بهینه‌سازی سرعت

بهینه سازی سرعت در توصیف

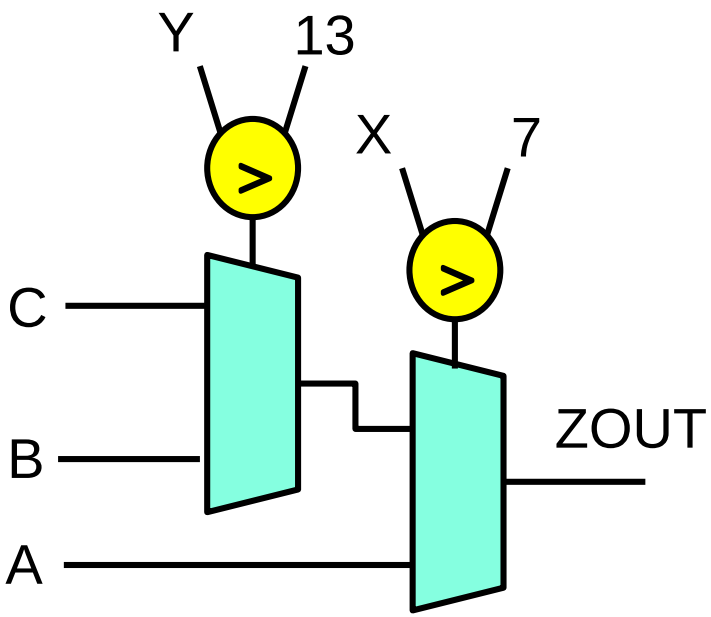
- **مزایای بهینه سازی سرعت در کد RTL:**
  - استفاده مجدد از پودمان بهینه سازی شده در جای دیگر
  - زمان کوتاه تر اجرای ابزارها
  - آسیب ندیدن با ارتقای ابزار به نسخهٔ جدید

روش های بهینه سازی

- **روش های بهینه سازی در کد RTL**
  - کدنویسی با توجه به رفتار ابزار (نتیجهٔ سنتز)
  - Pipelining (خط لوله گذاری)
  - Retiming (باززمانبندی)
  - Proper partitioning

کدنویسی با توجه به رفتار ابزار

- If-then-else تودرتو:
  - انتخاب مناسب سیگنال ها با توجه به زمان رسیدنشان



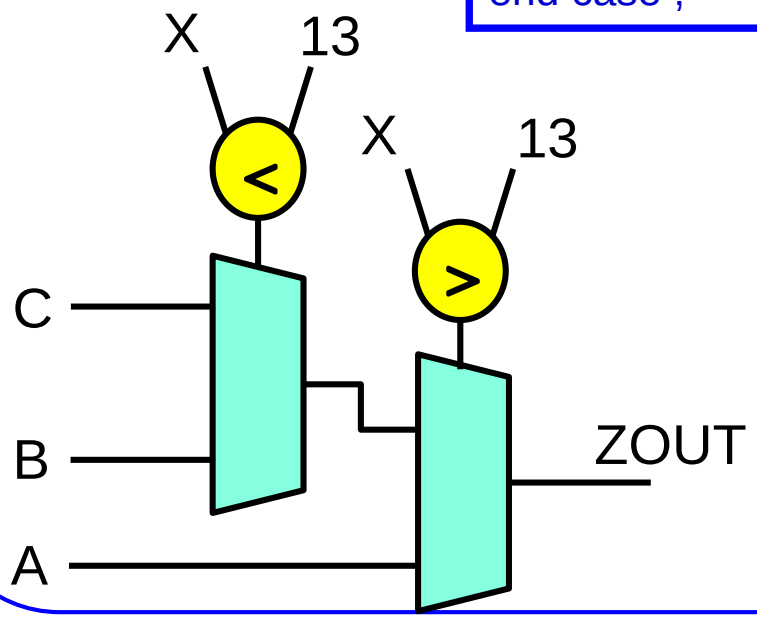

```
if (X > 7) then
   ZOUT <= A ;
elsif (Y > 13) then
   ZOUT <= B ;
else
   ZOUT <=  C ;
end if ;
```

