Lecture „HW-SW Codesign“

Exercise IV

Scheduling & SDF

The synchronous data flow (SDF) is a tool that is used to model dependencies in the program flow. This allows generating static schedules, i.e. for mapping the program tasks onto a single or multiple processors with respect to the valid order of execution. The SDF model is very simple. It consists of a set of actors (→ tasks) that are connected by wires (→ dependencies). Each output of an actor is annotated with an amount of data that it produces when it’s fired (i.e. when the task is executed). On the other hand each input of an actor is annotated with the amount of data that is required for the actor to be fired (threshold). The data at the input can be accumulated over several cycles. This enables the support of actors with different sample rates. A small example gives figure 1.

Fig. 1: SDF Example

1) Sequential Scheduling: Assume the following SDF given in figure 2.

Fig. 2: SDF with 5 actors (circles) and 6 dependencies (rectangles)
(a) Determine the firing rates for each actor.
(b) Build the topology matrix $\Gamma$ for the SDF graph in Fig. 2.
(c) Does any admissible (executable) schedule exist for this SDF? I.e. are there any sample rate inconsistencies?
   [Hint: The rank of the topology matrix yields some important information here!]
(d) What’s the situation if actor 4 produces 2 instead of 1 data token when it’s fired?
(e) Find an admissible sequential schedule. Try to find it by intuition first. Apply Lee’s sequential scheduling algorithm [1] thereafter:

\begin{enumerate}
    \item Find integer vector $q$, such that $\Gamma \cdot q = 0$ ($q_i$ determines how often actor $i$ is switched)
    \item Build arbitrary ordered list of all actor nodes $L$
    \item Schedule each $\alpha \in L$, if it is runnable $\rightarrow$ whereas runnable means:
        \begin{itemize}
            \item $\alpha$ was scheduled less than $q_\alpha$ times
            \item all input buffers of node $\alpha$ contain enough samples to be consumed, i.e.
            \end{itemize}
            \begin{equation}
                b_\alpha + \Gamma(a, \alpha) \geq 0 \text{ for all arcs } a
            \end{equation}
        \end{itemize}
    \item Stop, if every node $\alpha$ has been scheduled $q_\alpha$ times
    \item Deadlock, if no $\alpha$ is runnable anymore
\end{enumerate}

2) **Multi-Processor Scheduling:** Scheduling on $M$ multiple parallel processors means to find a set of $M$ sequential admissible schedules $\{\Psi_i; i = 1, \cdots, M\}$. For the case of MP scheduling the runtime of the tasks has also to be considered to be aware when a certain processor is idle and ready for new tasks. Assume the SDF from figure 3. This time the task execution times are annotated (in blue) at the output edges of each actor. For simplicity the node number of the actor itself is also identical with the runtime of the task:

\begin{center}
\begin{tikzpicture}
    \node (1) at (0,0) {1};
    \node (2) at (2,2) {2};
    \node (3) at (2,0) {3};
    \node (4) at (4,2) {4};
    \node (5) at (4,0) {5};
    \draw [->] (1) -- node[above] {1} (2);
    \draw [->] (2) -- node[above] {2} (4);
    \draw [->] (3) -- node[above] {3} (4);
    \draw [->] (2) -- node[above] {3} (3);
    \draw [->] (4) -- node[above] {4} (5);
    \draw [->] (5) -- node[above] {5} (1);
\end{tikzpicture}
\end{center}

Fig. 3: SDF with annotated task runtime

I.e. the task runtime for every actor can be given by the following vector: $T_R = (1, 2, 3, 4, 5)$. 
(a) Draw the Precedence Graph for J=1 (i.e. every actor has to be fired at least once)
(b) Consider the slightly modified SDF in figure 4 that now contains an internal loop between actor 2 and 4 (the input buffer of actor 2 initially contains 2 tokens). Draw the Precedence Graph for J=2 (i.e. every actor has to be fired at least twice).

(c) Determine the minimum execution time (longest path in figure 3) for any parallel schedule by using Max-Plus algebra: $A \oplus A^2 \oplus \ldots$

(d) Determine the parallelism profile ($N_i$: the number of parallel occupied processors at clock cycle $i$) by using Max-Plus algebra

(e) Find an optimum parallel schedule using the Hu-level scheduling algorithm [1][2] for M=2 parallel processors:

1. Build Precedence graph from sequential schedule $\rightarrow$ (a) (J=1)
2. Annotate each node with a level
   $\rightarrow$ level = critical path length from given node to any terminal (i.e. output) node (here it’s node 5)
3. Schedule node with highest level on any free processor (take node with highest runtime, if there are more than 1 possible nodes; schedule node only after all nodes with higher level have been finished)
4. Remove the scheduled node from the Precedence graph $\rightarrow$ stop if empty, otherwise continue with 3

References: