 0 | |
|---|---|---|---|---|---|---|
| 0000001 | rs2 | rs1 | 000 | rd | 0110011 | R mul |
| 0000001 | rs2 | rs1 | 001 | rd | 0110011 | R mulh |
| 0000001 | rs2 | rs1 | 010 | rd | 0110011 | R mulhsu |
| 0000001 | rs2 | rs1 | 011 | rd | 0110011 | R mulhu |
| 0000001 | rs2 | rs1 | 100 | rd | 0110011 | R div |
| 0000001 | rs2 | rs1 | 101 | rd | 0110011 | R divu |
| 0000001 | rs2 | rs1 | 110 | rd | 0110011 | R rem |
| 0000001 | rs2 | rs1 | 111 | rd | 0110011 | R remu |

**RV32M**

**mul**tiply

**mul**tiply **h**igh { _ / **u**nsigned / **s**igned **u**nsigned }

{ **div**ide / **rem**ainder } { _ / **u**nsigned }

Çarpma: 32-bit x 32-bit = 64-bit

Sonuç yazmacına alt 32-bit (mul) ya da üst 32-bit (mulh) yazılabilir

İşaretli 2 sayı (mulh), işaretsiz 2 sayı (mulhu) ya da biri işaretli biri işaretsiz 2 sayı (mulhsu) çarpılıyor olabilir

# İşaretsiz Çarpma (Unsigned Multiplication)

Çarpma: 4-bit x 4-bit

İşaretsiz ise sayı1 (0-15) sayı2 (0-15) olabilir ve sonuç en fazla 225 olabilir
8-bit gerekir (ceil(log2(225)) = 8)

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | 1 | 1 | 1 | 0 | (14) |
| | | | × | 1 | 0 | 1 | 0 | (10) |
| | | | | 0 | 0 | 0 | 0 | |
| | | | 1 | 1 | 1 | 0 | | |
| | | 0 | 0 | 0 | 0 | | | |
| + | 1 | 1 | 1 | 0 | | | | |
| 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | (140) |

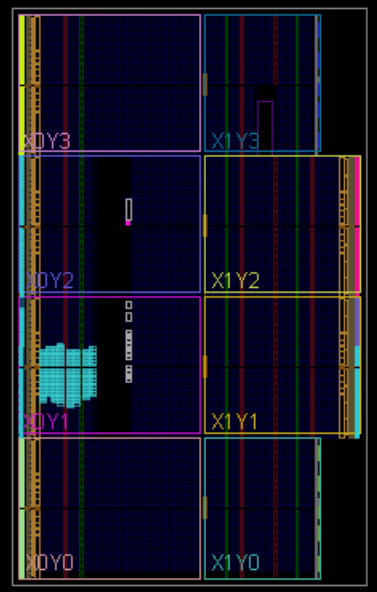
```
module multu
#(parameter N = 32)
(
input [N-1:0] a_i,
input [N-1:0] b_i,
output [2*N-1:0] mul_o
);

integer i;
integer j;
reg [2*N-1:0] temp;
reg [2*N-1:0] pp;

always@* begin
    temp = 0;
    for (i=0; i<N; i=i+1) begin
        pp = 0;
        for (j=0; j<N; j=j+1) begin
            pp[j] = a_i[j] & b_i[i];
        end
        pp = pp << i;
        temp = temp + pp;
    end
end

assign mul_o = temp;

endmodule
```

Nets (2287)

Leaf Cells (1650)

| Name | Slice LUTs (63400) | Slice (15850) | LUT as Logic (63400) | Bonded IOB (210) |
|---|---|---|---|---|
| multu | 1069 | 290 | 1069 | 128 |

Project Summary | Device | tb_multu.v




**Source File Properties**

multu.v

Enabled

General | Properties

Tcl Console | Messages | Log | Reports | Design Runs | Power | DRC | Timing

**Unconstrained Paths - NONE - NONE - Setup**

- Check Timing (0)
- Intra-Clock Paths
- Inter-Clock Paths
- Other Path Groups
- User Ignored Paths
- Unconstrained Paths
  - NONE to NONE
    - Setup (10)
    - Hold (10)

