AN UPPER BOUND OF THE THROUGHPUT OF MULTIRATE MULTIPROCESSOR SCHEDULES

Rainer Schoenen *Vojin Živojnović* *Heinrich Meyr*

Institute for Integrated Systems in Signal Processing
Aachen University of Technology
Templergraben 55, 52056-Aachen, Germany
{schoenen,zivojnov,meyr}@ert.rwth-aachen.de
WWW: http://www.iss.rwth-aachen.de/

ABSTRACT

Multirate Dataflow Graphs (MR-DFGs) are used for modelling iterative computations, allowing concurrency and arbitrary data rates at ports. This model is often used for signal processing algorithms. For static scheduling the *iteration*[1] *period bound* represents the final barrier for the computation speed, the approximation of which is often the goal of an implementation. For the singlerate case (SR-DFG), where all rates are one, an explicit bound exists and is subject of many published papers. This work presents a bound for the multirate case, which reduces to the known bound if applied to an SR-DFG. Assumptions made are a vectorized execution and a blocked schedule that organizes multiple iterations inside one period (also called execution cycle). The influence of characteristic properties in the multirate case is emphasized and related to terms from the Petri-Nets theory.

1. INTRODUCTION

Designing multiprocessor implementations of algorithms is a task where aspects of speed performance are important. The number of processors involved in parallel processing has to be determined and the necessary and achievable throughput[2] must be considered. It is very useful to have an overview on parameters that influence the throughput and see where critical loops exist that might be improved by changing obvious attributes of the algorithm. The feasibility of an implementation can also be checked by having a bound to which the requirements can be compared.

We assume that a dataflow [1] description of the algorithm exists, denoted as a dataflow graph (see section 2). Scheduling means finding an effective mapping of calculation blocks to processors and time (slots) [2]. For recursive algorithms the throughput is limited by the graph topology and the time needed for the processing blocks on hardware [3]. The system designer is interested in determining how an implementation will perform on a given multiprocessor architecture first assuming an infinite number of processors. A maximum utilizable number of processors can be computed as shown later so that early assumptions can be relaxed. Bottlenecks in the design can soon be detected and the calculated values help to decide whether to choose another platform or to spend less processors.

If there are no directed cycles in the underlying graph the throughput is theoretically unlimited [4]. If directed cycles are present, the maximum throughput can be calculated by analyzing all the decomposed fundamental cycles alone [5] and taking the minimum throughput over all cycles. Although the number of cycles in general is not polynomially bounded in the number of nodes [6], in a great deal of practical problems the number of cycles can be taken as limited. The minimum throughput over all cycles has to be taken because the algorithm can only be scheduled as fast as the slowest cycle (critical loop). The reduced problem of one single cycle can be tackled either by converting the multirate graph to a singlerate DFG [1] by applying the multirate to singlerate graph transformation, which is generally an NP-hard task, and then using known bounds [6] or by using the presented method on the MR-DFG. So, we address the problem of having an explicit formula for the maximum achievable throughput of the MR-DFG without exploring all the possible schedules of the graph. The generally NP-complete complexity of the multiprocessor scheduling algorithm and its reduced information about bottlenecks are reasons why an explicit approach is preferred.

The paper is organized as follows. After the introduction, the used terminology is given in Section 2. The related work is discussed in Section 3. We present our approach in Section 4, prove its validity in Section 5, and conclude with an example in Section 6.

2. THE DATAFLOW GRAPH

A MR-DFG is defined as a directed graph $G(V, E, I, O, d, T)$ containing the set of vertices V and edges E with weights $O(e)$ and $I(e)$ (the rates) on the output and input side of arc e respectively. The nodes represent data processing units while the edges mean data exchange channels. The weights $O(e)$ and $I(e)$ define how many discrete data tokens are created and consumed during execution of a node process. Nodes can only be activated if on each input arc e the number of tokens ist greater or equal than $I(e)$. The $\vec{d}$-vector denotes the (initial) sample (token,delay) distribution, also known as *marking* in the literature on Petri nets [7]. Each $d[e]$ gives the number of tokens in the exchange buffer associated with arc e between nodes. $T(v)$ is the known computation time of node v. This graph type is equivalent to the Petri-Nets class *multiple weighted marked graph* [8]. The

[1] one iteration is a complete task sequence after which the initial schedule state is reached again

[2] throughput is measured by iterations/time, the inverse of the iteration period.

