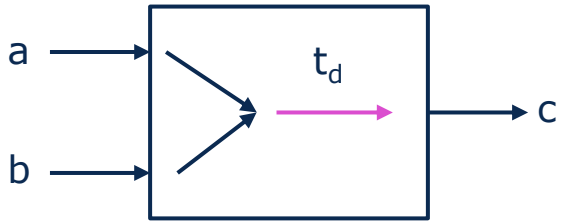
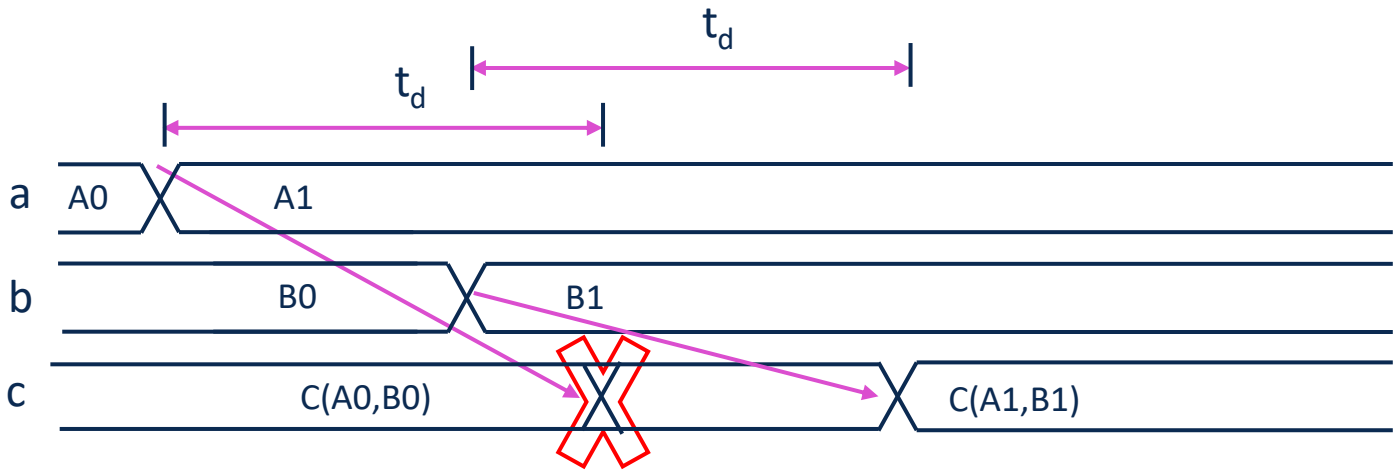
Verilog – Modellierung von Verzögerungszeiten