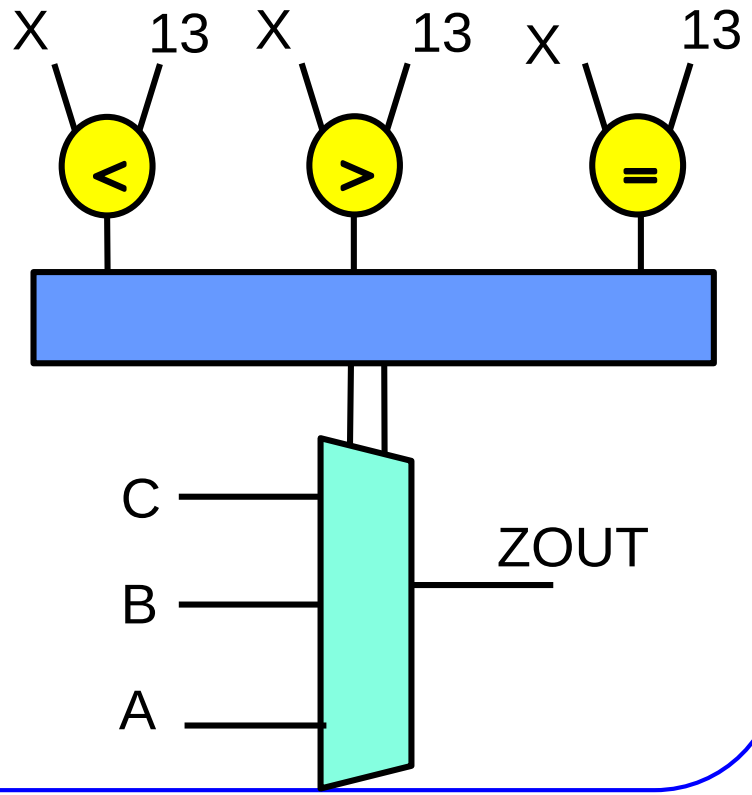
کدنویسی با توجه به رفتار ابزار

- برای شرط های پوشا، case: n:1 MUX ← سریع تر

```
if (X > 13) then
   ZOUT <= A ;
elsif (X < 13) then
   ZOUT <= B ;
else
   ZOUT <=  C ;
end if ;
```

```
case X is
   when 0 to 12 =>
        ZOUT <= B ;
   when 12 =>
        ZOUT <= C ;
   when others =>
        OUT <= A ;
end case ;
```





کدنویسی با توجه به رفتار ابزار

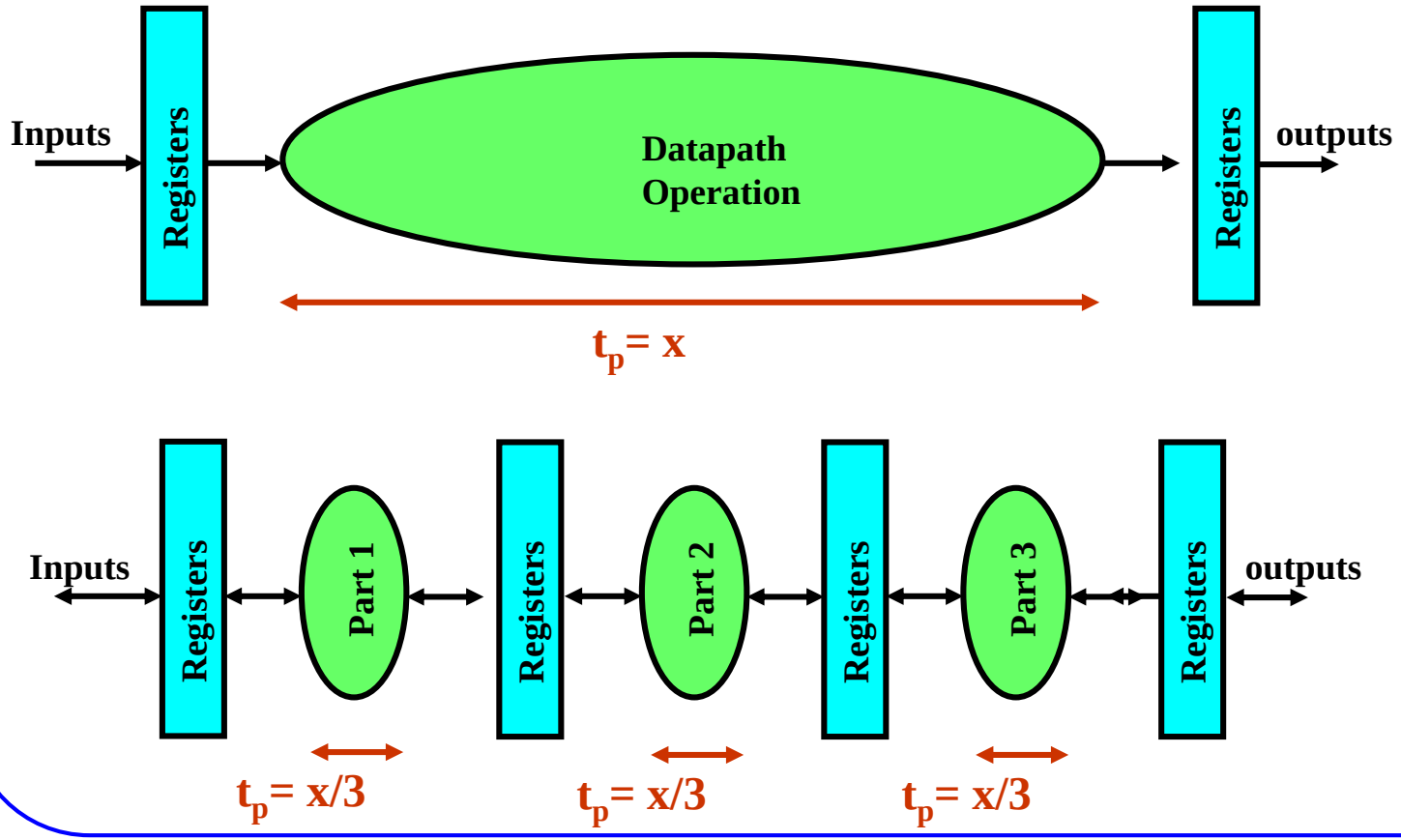
- انتخاب مناسب کدهای حالت در FSM
  - طراحی به صورت مدودف
  - ← آماده شدن خروجی تقریبا همزمان با تغییر حالت
  - کد one-hot
  - مدارهای ترکیبی کوچک ← تأخیر بین FFها: کم ← فرکانس کلاک بالاتر

```
type STATE_T is (STATE0, STATE1, STATE2, STATE3);
signal STATE, NEXT_STATE : STATE_T;

attribute fsm_encoding : string;
attribute fsm_encoding of STATE : signal is "one-hot";
```