$\vec{q}$-vector is the smallest integer vector of the null-space of the incidence matrix Γ ($\Gamma\vec{q} = \vec{0}$) and exists iff the graph is consistent[3]. It defines how many activations $q[v]$ of each node v are necessary to complete one iteration, at the end of which the delay distribution $\vec{d}$ is the same as at start time.

Each graph can be decomposed into c fundamental cycles [5]. The example graph (one cycle) in fig. 1 has the properties $\vec{d}^T = (8,0,0), \vec{q}^T = (10,5,4)$, so e.g. node v_2 must be activated 5 times in one iteration. Its input arc e_1 has only 8 initial tokens, so only two activations can actually occur at this time.

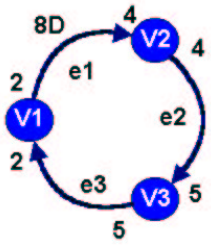

Figure 1. Example graph

In the singlerate case the above are the only necessary values, but here we must define additional terms from the Petri-Nets theory. Some of them are necessary for the application of our bound, others are characteristics of MR-DFGs and appear in the example.

We now define

$$WST_l = \vec{y}_l \cdot \vec{d} \tag{1}$$

This *Weighted Sum of Tokens* [7] is a measure of how many initial values are on cycle l. The subscript l denotes that the variable characterizes a particular cycle. The value of WST_l is invariant during the schedule. The weight vector $\vec{y}_l$ [7] is a row l of the fundamental circuit matrix C [7] ($y[e]$=weight of arc e). It is necessary for balancing the node gain [8] (e.g. one delay token can become two after an interpolator). Matrix C of dimension $c \times |E|$, with a number c of cycles, contains a row for each fundamental cycle, while each row contains zeroes for the arcs not contained in the corresponding cycle and integer values else. For the graph in fig. 1 this is simply $\vec{y}^T = (1,1,1)$.

The *Comfortable Marking*

$$CM_l = I[e] \cdot q[target(e)] \cdot y[e] \tag{2}$$

specifies the amount of tokens necessary on any arc e of cycle l to activate the following node $v = target(e)$ exactly $q[v]$ times, the total execution count of this node v in one iteration. Because of consistency each other node can then also be activated that way (i.e. an arbitrary e works).

Liveness is necessary for a valid schedule, but in the MR case this is not just given by $WST_l > 0$ for each loop, as it was in the SR case. We need to have at least a specific number WST on the loop that even depends on the reachability set of a given $\vec{d}_0$ [9] [10]. We can also define the number LLM (lightest live marking) [9] , below which a cycle cannot be live, and HDM (heaviest dead marking) [9], above which a cycle with WST is always live (schedulable).

$$HDM_l = \sum_{e \in Loop} [I(e) - 1] \cdot y[e] \tag{3}$$

The relations $1 \leq LLM \leq HDM + 1$ and $CM \geq LLM$ always hold. In contrast, for singlerate graphs we simply have

$$y[e] \in \{0,1\}, \vec{q} = \vec{1},$$
$$WST = D_{sum}, CM = 1, LLM = 1, HDM = 0. \tag{4}$$

The packing factor m is used to denote the compaction of m iterations to one period (also called execution cycle [11]). The resulting schedule is then only periodic over the whole period, not necessarily over one interation. This m corresponds to the unfolding or blocking factor J [11].

For reasons of optical simplification of the results an *equalization procedure* (see [10] for details) can be applied. This normalizes all rates at one node to the same value $s[v]$ leading to $y[e] \in \{0,1\}$, preserves all scheduling characteristics, multiplies the delays on arcs and lets us use a simpler D_{sum}, which is the sum of the new number of delays but equals the previous number WST of the original graph, where WST is generally not the simple sum of original delays. This works by multiplying all values $O(e), I(e), D[e]$ of any arc by $y[e]$.

Vectorization [12] of a DFG means applying the retiming transformation to accumulate tokens before single nodes to enable the maximum number of parallel (vectorized) executions of these nodes.