Modellierung von Verzögerungszeiten

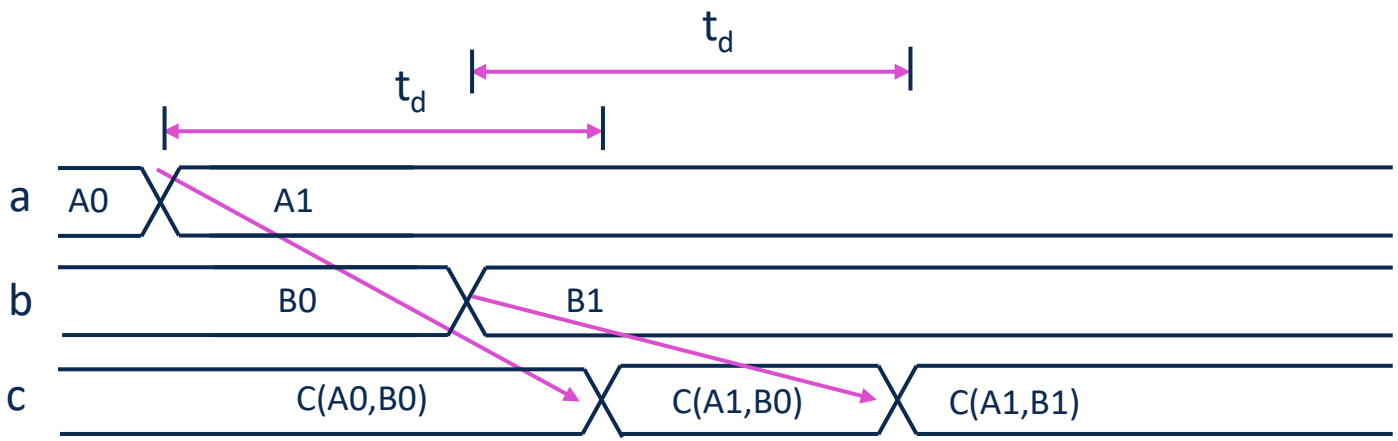


- Abstraktion der Verzögerungszeiten von kombinatorischen Logikblöcken
- Zuweisung von Delays t_d für Timing Arcs
- Berücksichtigung bei der digitalen Schaltungssimulation
- Modellierung zur Simulation als
  - **Inertiales Delay** oder **Transport Delay**

- **Inertiale Delays:**
  - Bei Eingangssignaländerung nach t< t_d → Generierung eines neuen Events, Verwerfen des vorherigen Events
  - Signal-Propagierung zu einem Ausgangssignal **nachdem** die Eingangssignale stabil sind

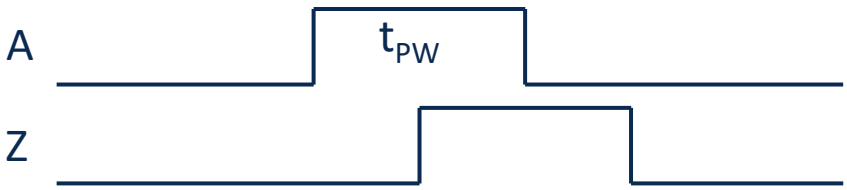


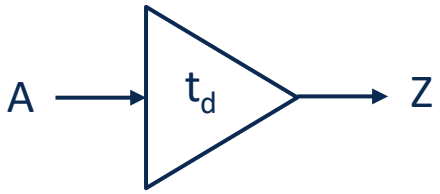
- **Transport Delays**:
  - Propagierung **aller** Eingangsänderungen an den Ausgang
  - Erzeugung von neuen Events bei jeder Eingangssignaländerung
  - Nachteile bei Modellierung kombinatorischer Logik:
    - Propagierung von auch kurzer Glitches
    - Erzeugen vieler Events → langsamere Simulation