Pipelining

- ایدهٔ اصلی:
  - ❑ عملیات datapath بزرگی را که در یک سیکل ساعت انجام می شود به چند عمل کوچک که در چند سیکل انجام می شوند تقسیم کنیم:



Pipelining

- **تحلیل زمانی:**
  - ❑ f سه برابر می شود
    - دقیقاً سه برابر؟
  - ❑ Throughput سه برابر می شود اما خروجی ها سه کلاک دیرتر حاضر می شوند:
    - Latency:
      - Number of clock cycles from the first valid input till the corresponding result available on the output of the pipeline.
    - Throughput:
      - A measure of how often the pipeline produces the valid output.

Pipelining

- **هزینهٔ افزودن رجیسترها:**

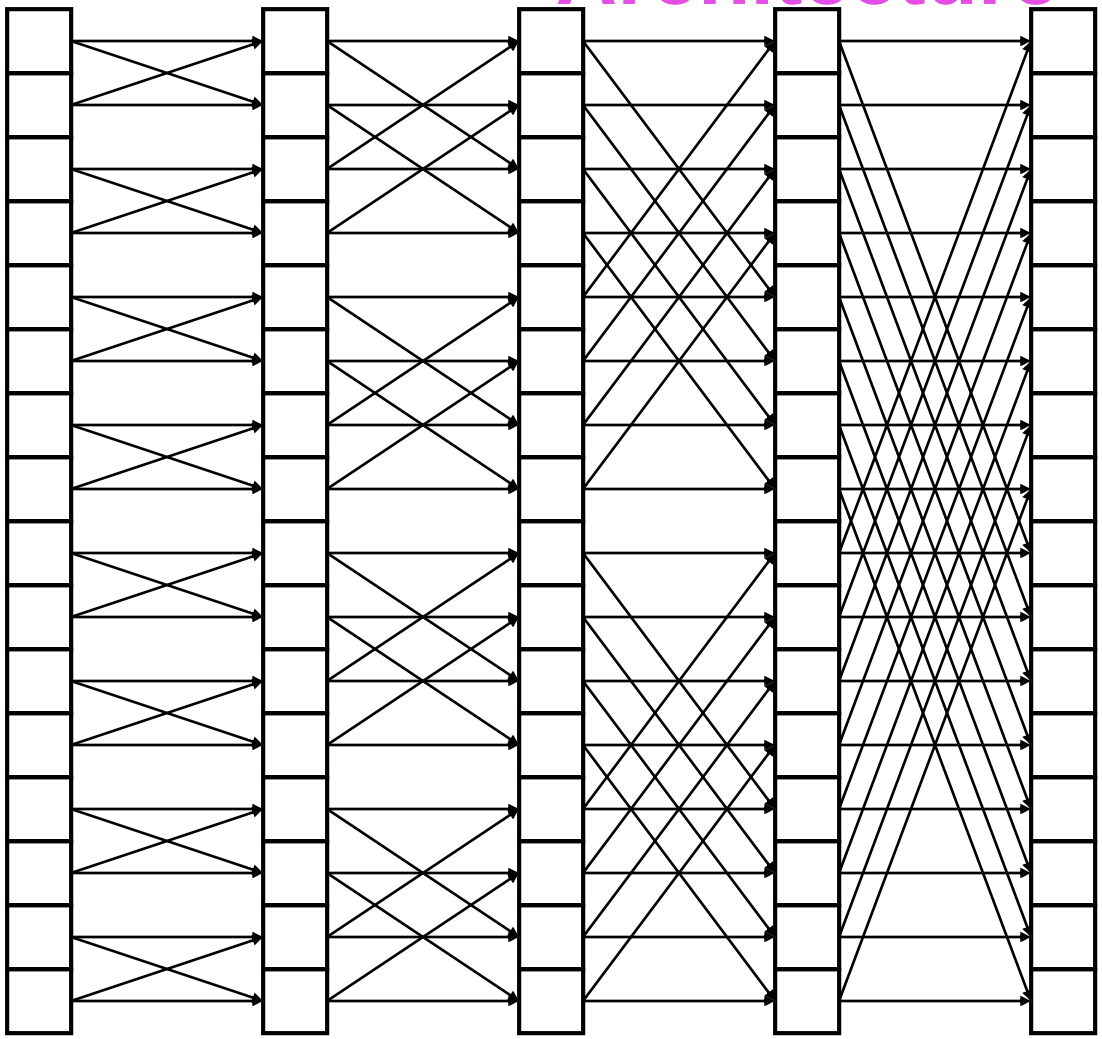
  - بیشتر FPGAها مشکلی ندارند اما در CPLDها pipeline کمتر به کار می رود.
  - CPLDها در یک سطح از آرایهٔ منطقی، عملیات زیادی را می توانند انجام دهند

ANIMATION EXAMPLE,
taken from:

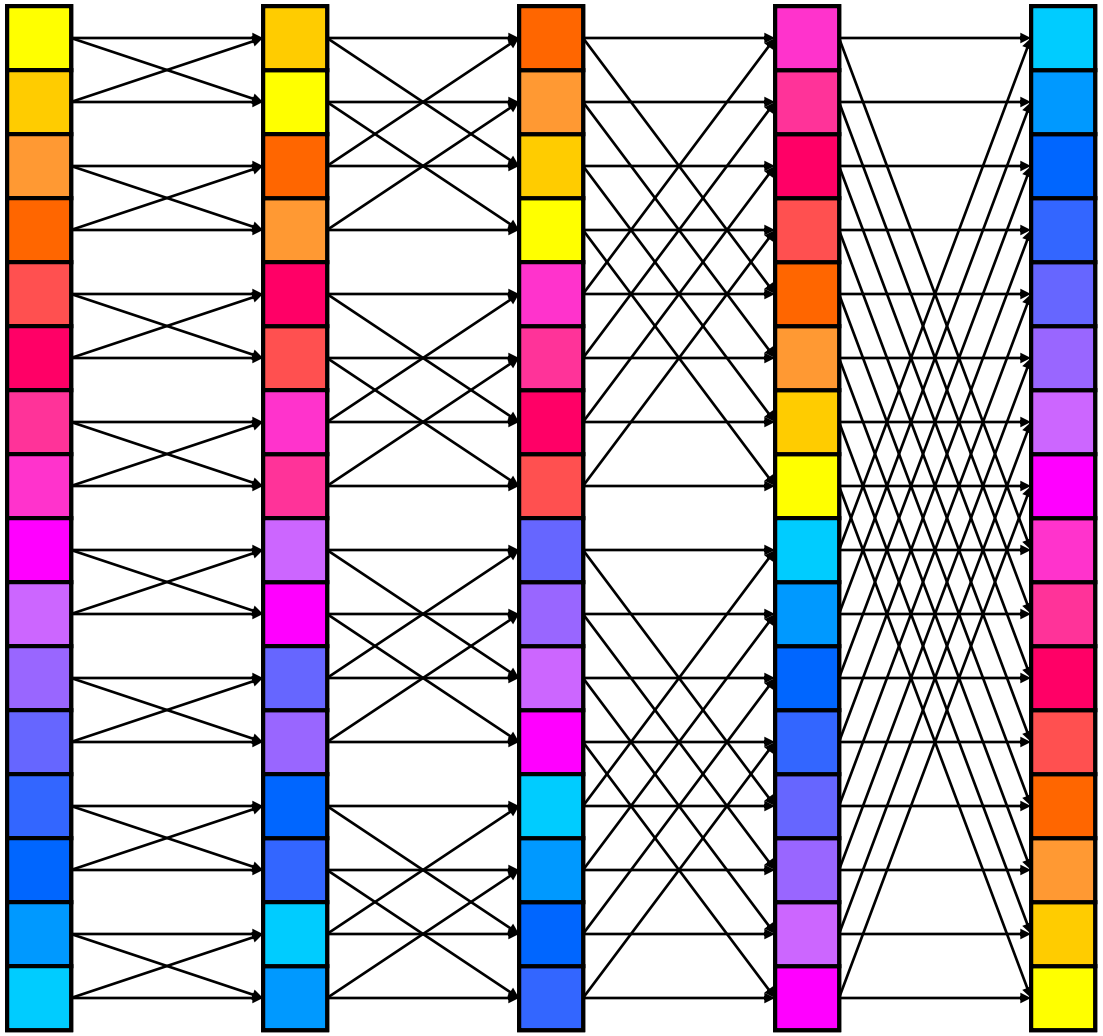
A Systolic FFT Architecture for Real Time FPGA Systems