3. RELATED WORKS

The throughput bound TB of SR-DFGs is shown to be [3]

$$TB_{SR} = \min_{all\ cycles\ l} \frac{D_{sum,l}}{T_{sum,l}} = \min_{all\ cycles\ l} \frac{\sum_{e \in l} D[e]}{\sum_{v \in l} T[v]} \tag{5}$$

where the sum of tokens $D[e]$ and the sum of the execution times $T[v]$ need to be known. This result is also known from the theory of marked graphs in the context of Petri-Nets [7], the theory of which is also very useful for DFG related topics. Approaches to reduce the asymptotic complexity in this case are given in [13] and [6].

There are publications concerning the multirate iteration bounds. A quite imprecise metric is given in [7] (linear in WST). Another approach [8] is only valid at certain points (this point is the defined CM from eq. 2). An optimized approach for the conversion to a SR-DFG with removal of some redundancy is given in [14].

A typical relation of the maximum throughput vs. number of delays is presented on figure 2. We observe a huge gap between the linear limit and the maximum throughput depending of the value of WST. The more tokens we add on the arcs the more we increase WST. The throughput is monotonic over WST, but not each added token leads to an increase in throughput.

[3] Consistency is now assumed to be valid in this paper

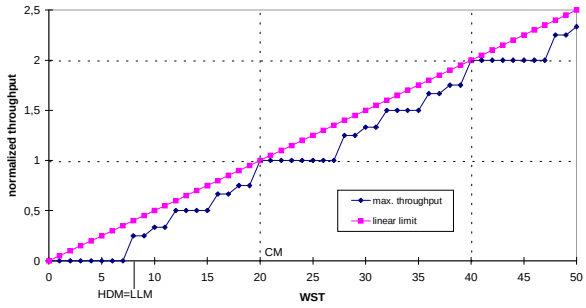

Figure 2. Normalized throughput vs. WST

Results regarding the optimum unfolding or blocking factor $m = J$ are given in [11]. In the multirate case the proposed value is only valid for WST being an integer multiple of CM. The mentioned sum of normalized delays is exactly WST/CM.

4. COMPUTATION OF THE UPPER BOUND

Beginning with a multirate graph we apply a procedure to calculate our throughput bound TB with equation 6. The packing factor m of the final implementation is assumed to be given, because the results depend on the way how effective a time interval, the mth multiple of the iteration period time, can be filled with scheduled nodes in a blocked way (non overlapping). Additional information needed is the activation frequency q_i of node i, the number of delays D_{sum}, the input rate $I(i)$ on the input arc of node i of the cycle l and a scaling factor $y[E(i)]$ of this input arc $E(i)$, if the graph is not equalized. The scaling is not mentioned in the formula, so an *equalized* graph is assumed. It can be generalized by replacing D_{sum} with $\lfloor WST/y[E(i)] \rfloor$.

The procedure to compute the throughput bound is as follows:

- Equalize Graph G [10] or replace D_{sum} by $\lfloor WST/y[E(i)] \rfloor$ later in eq. 6.
- Choose an m, e.g. J_{opt} from [3].
- Decompose G into its cycles [5].
- Calculate D_{sum} or WST, CM and $\sum T_i$ on this cycle.
- Check the range of D_{sum} or WST w.r.t. CM, apply eq. 7 if necessary.
- If $D_{sum} \geq LLM$ apply eq. 6, else TB is zero.

The following formula represents a bound for multirate multiprocessor schedules for one cycle (D_{sum} is the independent variable).

$$TB(D_{sum}, m) = \frac{m}{\max_{i \in L} \lceil \frac{m \cdot q_i}{\lfloor \frac{D_{sum}}{I(i)} \rfloor} \rceil \cdot \sum_{i \in L} T_i} \qquad (6)$$

Equation 6 is valid in the range $LLM \leq WST = D_{sum} < CM$. Below LLM $TB = 0$ holds (the graph isn't live). At each integer multiple z of CM the behavior is periodic plus a stair of $z/\sum T_i$, therefore

$$TB(D_{sum}) = \lfloor \frac{D_{sum}}{CM} \rfloor \cdot \frac{1}{\sum T_i} + TB_{eq.6}(D_{sum} \bmod CM). \qquad (7)$$

We could even normalize the formula by multiplying $\sum T_i$, leaving the resulting equation 6 independent of time, emphasizing the influence of the structure. The characteristic invariant value CM describes where a time-normalized throughput reaches integer multiples of one. The bound in [8] is only valid in these points. The overall-bound is then the minimum of TB_l over all cycles l. We emphasize that this bound is not tight in all cases. The problem is that the special tight bound not only depends on WST but also on the reachability set $R(\vec{d})$. It is possible that some token remainders are never movable within a path, these hardly contibute to the throughput [10].

