Latch-Based Design

The first example illustrates how latch-based design can help when pipeline stages are significantly unbalanced. If we assume we can only place state elements between the combinational blocks labeled A, B, and C, then the clock period using flip-flop based design is 12 (regardless of whether we put the middle register between A and B or between B and C). As drawn in the example, the first stage requires significantly more time than the second stage. Using latch-based design allows the first stage to "borrow" time from the second stage, and ultimately results in a cycle time of 10. Notice all of the additional constraints which latch-based design introduces to ensure proper operation.

The second example illustrates how latch-based design can help mitigate clock uncertainty, even when the pipeline stages are balanced. In this example, the clock skew is 1 and the clock jitter is also 1 for a total clock uncertainty of 2. Clock skew refers to the uncertainty of when the clock arrives at two different state elements, while clock jitter refers to the uncertainty of the clock period for a single state element (i.e., time between when clock edges arrive at any given element). Latch-based design can help amortize clock skew uncertainty over several pipeline stages. In the example, you can see how the flip-flop based design requires an extra delay of 2 before the middle statement and the last state element. In the latch-based design, we only need to budget the timing uncertainty into the last state element assuming the data arrives at the latches in the middle of the transparent window. Also notice that we still need to budget for 2*t_J (twice the clock jitter). This is because in the worst case clock jitter is additive -- we need to budget 1 for the first cycle and 1 for the second click -- the total jitter over the two cycles might be up to 2. While loops can limit the effectiveness of latch-based design, clock jitter can also limit its benefit.

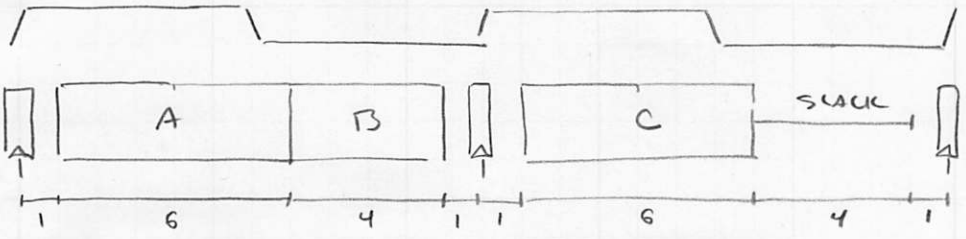
The third example illustrates how we can use useful clock skew instead of latch-based design to balance the pipeline stages from the first example. Notice how we purposefully delay the clock for the middle state element so that it samples the data later than in the original flip-flop based design. Essentially, useful clock skew also allows the first stage to borrow time from the second stage. Since we are still using flip-flops with "hard timing constraints", useful clock skew does not help mitigate clock skew uncertainty.

Our textbook also discusses time borrowing in Section 10.2.4 and the impact of clock skew in Section 10.2.5.

EXAMPLE 1 : UNBALANCED PIPELINE STAGES

ASSUME: $t_{SETUP} = 1$, $t_{PCQ} = 1$, $t_{PDQ} = 1$

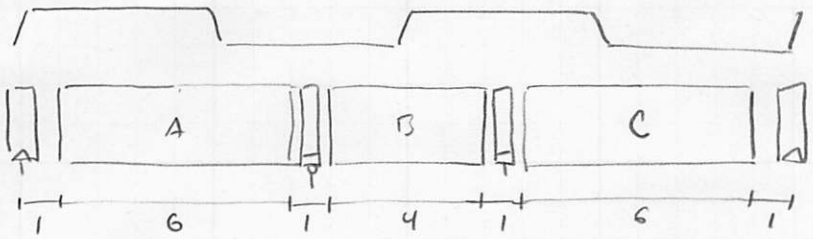
FLIP-FLOP BASED DESIGN



$$t_C = MAX(t_{PCQ} + t_{PD,A} + t_{PD,B} + t_{SETUP},\ t_{PCQ} + t_{PD,C} + t_{SETUP})$$