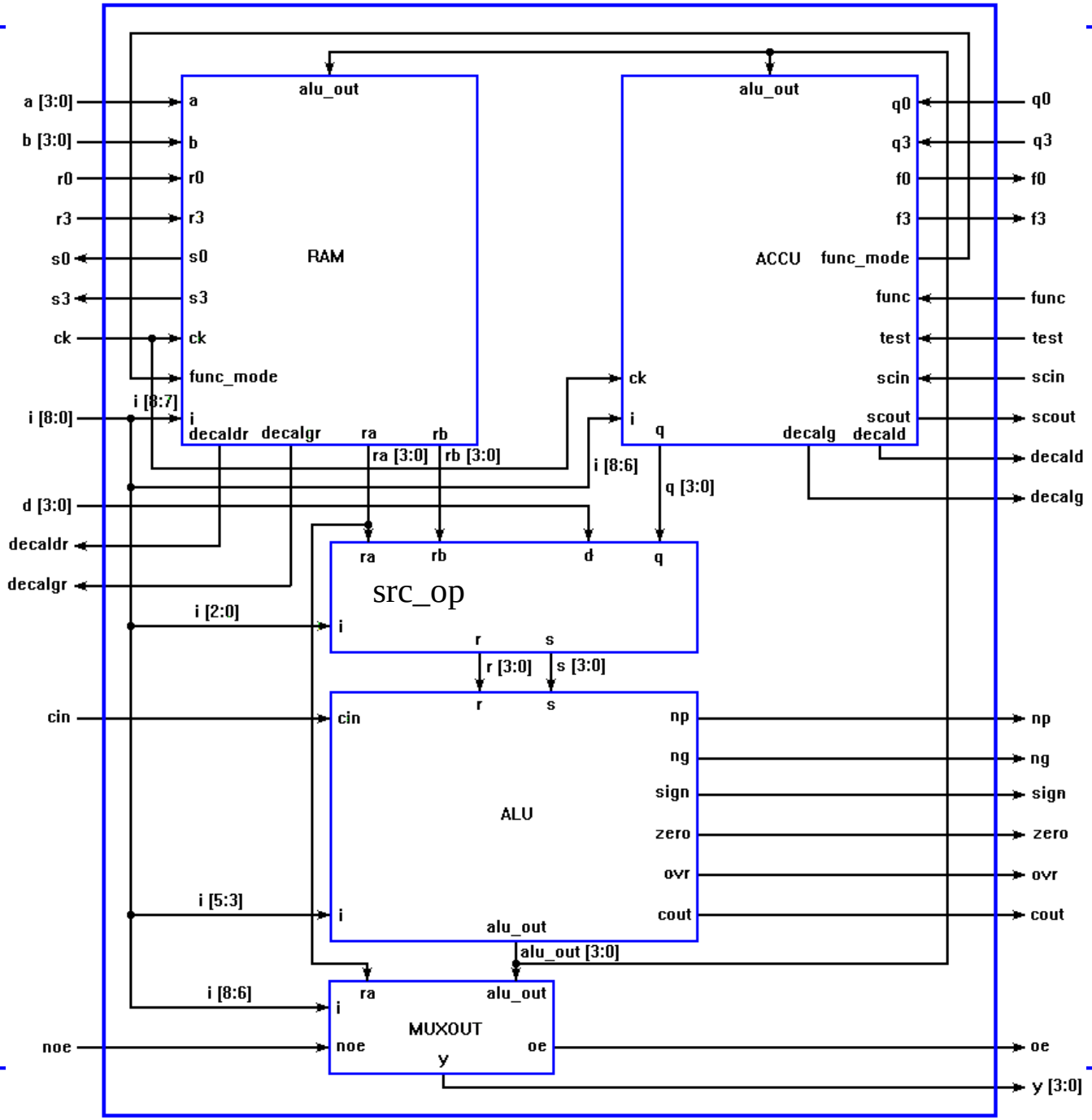
Preston Jackson, Cy Chan, Charles Rader,
Jonathan Scalera, and Michael Vai
HPEC