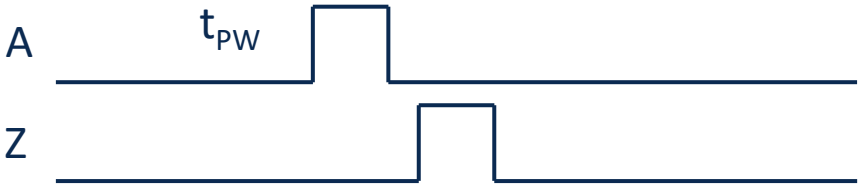
Beispiel: Pulse

$t_{PW}>t_d$

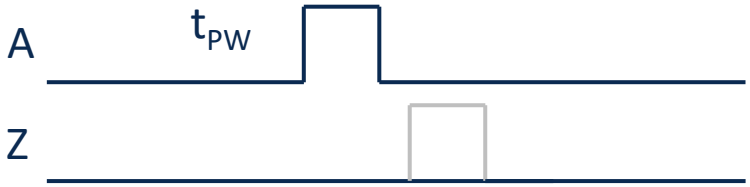




$t_{PW}<t_d$, Transport Delay Modell



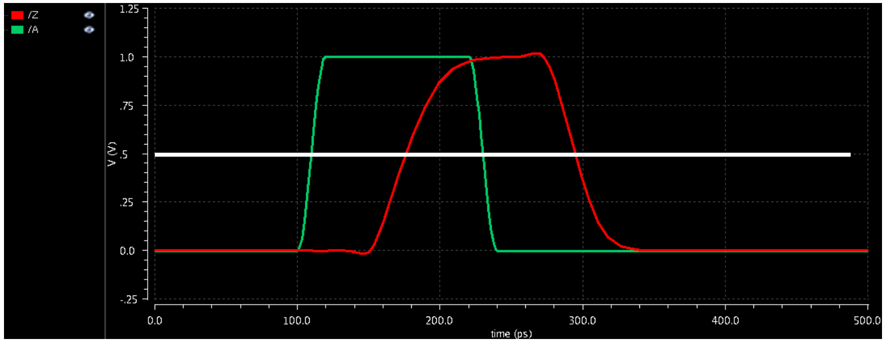
$t_{PW}<t_d$, Inertiales Delay Modell

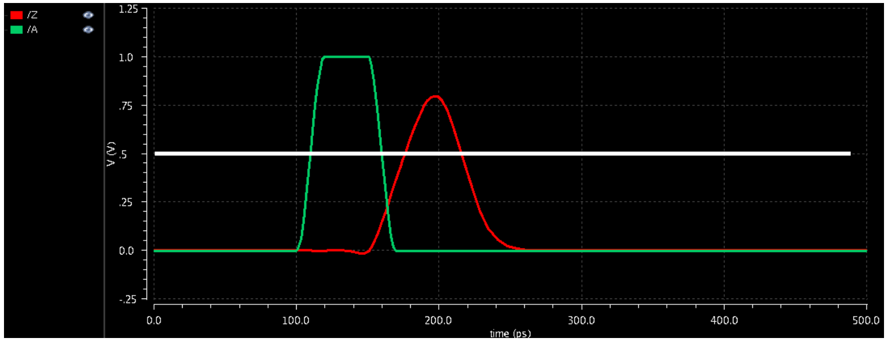


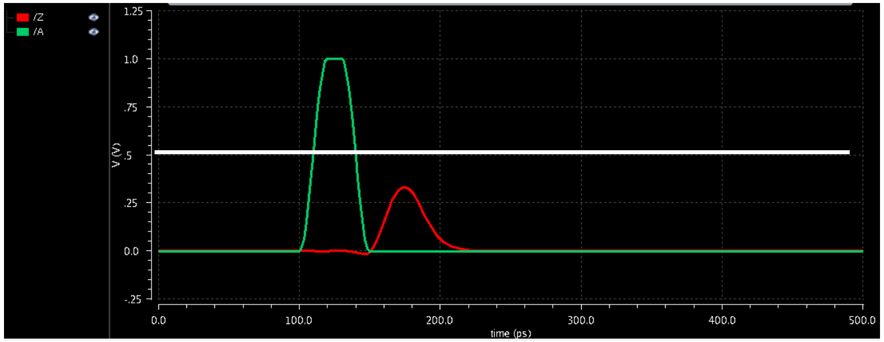
- Typischer Fehler:
  - Default Delay Zuweisungen in Gattern (z.B. 1ns)
  - → Puls von <1ns verschwindet!

Beispiel: Pulse CMOS Gatter

- Simulation von realen Signalverläufen (Transistor-Level)
- CMOS Buffer Kette
  - 1. Pulsweite: 120ps, $t_{d,rise}$ 65ps
  - 2. Pulsweite: 50ps, $t_{d,rise}$ 65ps
    - Puls am Ausgang
    - → Intertial Delay Modell nicht gültig
  - 3. Pulsweite: 30ps, kein Z-Puls
    - → Transport Delay Modell nicht gültig







- Intertial und Transport Delay Modelle sind idealisierte Annahmen
- Ausreichend für Gate-Level Simulationen
- Das reale Delay ist abhängig von der inneren Schaltung, welche hier abstrahiert ist!
- → CMOS Schaltungen

Verzögerungszeiten in Verilog Verhaltensmodellen

- Zuweisungen können mit Verzögerungszeiten versehen werden.
  → #
- ‘timescale <unit>/<resolution>
  - <unit>: Maßeinheit der Zeitangaben
  - <resolution>: Zeitauflösung in Simulation

```
//timescale definition
// 1 ns unit, 10ps resolution
`timescale 1ns/10ps
module testbench ();
…
endmodule
```

- Zuweisungen mit Delay (#) sind nicht synthetisierbar!