| Name | Slack | Levels | High Fanout | From | To | Total Delay | Logic Delay | Net Delay | Requirement | Source Clock | Destination Clock | Exception |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Path 1 | ∞ | 19 | 63 | b_i[27] | mul_o[61] | 25.161 | 8.266 | 16.895 | ∞ | input port clock | | |
| Path 2 | ∞ | 19 | 63 | b_i[27] | mul_o[60] | 25.066 | 8.167 | 16.899 | ∞ | input port clock | | |
| Path 3 | ∞ | 19 | 63 | b_i[27] | mul_o[63] | 24.952 | 8.237 | 16.715 | ∞ | input port clock | | |
| Path 4 | ∞ | 18 | 63 | b_i[27] | mul_o[57] | 24.877 | 8.176 | 16.701 | ∞ | input port clock | | |
| Path 5 | ∞ | 17 | 63 | b_i[27] | mul_o[52] | 24.798 | 7.932 | 16.866 | ∞ | input port clock | | |
| Path 6 | ∞ | 18 | 63 | b_i[27] | mul_o[59] | 24.795 | 8.145 | 16.651 | ∞ | input port clock | | |
| Path 7 | ∞ | 17 | 63 | b_i[27] | mul_o[53] | 24.735 | 8.039 | 16.695 | ∞ | input port clock | | |

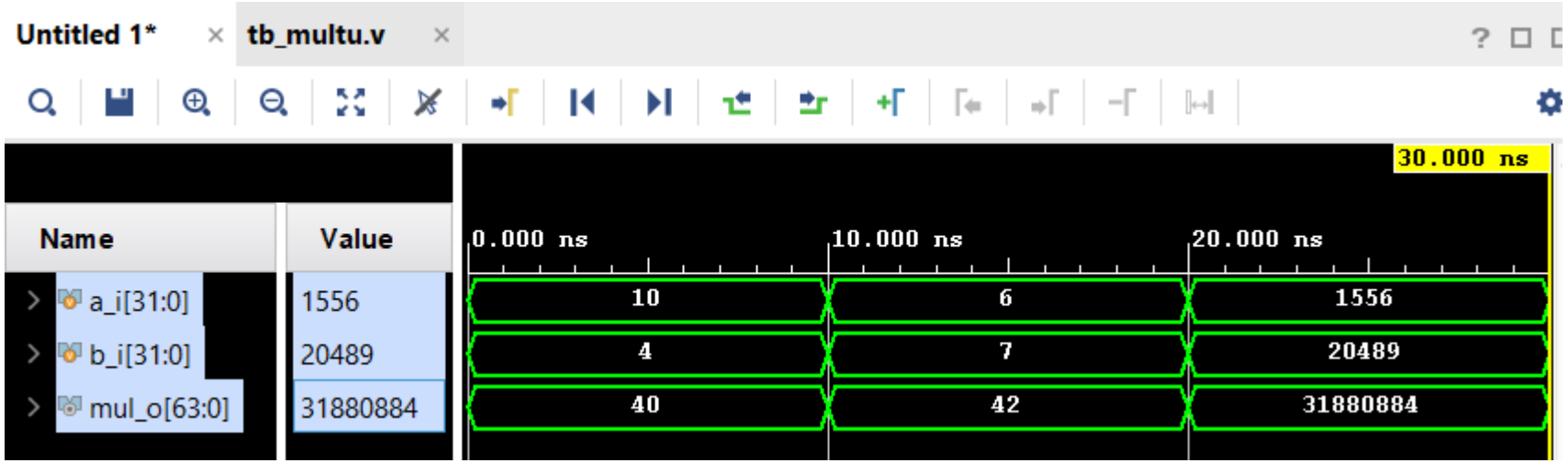
Timing Summary - impl_1 (saved) | **Timing Summary - timing_1**

# İşaretsiz Çarpma (DSP Kaynak Kullanımı)

```
`include "defines.vh"

module multu
#(parameter N = 32)
(
input [N-1:0] a_i,
input [N-1:0] b_i,
output [2*N-1:0] mul_o
);

`ifndef USE_DSP

integer i;
integer j;
reg [2*N-1:0] temp;
reg [2*N-1:0] pp;

always@* begin
    temp = 0;
    for (i=0; i<N; i=i+1) begin
        pp = 0;
        for (j=0; j<N; j=j+1) begin
            pp[j] = a_i[j] & b_i[i];
        end
        pp = pp << i;
        temp = temp + pp;
    end
end

assign mul_o = temp;
```

```
`else

assign mul_o = a_i * b_i;

`endif

endmodule
```

```
`ifndef DEFINES_SV
`define DEFINES_SV

`define USE_CARRY_LOOK_AHEAD
`define USE_DSP

`endif
```