Example: Baseline Parallel Architecture



Parallel-Pipelined Architecture




a [3:0]
b [3:0]
r0
r3
s0
s3
ck
i [8:0]
i [8:7]
d [3:0]
decaldr
decalgr
i [2:0]
cin
i [5:3]
i [8:6]
noe
alu_out
a
b
r0
r3
s0
s3
ck
func_mode
i
decaldr
decalgr
ra
rb
RAM
ra [3:0]
rb [3:0]
i [8:6]
q [3:0]
ra
rb
d
q
src_op
i
r
s
r [3:0]
s [3:0]
cin
np
ng
sign
zero
ovr
cout
ALU
i
alu_out
alu_out [3:0]
ra
alu_out
i
MUXOUT
noe
oe
y
alu_out
q0
q3
f0
f3
ACCU
func_mode
func
test
scin
scout
decalg
decald
ck
i
q
q0
q3
f0
f3
func
test
scin
scout
decald
decalg
np
ng
sign
zero
ovr
cout
oe
y [3:0]

AMD AM2901

```
library ieee;
use ieee.std_logic_1164.all;
use work.numeric_std.all;
use work.am2901_comps.all;
entity am2901 is port(
  clk, rst:   in std_logic;
  a, b:      in unsigned(3 downto 0);      -- address inputs
  d:      in unsigned(3 downto 0);       -- direct data
  i:      in std_logic_vector(8 downto 0);    -- micro instruction
  c_n:      in std_logic;           -- carry in
  oe:      in std_logic;           -- output enable
  ram0, ram3:   inout std_logic;           -- shift lines to ram
  qs0, qs3:   inout std_logic;           -- shift lines to q
  y:      buffer unsigned(3 downto 0);      -- data outputs (3-state)
  g_bar,p_bar:buffer std_logic;        -- carry generate, propagate
  ovr:      buffer std_logic;           -- overflow
  c_n4:      buffer std_logic;           -- carry out
  f_0:      buffer std_logic;           -- f = 0
  f3:        buffer std_logic);      -- f(3) w/o 3-state
end am2901;
```