Zuweisungen mit Verzögerungszeiten

- Verzögerungszeiten können angegeben werden:
  - bei prozeduralen Zuweisungen der rechten (RHS) und linken Seite (LHS)
  - bei kontinuierlichen Zuweisungen auf der linken Seite (LHS)

```
reg [3:0] opa, opb;
reg [4:0] res_b_rhs,res_nb_rhs,res_b_lhs,res_nb_lhs;
wire [4:0] res_c;
…

//continuous
assign #10 res_c=opa+opb;

//blocking RHS
always @(opa or opb) res_b_rhs= #10 opa+opb;

//blocking LHS
always @(opa or opb) #10 res_b_lhs= opa+opb;

//non-blocking RHS
always @(opa or opb) res_nb_rhs<= #10 opa+opb;

//non-blocking LHS
always @(opa or opb) #10 res_nb_lhs<= opa+opb;
```

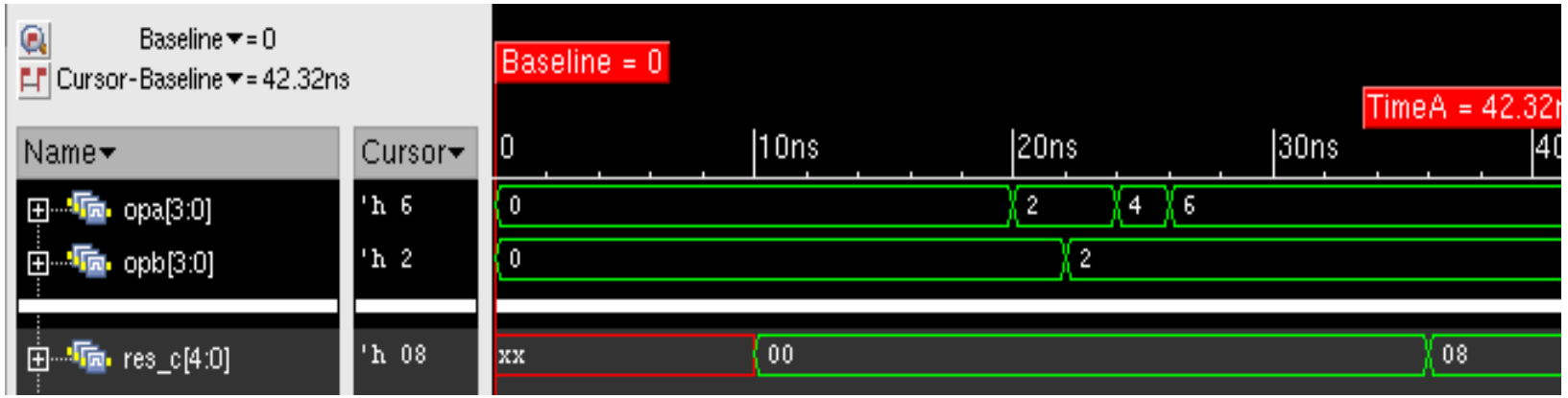
Delay Zuweisung: Continuous Assignment LHS

- Modellierung eines inertialen Delays

```
reg [3:0] opa, opb;
reg [4:0] res_b_rhs,res_nb_rhs,res_b_lhs,res_nb_lhs;
wire [4:0] res_c;
…

//continuous
assign #10 res_c=opa+opb;
```

t [ns]	
20	new event 2→res_c @30
22	cancel event 2→ res_c @30 new event 4→ res_c @32
24	cancel event 4→ res_c @32 new event 6→ res_c @34
26	cancel event 6→ res_c @34 new event 8→ res_c @36
36	**8→ res_c**