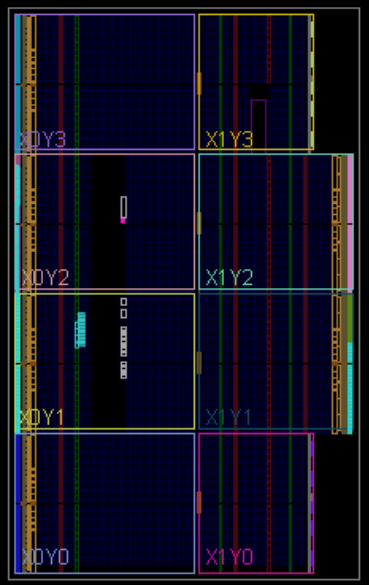
SIGN - xc7a100tcsg324-1



N multu
- Nets (539)
- Leaf Cells (195)

| Name 1 | Slice LUTs (63400) | Slice (15850) | LUT as Logic (63400) | DSPs (240) | Bonded IOB (210) |
|---|---|---|---|---|---|
| N multu | 47 | 12 | 47 | 4 | 128 |

Project Summary | Device




**Source File Properties**

Select an object to see properties

Tcl Console | Messages | Log | Reports | Design Runs | Power | DRC | Methodology | Timing | Utilization

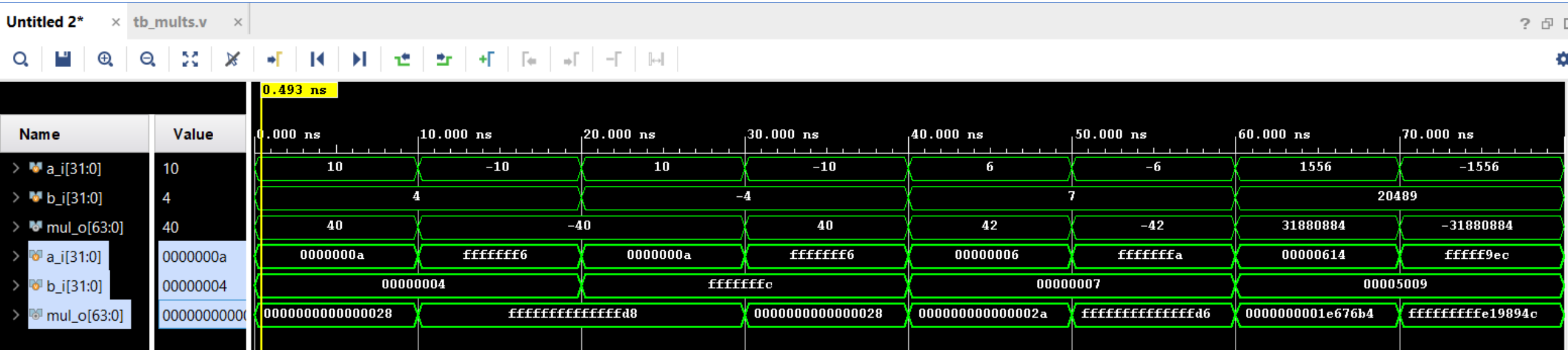
**Unconstrained Paths - NONE - NONE - Setup**

- Check Timing (0)
- Intra-Clock Paths
- Inter-Clock Paths
- Other Path Groups
- User Ignored Paths
- Unconstrained Paths
  - NONE to NONE
    - Setup (10)
    - Hold (10)

| Name | Slack 1 | Levels | High Fanout | From | To | Total Delay | Logic Delay | Net Delay | Requirement | Source Clock | Destination Clock | Exception |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Path 1 | ∞ | 17 | 2 | b_i[1] | mul_o[61] | 21.154 | 11.341 | 9.813 | ∞ | input port clock | | |
| Path 2 | ∞ | 17 | 2 | b_i[1] | mul_o[63] | 21.095 | 11.321 | 9.775 | ∞ | input port clock | | |
| Path 3 | ∞ | 17 | 2 | b_i[1] | mul_o[60] | 21.045 | 11.226 | 9.819 | ∞ | input port clock | | |
| Path 4 | ∞ | 16 | 2 | b_i[1] | mul_o[59] | 21.003 | 11.218 | 9.785 | ∞ | input port clock | | |
| Path 5 | ∞ | 16 | 2 | b_i[1] | mul_o[56] | 20.719 | 11.095 | 9.624 | ∞ | input port clock | | |
| Path 6 | ∞ | 15 | 2 | b_i[1] | mul_o[53] | 20.704 | 11.089 | 9.615 | ∞ | input port clock | | |
| Path 7 | ∞ | 16 | 2 | b_i[1] | mul_o[57] | 20.703 | 11.201 | 9.501 | ∞ | input port clock | | |