5. PROOF

The derivation of eq. 6 is presented below. We assume the tokens of the cycle to be concentrated on a single arc (achievable by applying the multirate retiming transformation [12] [10]). Even if possibly not all of them can be placed there [10], assuming all tokens to be there does not decrease the resulting throughput. Therefore the assumption is safe, i.e. no throughput greater than the calculated maximum is possible. The token concentration enables a vectorized execution [12] which makes sense for non-overlapping schedules of one loop. Let us further take an equalized graph, for which we use D_{sum} instead of WST. So, if all tokens are on arc e, only the following node $v = target(e)$ can be activated and that

$$Act_{max}(v) = \lfloor \frac{D_{sum}}{I(v)} \rfloor \qquad (8)$$

times maximum in parallel. By the way, if we take the maximum of $Act_{max}(v)$ over all v of the cycle, we get a maximum utilizable number of processors. During one period (m times $\vec{q}$ activations; m iterations) node v must be activated exactly $m \cdot q[v]$ times. If node v were the only node of the cycle then n_v sequential activation bursts (each of duration T_v) are at least necessary to complete that period.

$$n_v = \lceil \frac{m \cdot q_v}{\lfloor \frac{D_{sum}}{I(v)} \rfloor} \rceil \qquad (9)$$

For the nodes with different names can only be activated in ordered sequence, one timeslot of the schedule is $\sum T_i$.

The necessary count of timeslots is now the maximum of all n_v for all nodes v of the cycle l.

$$n(m) = \max_{v \in l} \lceil \frac{m \cdot q_v}{\lfloor \frac{D_{sum}}{I(v)} \rfloor} \rceil \qquad (10)$$

So eq. 10 shows that we need at least a time $n(m) \cdot \sum T_i$ to complete one period or m iterations. The throughput is therefore limited by

$$TB = \frac{m}{n(m) \cdot \sum T_i} \qquad (11)$$

This leads to the bound in equation 6, completing the proof.

If we apply the characteristic values of a singlerate graph (eqs. 4) to eq. 6, we get

$$TB_{SR,blocked} = \frac{m}{\lceil \frac{m}{D_{sum}} \rceil \cdot \sum T_i} \qquad (12)$$

Eq. 12 is exactly the throughput bound for a singlerate blocked schedule[4]. We could eliminate m by using the optimum [11] (integer multiples of D_{sum}) or by applying the limes with $m \to \infty$ to get equation 5.

6. EXAMPLE

Let us now have a look at the graph G in picture 1. Its characteristic values are:

$$\vec{q}^T = (10, 5, 4), HDM = 8, LLM = 8, CM = 20, T_i = 1 \quad (13)$$

The figure shows G with $WST = 8$. Because it is already equalized (all rates at one node equal) here $WST = D_{sum} = 8$ holds. This is the smallest D_{sum} that keeps the graph alive, although a marking is possible with this D_{sum} that does not guarantee the liveness (e.g. an odd number of delays on e_1). For this graph we can apply the ASAP (as soon as possible) schedule shown on Figure 3.

We compute the necessary activation bursts by eq. 9 as

$$n_1 = \lceil 10/4 \rceil = 3, n_2 = \lceil 5/2 \rceil = 3, n_3 = \lceil 4/1 \rceil = 4. \quad (14)$$

Therefore we need 4 time-slots of $t_1+t_2+t_3$ to complete $m = 1$ iteration. What we observe is that one node execution is the bottleneck constraining the schedule. Figure 2 shows

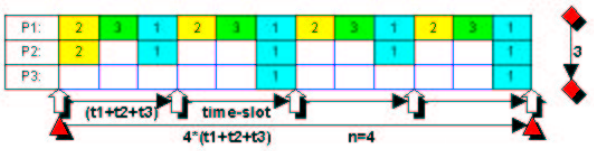

Figure 3. Example schedule

the normalized throughput as a function of the WST. The linear dependency is the bound provided by Murata [7] et al..