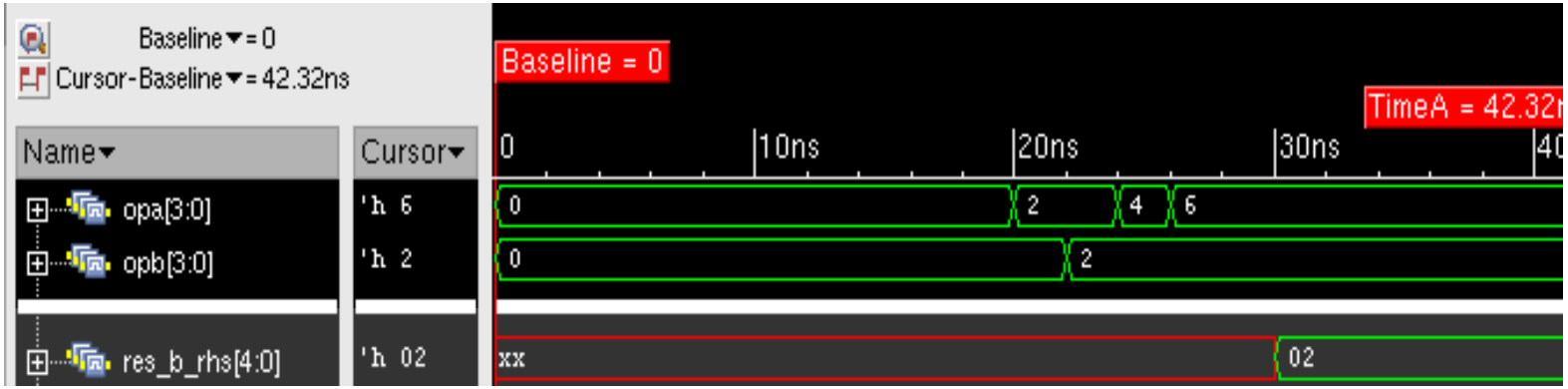


Delay Zuweisung: Blocking Assignment (RHS)

```
reg [3:0] opa, opb;
reg [4:0] res_b_rhs,res_nb_rhs,res_b_lhs,res_nb_lhs;
wire [4:0] res_c;
…

//blocking RHS
always @(opa or opb) res_b_rhs= #10 opa+opb;
```

t [ns]	
20	trigger always block new event 2→res_b_rhs @30 stay in always block until 30
22	toggle opb does not trigger!
24	toggle opa does not trigger!
26	toggle opa does not trigger!
30	**2→ res_b_rhs**

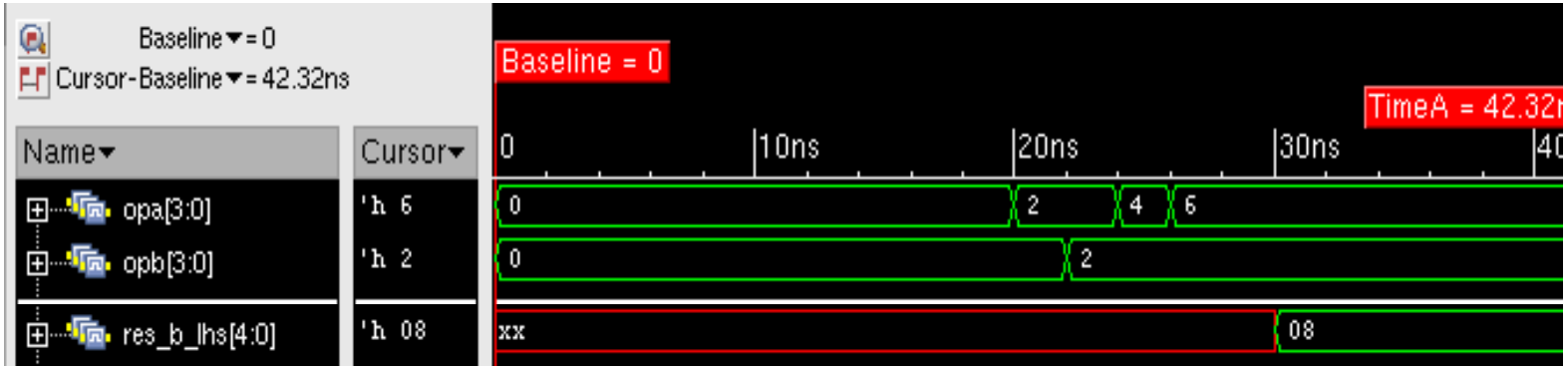


Delay Zuweisung: Blocking Assignment (LHS)

```
reg [3:0] opa, opb;
reg [4:0] res_b_rhs,res_nb_rhs,res_b_lhs,res_nb_lhs;
wire [4:0] res_c;
…

//blocking LHS
always @(opa or opb) #10 res_b_lhs= opa+opb;
```

t [ns]	
20	trigger always block wait until 30
22	toggle opb does not trigger!
24	toggle opa does not trigger!
26	toggle opa does not trigger!
30	new event 8→res_b_lhs @30 **8→ res_b_rhs**