Timing Summary - impl_1 (saved) | **Timing Summary - timing_1**

# İşaretli Çarpma (Signed Multiplication)




```
module mults
#(parameter N = 32)
(
input signed [N-1:0] a_i,
input signed [N-1:0] b_i,
output signed [2*N-1:0] mul_o
);

assign mul_o = a_i * b_i;

endmodule
```

```
module tb_mults;

reg signed [31:0] a_i;
reg signed [31:0] b_i;
wire signed [63:0] mul_o;

mults
#(.N(32))
DUT
(
    .a_i    (a_i   ),
    .b_i    (b_i   ),
    .mul_o  (mul_o   )
);
```

```
initial begin
a_i = 10;
b_i = 4;
#10;

a_i = -10;
b_i = 4;
#10;

a_i = 10;
b_i = -4;
#10;

a_i = -10;
b_i = -4;
#10;
```

```
a_i = 6;
b_i = 7;
#10;

a_i = -6;
b_i = 7;
#10;
```

```
a_i = 1556;
b_i = 20489;
#10;

a_i = -1556;
b_i = 20489;
#10;

$finish;

end
```

# Farklı Çarpıcı Algoritmaları

* Wallace Tree Multiplier

* Dadda Multiplier

* Booth’s Multiplication Algorithm

…

# Bölme (Division)

| 31 25 | 24 20 | 19 15 | 14 12 | 11 7 | 6 0 | |
|---|---|---|---|---|---|---|
| 0000001 | rs2 | rs1 | 000 | rd | 0110011 | R mul |
| 0000001 | rs2 | rs1 | 001 | rd | 0110011 | R mulh |
| 0000001 | rs2 | rs1 | 010 | rd | 0110011 | R mulhsu |
| 0000001 | rs2 | rs1 | 011 | rd | 0110011 | R mulhu |
| 0000001 | rs2 | rs1 | 100 | rd | 0110011 | R div |
| 0000001 | rs2 | rs1 | 101 | rd | 0110011 | R divu |
| 0000001 | rs2 | rs1 | 110 | rd | 0110011 | R rem |
| 0000001 | rs2 | rs1 | 111 | rd | 0110011 | R remu |

**RV32M**

**mul**tiply

**mul**tiply **h**igh { _ / **u**nsigned / **s**igned **u**nsigned }

{ **div**ide / **rem**ainder } { _ / **u**nsigned }

* Goldschmidt

* Newton-Raphson

* Restoring, non-restoring

...

# Karekök (Square Root)

* Goldschmidt

* Newton-Raphson

* Restoring, non-restoring

...

# CORDIC (Coordinate Rotation Digital Computer)

* Trigonometric functions (sine, cosine, tangent, …)

* Hyperbolic functions (sinh, cosh, tanh, …)

* Square-roots

* Exponentials

* Logarithms

…

# CORDIC (Coordinate Rotation Digital Computer)

**Details**

| | |
|---|---|
| Name: | **CORDIC** |
| Version: | 6.0 (Rev. 17) |
| Interfaces: | AXI4-Stream |
| Description: | The Xilinx CORDIC LogiCORE is a module for generation of the generalized coordinate rotational digital computer (CORDIC) algorithm which iteratively solves trigonometric, hyperbolic and square root equations. The core is fully synchronous using a single clock and has AXI4 Stream compliant interfaces. Options include parameterizable data width. The core supports either serial architecture for minimal area implementations, or parallel architecture for speed optimization. The core is delivered through the Xilinx Vivado IP Catalog and integrates seamlessly with the Xilinx design flow. |
| Status: | Production |
| License: | Included |
| Change Log: | View Change Log |
| Vendor: | Xilinx, Inc. |
| VLNV: | xilinx.com:ip:cordic:6.0 |
| Repository: | D:/Xilinx/Vivado/2021.1/data/ip |

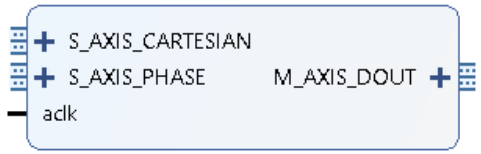
Customize IP

# CORDIC (6.0)

Documentation | IP Location | Switch to Defaults

IP Symbol | Implementation Details

Show disabled ports




Component Name: cordic_0

Configuration Options | AXI4 Stream Options

## Configuration Parameters

| | |
|---|---|
| Functional Selection | Rotate |
| Architectural Configuration | Parallel |
| Pipelining Mode | Maximum |
| Data Format | SignedFraction |
| Phase Format | Radians |

## Input/Output Options

| | | |
|---|---|---|
| Input Width | 16 | [8 - 48] |
| Output Width | 16 | [8 - 48] |
| Round Mode | Truncate | |

## Advanced Configuration Parameters

| | | |
|---|---|---|
| Iterations | 0 | [0 - 48] |
| Precision | 0 | [0 - 48] |

☑ Coarse Rotation

Compensation Scaling: No Scale Compensation