Logic Synthesis and Implementation Styles in Asynchronous Circuits Design

Victor.Khomenko@ncl.ac.uk
School of Computing Science, Newcastle University, UK
Speed-independence assumptions

- Gates/latches are \textit{atomic} (so no internal hazards)
  
  
  \begin{itemize}
    \item Gate delays are positive and finite, but variable and unbounded
    \item Wire delays are negligible (SI)
    \item Alternatively, [some] wire forks are isochronic (QDI), i.e. wire delays can be added to gate delays
  \end{itemize}
Hazards can be introduced due to these delays!
Good citizens: unate gates/latches, e.g. BUFFER, AND, OR, NAND, NOR, AND-OR, OR-AND, C-element, SR-latch, RS-latch

- Output inverters (‘bubbles’) can be used liberally, e.g. NAND, NOR, as the invertor’s delay can be added to the gate’s delay
- Input inverters are suspect as they introduce delays, but in practice are ok if the wire between the inverter and the gate is short

Suspects: binate gates, e.g. XOR, NXOR, MUX, D-latch – may have internal hazards, but may still be useful
Logic synthesis

- Encoding (CSC) conflicts must be resolved first
- Several kinds of implementation can then be derived automatically:
  - complex-gate (CG)
  - generalised C-element (gC)
  - standard-C implementation (stdC)
- Can mix implementation styles on per-signal basis
- Logic decomposition may still be required if the gates are too complex
Example: complex-gate synthesis

\[ Nxt_z(s) = Code_z(s) \oplus Out_z(s) \]

The size of this Boolean expression is not limited!
Support, triggers and context

Signals that are the inputs of the gate producing a signal form its **support**, e.g. the support of c is \{a,b,c,d\}. Supports are not unique in general.

Signals whose occurrence can immediately enable a signal are called its **triggers**, e.g. the triggers of c are \{b,d\}. Triggers are unique, and are always in the support.

Signals in the support which are not triggers are called the **context**, e.g. the context of c is \{a,c\}. Context is not unique in general.

**support** = **triggers** + **context**
Example: gC implementation

Implemented as pull-up and pull-down networks of transistors + ‘keeper’; assumed to be atomic; risk of transient short-circuit during initialisation
Example: stdC implementation

\[
\begin{array}{cccc}
\text{Code} & \text{Set}_c & \text{Reset}_c \\
0100 & 1 & 0 \\
0000 & 0 & - \\
1000 & 0 & - \\
0110 & - & 0 \\
0010 & 0 & 1 \\
1100 & 0 & 1 \\
1110 & - & 0 \\
1111 & - & 0 \\
1101 & 1 & 0 \\
\text{else} & - & -
\end{array}
\]

‘Monotonic cover’ constraints

\[
\text{Eqn: } \overline{ab} \overline{c} + d, \quad \overline{b}
\]

Hazard due to a new delay

\[\overline{ab} + d\]
Logic Decomposition

- Often complex-gates are too complex to be mapped to a gate library, and so logic decomposition is required.
- Cannot naively break up complex-gates – this is likely to introduce hazards (at least, timing assumptions are required).
- Decomposition is one of the most difficult tasks – no guarantee that automatic decomposition will succeed.
- Online tutorial on logic decomposition and technology mapping:

  https://workcraft.org/tutorial/synthesis/technology_mapping/start