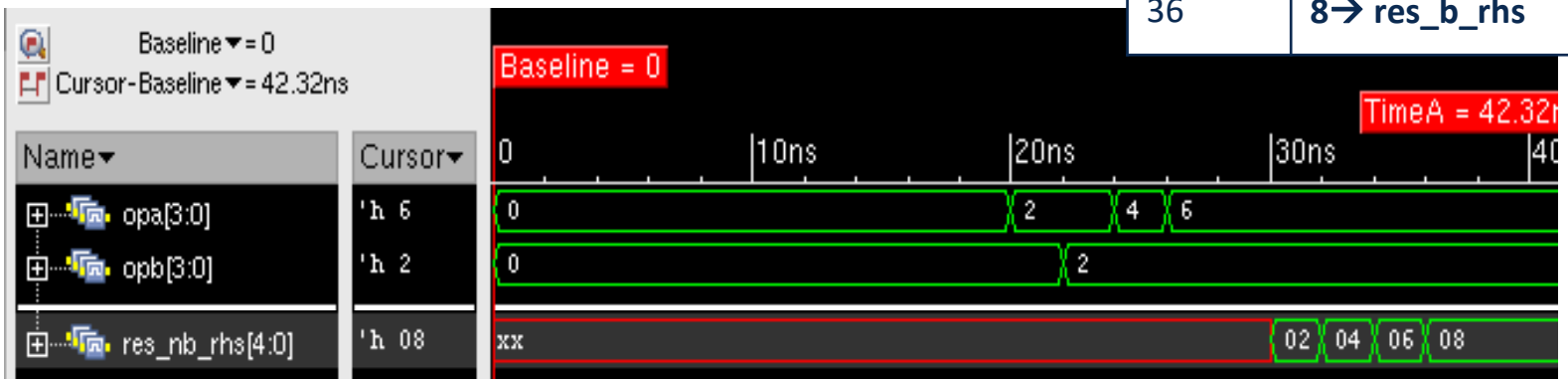
Delay Zuweisung: Non-blocking Assignment (RHS)

- Modellierung eines Transport Delays

```
reg [3:0] opa, opb;
reg [4:0] res_b_rhs,res_nb_rhs,res_b_lhs,res_nb_lhs;
wire [4:0] res_c;
…

//non-blocking RHS
always @(opa or opb) res_nb_rhs<= #10 opa+opb;
```

t [ns]	
20	trigger always block new event 2→res_nb_rhs @30
22	trigger always block new event 4→res_nb_rhs @32
24	trigger always block new event 6→res_nb_rhs @34
26	trigger always block new event 8→res_nb_rhs @36
30	**2→ res_b_rhs**
32	**4→ res_b_rhs**
34	**6→ res_b_rhs**
36	**8→ res_b_rhs**