The liveness condition has to be checked to obtain the zeroes up to $WST = 7$, otherwise erraneous (but valid) bounds might be introduced by eq. 6. Above CM the same pattern repeats plus one full throughput step of value one and so on. We could consider to decompose one cycle into several with integer multiples of CM plus one with the remaining delays, the throughput of them all accumulates algebraically.

We observed from numerous simulations that a certain kind of point symmetry exists around $WST = (CM + LLM - 1)/2$. In our example this is at $(WST = 13.5, NTB = 1/2)$. We are not able to explain this yet, it seems like a kind of nonexistent-delay-brake that slows down the system if we have less than CM delays.

7. CONCLUSION

We have adressed the problem of finding the quantitative prediction of the maximum throughput the implementation of an multirate algorithm can reach. The formula eq. 6 can be used to determine throughput bounds of each cycle in the multirate dataflow graph. The results show obvious bottlenecks in the design (critical loops) and can be used to selectively improve an algorithm by changing some of the influencing parameters. The anomalies of multirate dataflow graphs [10] lead to the conclusion that a more precise metric cannot be found at all. The expensive transformation to a singlerate graph will give exact information, but the explosion of problem size does not enable the analysis of the bottlenecks. However, the assumption to have a blocking factor m is not always desired. With limited ressources only low values of m are practical, so all of them have to be tested to determine the maximum.

[4] blocked schedule means m iterations in one period, where there is no overlapping between periods

REFERENCES

[1] E. A. Lee and D. G. Messerschmitt, "Synchronous data flow," *Proc. of the IEEE*, vol. 75, pp. 1235–1245, September 1987.

[2] P. D. Hoang and J. M. Rabaey, "Scheduling of DSP programs onto multiprocessors for maximum throughput," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 41, pp. 2225–2235, June 1993.

[3] K. Parhi and D. Messerschmitt, "Static rate-optimal scheduling of iterative data-flow programs via optimum unfolding," *IEEE Transactions on Computers*, vol. 40, pp. 178–195, February 1991.

[4] A. Fettweis, "Realizability of digital filter networks," *AEU*, vol. 30, pp. 90–96, Feb. 1976.

[5] D. B. Johnson, "Finding all the elementary circuits of a directed graph," *Siam Journal on Computing*, vol. 4, pp. 77–84, Mars 1975.

[6] D.Y.Chao and D.T.Wang, "Iteration bounds of singlerate data flow graphs for concurrent processing," *IEEE Transactions on Circuits and Systems*, pp. 629–634, Sept. 1993.

[7] T. Murata, "Petri nets: Properties, analysis and applications," *Proceedings of the IEEE*, vol. 77, pp. 541–580, Apr. 1989.

[8] M. Chao, D.T. Zhou and D. Wang, "Multiple-weighted marked graphs," *Proceedings of IFAC*, pp. 259–262, 1993.

[9] E. Teruel, P. Chrzatowsky-Wachtel, J. Colom, and M. Silva, "On weighted T-systems," *Application and Theory of Petri Nets*, pp. 348–367, 1992.

[10] V. Živojnović, R. Schoenen, and H. Meyr, "On retiming of multirate DSP algorithms," in *Proc. of ICASSP*, vol. VI, (Atlanta), pp. 3310–3313, May 1996.

[11] K. Parhi, "Algorithm transformation techniques for concurrent processors," *Proceedings of the IEEE*, vol. 77, pp. 1879–1895, Dec. 1989.

[12] V. Živojnović, S. Ritz, and H. Meyr, "Retiming of DSP programs for optimum vectorization," in *Proceedings of the ICASSP'94 - Adelaide*, 1994.

[13] S. H. Gerez, S. M. Heemstra de Groot, and O. E. Herrmann, "A polynomial-time algorithm for the computation of the iteration-period bound in recursive dataflow graphs," *IEEE Trans. on Circuits and Systems I*, vol. 39, pp. 49–52, January 1992.

[14] K.Ito and K.K.Parhi, "Determining the minimum iteration period of an algorithm," *Journal of VLSI Signal Processing*, vol. 11, 1995.