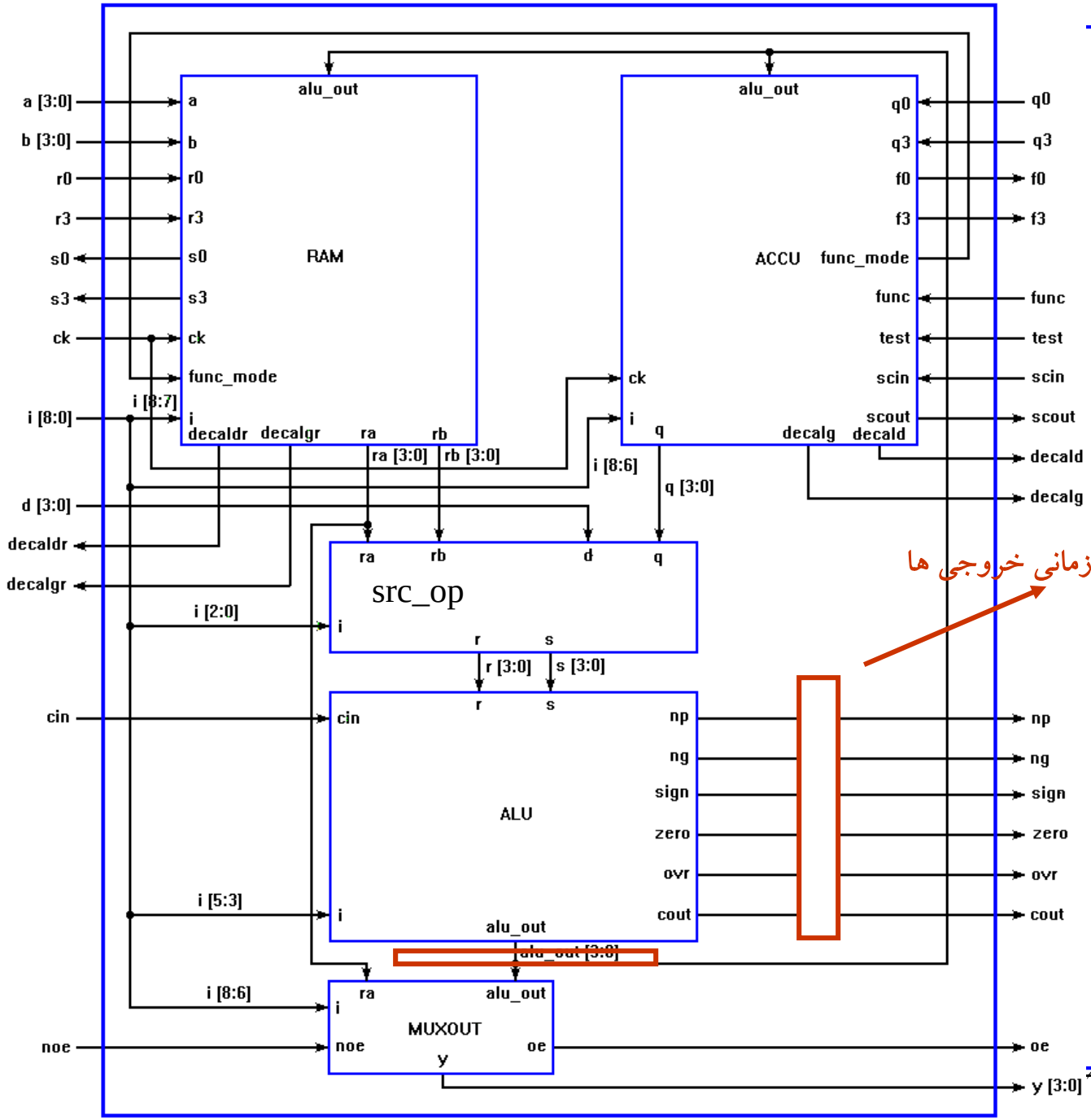
```
architecture am2901 of am2901 is
  alias dest_ctl: std_logic_vector(2 downto 0) is i(8 downto 6);
  alias alu_ctl:  std_logic_vector(2 downto 0) is i(5 downto 3);
  alias src_ctl:  std_logic_vector(2 downto 0) is i(2 downto 0);

  signal ad, bd:   unsigned(3 downto 0);
  signal q:      unsigned(3 downto 0);
  signal r, s:   unsigned(3 downto 0);
  signal alu_out:      unsigned(3 downto 0);
begin

 --instantiate and connect components
u1: ram_regs port map(clk => clk, rst => rst, a => a, b => b, alu_out => alu_out,
            dest_ctl => dest_ctl, ram0 => ram0, ram3 => ram3,
            ad => ad, bd => bd);
u2: q_reg port map(clk => clk, rst => rst, alu_out => alu_out, dest_ctl => dest_ctl,
            qs0 => qs0, qs3 => qs3, q => q);
u3: src_op port map(d => d, ad => ad, bd => bd, q => q,
            src_ctl => src_ctl, r => r, s => s);
u4: alu port map(r => r, s => s, c_n => c_n, alu_ctl => alu_ctl,
            alu_out => alu_out, g_bar => g_bar, p_bar => p_bar,
            c_n4 => c_n4, ovr => ovr);
u5: out_mux port map(ad => ad, alu_out => alu_out, dest_ctl => dest_ctl,
                oe => oe, y => y);

 --define f_0 and f3 outputs
f_0 <= '0' when alu_out = "0000" else 'Z';
f3 <= alu_out(3);

end am2901;
```