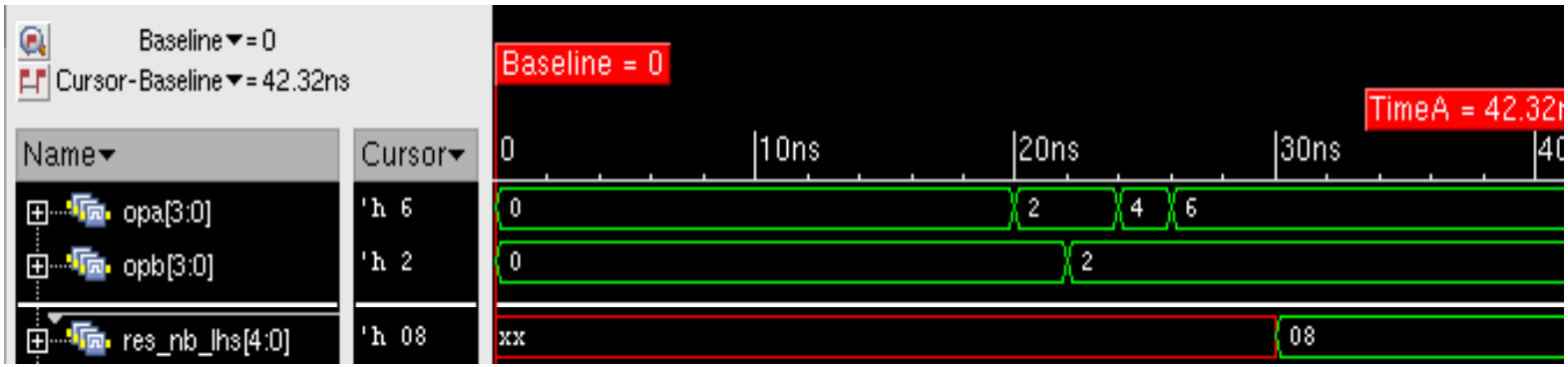


Delay Zuweisung: Non-blocking Assignment (LHS)

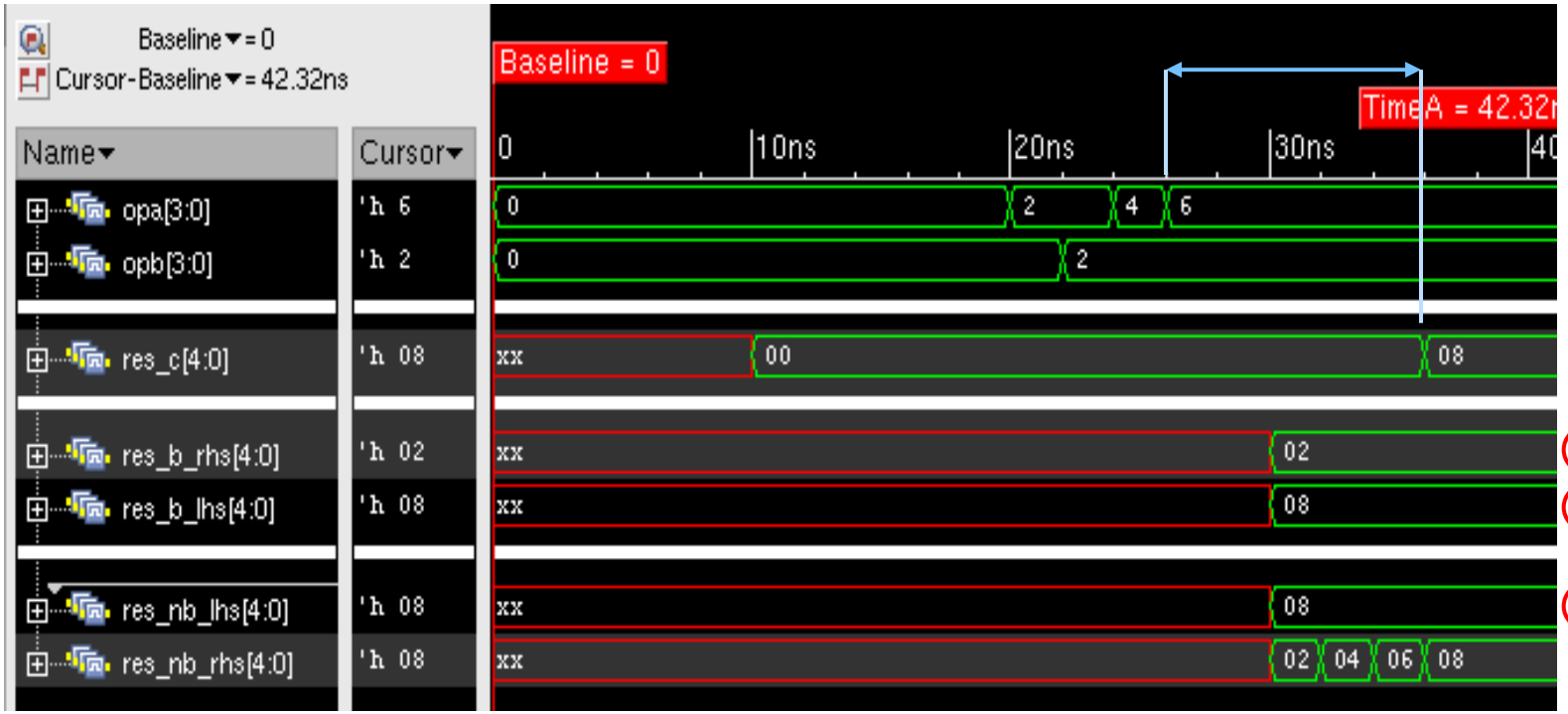
```
reg [3:0] opa, opb;
reg [4:0] res_b_rhs,res_nb_rhs,res_b_lhs,res_nb_lhs;
wire [4:0] res_c;
…

//non-blocking LHS
always @(opa or opb) #10 res_nb_lhs<= opa+opb;
```