$$= MAX(1+6+4+1,\ 1+6+1) = MAX(12, 8) = \underline{\underline{12}}$$

LATCH-BASED DESIGN



$$t_C = MAX\Big(\ t_{PCQ} + t_{PD,A} + t_{SETUP}$$

$$t_{PCQ} + t_{PD,C} + t_{SETUP}$$

$$(t_{PCQ} + t_{PD,A} + t_{PDQ} + t_{PD,B} + t_{SETUP})/1.5$$

$$(t_{PCQ} + t_{PD,B} + t_{PDQ} + t_{PD,C} + t_{SETUP})/1.5$$

$$t_{PCQ} + t_{PD,B} + t_{SETUP}$$

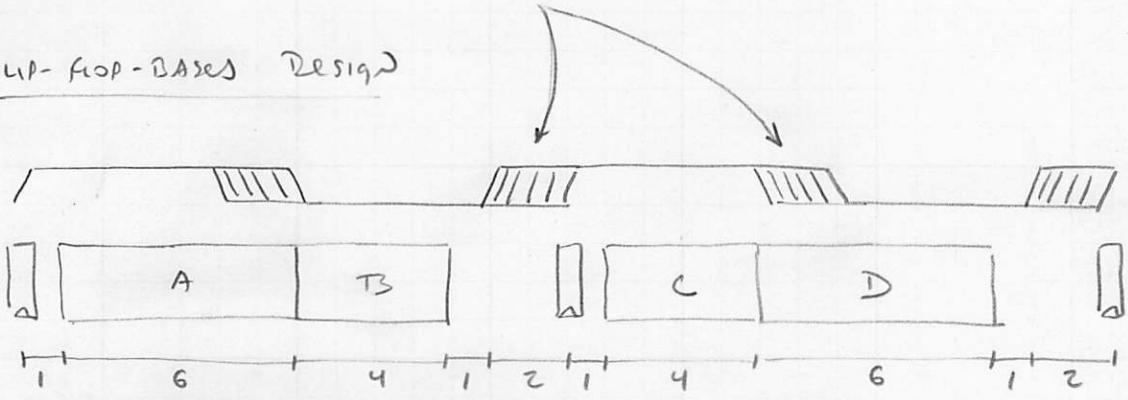
$$(t_{PCQ} + t_{PD,A} + t_{PDQ} + t_{PD,B} + t_{PDQ} + t_{PD,C} + t_{SETUP})/2\ \Big)$$

$$= MAX(8, 8, 8.6, 8.6, 6, 10) = \underline{\underline{10}}$$

EXAMPLE 2: HIDING CLOCK UNCERTAINTY

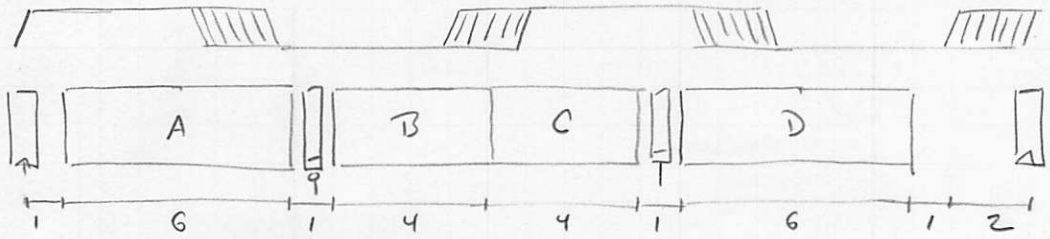
ASSUME: CLOCK UNCERTAINTY, SKEW = $t_{SK} = 1$, JITTER = $t_J = 1$

FLIP-FLOP-BASED DESIGN



$t_c = 14$

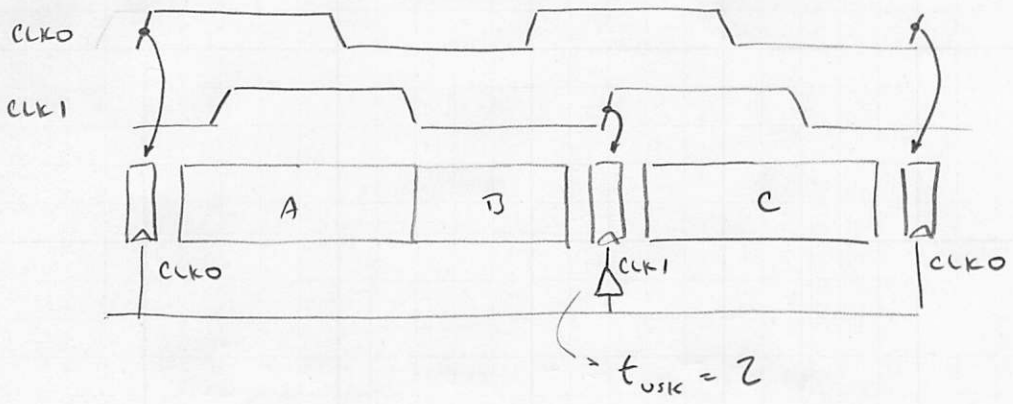
LATCH-BASED DESIGN



$$
\begin{aligned}
t_c = \max\Big(& t_{PCQ} + t_{PD,A} + t_{SETUP} + t_{SK} + t_J, \\
& t_{PCQ} + t_{PD,B} + t_{SETUP} + t_{SK} + t_J, \\
& (t_{PCQ} + t_{PD,A} + t_{PDQ} + t_{PD,B} + t_{PD,C} + t_{SETUP} + t_{SK} + t_J)/1.5, \\
& (t_{PCQ} + t_{PD,B} + t_{PD,C} + t_{PDQ} + t_{PD,B} + t_{SETUP} + t_{SK} + t_J)/1.5, \\
& t_{PCQ} + t_{PD,B} + t_{PD,C} + t_{SETUP} + t_{SK} + t_J \\
& (t_{PCQ} + t_{PD,A} + t_{PDQ} + t_{PD,B} + t_{PD,C} + t_{PDQ} + t_{PD,B} \\
& \quad + t_{SETUP} + t_{SK} + 2 \cdot t_J)/2 \Big)
\end{aligned}
$$

$$= \max(10, 10, 12.6, 12.6, 12, 13.5) = \underline{13.5}$$

EXAMPLE 3: USEFUL CLOCK SKEW



$$t_c = MAX\left(t_{PCQ} + t_{PD,A} + t_{PD,B} + t_{SETUP} - t_{USK},\ t_{USK} + t_{PCQ} + t_{PD,C} + t_{SETUP} \right)$$

$$= MAX(10, 10) = \underline{10}$$

CAN ENABLE TIME BORROWING BUT DOES NOT HELP HIDE CLOCK SKEW UNCERTAINTY.