a [3:0]
b [3:0]
r0
r3
s0
s3
ck
i [8:0]
d [3:0]
decaldr
decalgr
cin
noe
RAM
alu_out
a
b
r0
r3
s0
s3
ck
func_mode
i [8:7]
i
decaldr decalgr ra rb
ra [3:0] rb [3:0]
ACCU
alu_out
q0
q3
f0
f3
func_mode
func
test
scin
scout
decalg decald
ck
i
q
i [8:6]
q [3:0]
src_op
ra rb d q
i
r s
r [3:0] s [3:0]
i [2:0]
ALU
cin
np
ng
sign
zero
ovr
cout
i
alu_out
i [5:3]
MUXOUT
ra alu_out
i
noe
oe
y
i [8:6]
q0
q3
f0
f3
func
test
scin
scout
decald
decalg
np
ng
sign
zero
ovr
cout
oe
y [3:0]
برای هماهنگی زمانی خروجی ها

Pipelined AMD AM2901

```
library ieee;
use ieee.std_logic_1164.all;
use work.numeric_std.all;
use work.am2901_comps.all;
entity am2901 is port(
  clk, rst:   in std_logic;
  a, b:      in unsigned(3 downto 0);      -- address inputs
  d:      in unsigned(3 downto 0);       -- direct data
  i:      in std_logic_vector(8 downto 0);    -- micro instruction
  c_n:      in std_logic;           -- carry in
  oe:      in std_logic;           -- output enable
  ram0, ram3:   inout std_logic;           -- shift lines to ram
  qs0, qs3:   inout std_logic;           -- shift lines to q
  y:      buffer unsigned(3 downto 0);      -- data outputs (3-state)
  g_bar_q,p_bar_q:buffer std_logic;        -- carry generate, propagate
  ovr_q:      buffer std_logic;           -- overflow
  c_n4_q:      buffer std_logic;           -- carry out
  f_0:      buffer std_logic;           -- alu_out = 0
  f3:        buffer std_logic);      -- alu_out(3) w/o 3-state
end am2901;
```

```
architecture am2901 of am2901 is
   alias dest_ctl: std_logic_vector(2 downto 0) is i(8 downto 6);
   alias alu_ctl:  std_logic_vector(2 downto 0) is i(5 downto 3);
   alias src_ctl:  std_logic_vector(2 downto 0) is i(2 downto 0);

   signal ad, bd:   unsigned(3 downto 0);
   signal q:      unsigned(3 downto 0);
   signal r, s:   unsigned(3 downto 0);
   signal alu_out, alu_out_q:      unsigned(3 downto 0);
begin

 --instantiate and connect components
u1: ram_regs port map(clk => clk, rst => rst, a => a, b => b, alu_out => alu_out_q,
            dest_ctl => dest_ctl, ram0 => ram0, ram3 => ram3,
            ad => ad, bd => bd);
u2: q_reg port map(clk => clk, rst => rst, alu_out => alu_out _q, dest_ctl => dest_ctl,
            qs0 => qs0, qs3 => qs3, q => q);
u3: src_op port map(d => d, ad => ad, bd => bd, q => q,
            src_ctl => src_ctl, r => r, s => s);
u4: alu port map(r => r, s => s, c_n => c_n, alu_ctl => alu_ctl,
            alu_out => alu_out, g_bar => g_bar, p_bar => p_bar,
            c_n4 => c_n4, ovr => ovr);
u5: out_mux port map(ad => ad, alu_out => alu_out _q, dest_ctl => dest_ctl,
                oe => oe, y => y);

 --define f_0 and f3 outputs
f_0 <= '0' when alu_out _q = "0000" else 'Z';
f3 <= alu_out _q(3);
```

No change

Pipelined AMD AM2901

```
  process (clk)
    if (rising_edge(clk) then
      alu_out_q <= alu_out;
      g_bar_q <= g_bar;
      p_bar_q <= p_bar;
      ovr_q <= ovr;
      c_n4_q <= c_n4;
    end if;
  end process;
end am2901;
```

Retiming (باززمانبندی)

- **هدف اصلی:**
  - افزایش فرکانس کلاک مدار ترتیبی
- **اهداف فرعی:**
  - کاهش مساحت
  - کاهش توان مصرفی

Retiming

- **تعریف:**
  - جابجایی FFها در مدار برای بهبود کارایی بدون تغییر latency