t [ns]	
20	trigger always block wait until 30
22	toggle opb does not trigger!
24	toggle opa does not trigger!
26	toggle opa does not trigger!
30	new event 8→res_nb_lhs @30 **8→ res_b_rhs**



Delay Zuweisungen: Zusammenfasssung



inertiales Delay

transport delay

Zusammenfassung: Delay Zuweisungen

Delay	Modelle	Testbench
Continuous Assignment RHS	OK, korrekte Inertial-Delay Modellierung	
Blocking Assignment LHS	Nicht OK	OK, zeitliche Abläufe/Stimuli
Blocking Assignment RHS	Nicht OK	Nicht OK
Non-Blocking Assignment LHS	Nicht OK	Nicht OK
Non-Blocking Assignment RHS	Transport Delays, **ABER: Non-Blocking Assignments für kombinatorische Blöcke nicht OK**	OK,Modellierung von Transport Delays

Modellierung kombinatorischer Logik mit Delay

- Empfohlene Methode zur Modellierung kombinatorischer Logik mit Delay:
  - **Zero-Delay** always block (blocking assignment)
  - Zuweisung des Delays in kontinuierlicher Zuweisung (Intertiales Delay)

```
reg [3:0] opa, opb;
reg [4:0] res_tmp;
wire [4:0] res;

always @(opa or opb) res_tmp= opa+opb;
assign #10 res=res_tmp;
```

- Details unter:

Correct Methods For Adding Delays To Verilog Behavioral Models (1999) by Clifford Cummings; International HDL Conference 1999 Proceedings

Specify Blöcke

- Komplexe Delay Zuweisung für Gate-Level Simulation mit **specify** Statement
  - Separate Rising/Falling Delays
  - Sequentielle Modelle (z.B. posedge CLK → Q Delay)

- Zuweisung von Delays als **Inertiale Delays** für einzelne **Timing Arcs**
- Annotieren der Timings typischer Weise nach Synthese und Place&Route als Ergebnis der Statischen Timing Analyse (STA)
- Zusätzlich Angabe von Constraints möglich
  - Setup & Hold Zeiten $setuphold()
  - Minimale Pulsweite $width()

Beispiel: Verhaltensmodell einer Standardzelle

```
`celldefine
module H_AND2X1_func( B, A, Z );
input A, B;
output Z;

and MGM_BG_0( Z, A, B );

endmodule
`endcelldefine

`celldefine
module H_AND2X1( B, A, Z );
input A, B;
output Z;

H_AND2X1_func H_AND2X1_inst(.B(B),.A(A),.Z(Z));

specify
          (A => Z) = (1.0,1.0); // comb arc A --> Z
          (B => Z) = (1.0,1.0); // comb arc B --> Z
endspecify

endmodule
`endcelldefine
```

Beispiel: Timing-Modell eines FlipFlops

```
`celldefine
module H_DFPQX1( D, CP, Q );
input CP, D;
output Q;
reg notifier;
wire D_delay ;
wire CP_delay ;

H_DFPQX1_func H_DFPQX1_inst(.D(D_delay),.CP(CP_delay),.Q(Q),.notifier(notifier));

   specify
          (posedge CP => (Q : D))  = (1.0,1.0); // seq arc CP --> Q

          $setuphold(posedge CP,posedge D,1.0,1.0,notifier,,,CP_delay,D_delay);
          $setuphold(posedge CP,negedge D,1.0,1.0,notifier,,,CP_delay,D_delay);
          $width(posedge CP,1.0,0,notifier); // mpw CP_lh
          $width(negedge CP,1.0,0,notifier); // mpw CP_hl
   endspecify

endmodule
`endcelldefine
```

Beispiel: Simulation mit annotiertem Timing