Impact of Bit-Width Specification on the Memory Hierarchy for a Real-Time Video Processing System

Benny Thörnberg and Mattias O'Nils
Mid-Sweden University, Sundsvall, SWEDEN
Email: {benny.thornberg|mattias.onils}@miun.se

Abstract

Bit-width specification will affect the total memory storage requirement of a video processing system. However, what is not so obvious is that the bit-width specification will also affect the design of the memory hierarchy. Experiments with a real-life surveillance system show how the optimal allocation of shift registers for the storage of intermediate results is sensitive to bit-widths. It is shown that the total on-chip memory storage requirement can be reduced by 61 percent compared to a non-optimal design.

1. Introduction

In Real-Time Video Processing Systems (RTVPS) huge amounts of information are processed in real-time. Memory accesses are the biggest contributor to power consumption in these systems, which makes the optimization of memory structures and memory access the key design challenge in achieving cost effective implementations for embedded applications [1].

The pre-processing parts of an RTVPS are usually neighborhood oriented. Examples such 3-D operations are convolution, optical flow calculations and scene change detection [2].

Digital Signal Processing (DSP) algorithms are typically initially modeled using floating-point data types. However, at the implementation phase, the data types are refined into a fixed-point representation where the number of used bits must be carefully selected.

Algorithms for digital signal processing systems in general and RTVPS in particular, are often described using a Synchronous Data Flow graph (SDFG) as the dominant programming model.

Fig. 1 shows an example of a memory architecture that implements a spatio-temporal neighborhood. This neighborhood is a group of pixels, used by an image-processing operator as its input and from which it calculates the corresponding output pixel. The computational hardware is not shown in this figure. Using

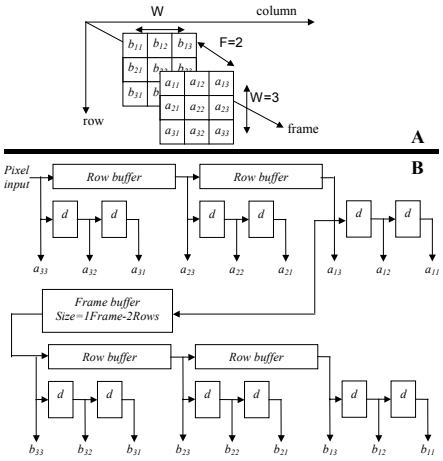

Fig. 1: Spatio-temporal neighborhood.

delay elements, as exemplified in Fig. 1B, is an efficient method of implementing a neighborhood on an FPGA. The corresponding pixel positions in Fig. 1A can be identified as a tap in the delay network as depicted in Fig. 1B. An image-processing operator then calculates one output pixel at every clock cycle. This can be achieved by a pipelined implementation of the computation. Let F be the number of frames that the neighborhood spans, see Fig. 1A. W is the odd number of spanning spatial pixels of a square spatial neighborhood shape.

2. A video camera based surveillance system

We have set up an experiment using a video camera for the surveillance of a car park. The captured video was later used as the input stimuli for the simulation of an RTVPS. Fig. 2A depicts a SDFG of the modeled system. The input video camera is labeled *A*. Nodes *B* to *L* are all neighborhood oriented video operations. Bit-widths are assigned to each edge in this graph. Fig. 2B depicts a series of sample output frames from nodes *A, C, E, H* and *K*. Frame *C* is a calculated background and frame *K* shows the highlighted border of a moving object.

3-9810801-0-6/DATE06 © 2006 EDAA

3. Results and discussion

The impact of alternative buffer allocations have been thoroughly demonstrated in Fig. 2A and Tab. 1. The first column in Tab. 1 is a list of all nodes in the SDFG. Column *2* indicates the input stream for each node. Column *3* and *4* are the size-parameters for each operator. Column *5* lists the latencies for all operators L^{OP} and for all intermediate buffers L^{B}. To simplify the analysis, the latencies caused by the pipeline stages of the computation are all assumed to be one clock cycle. Column *6* lists the memory storage requirement caused by the neighborhood for each operator input S^{M}, and for all intermediate buffers S^{B}. The last column lists the storage required by the computational pipeline stages S^{C}. The numbers in Tab. 1 are based on a frame size of *Rows=460* and *Columns=620*. The size of the buffer at node *O* is reduced by two thirds from *178992* to *59664* bits by re-allocating the buffer from the non-optimal *Allocation 1* to the optimal *Allocation 2*. The total on-chip memory storage is summarized for *Allocation 1* and *Allocation 2*. The storage requirement of buffer *M* is not included in these two sums because this dependency can share storage with the neighborhood of node *I*. Nodes *B* and *C* have large frame buffers that will be stored off-chip. We see that *Allocation 2* reduces the total on-chip memory storage by 61 percent compared to *Allocation 1*. We see in Fig. 2A that the intermediate buffers labeled *M*, *N* and *O* resolve a timing problem for operations *D*, *J* and *K*. The latencies

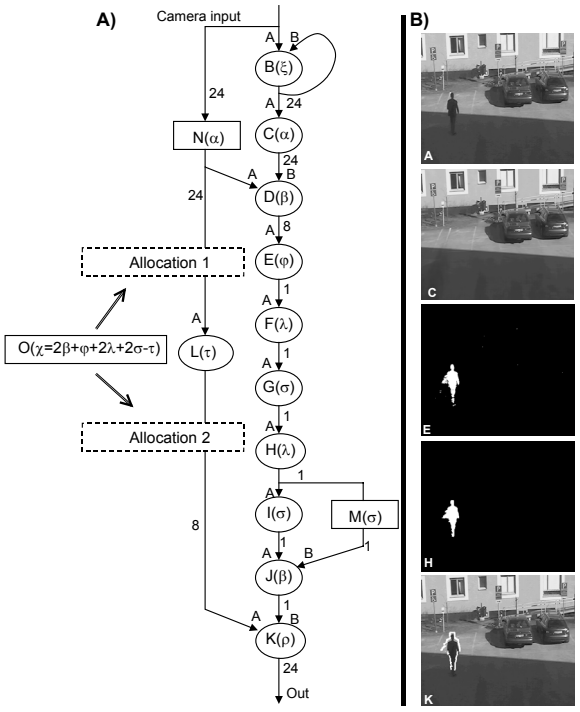

Fig. 2: Surveillance system.

Node	In	W	F	L^{B}/ L^{OP} [Cycles]	S^{M}/S^{B} [Bits]	S^{C} [Bits]
B	A	1	1	ξ=1	0	24
	B	1	3	ξ=1	13689600	
C	A	1	3	α=1	13689600	24
D	A	1	1	β=1	0	8
	B	1	1	β=1	0	
E	A	1	1	φ=1	0	1
F	A	5	1	λ=1243	2484	1
G	A	9	1	σ=2485	4968	1
H	A	5	1	λ=1243	2484	1
I	A	9	1	σ=2485	4968	1
J	A	1	1	β=1	0	1
	B	1	1	β=1	0	
K	A	1	1	ρ=1	0	24
	B	1	1	ρ=1	0	
L	A	1	1	τ=1	0	8
M	A	-	-	σ=2485	2485	0
N	A	-	-	ξ+α=2	48	0
O	A	-	-	χ=7458	178992	0
On-chip storage Allocation 1					194038	
O	A	-	-	χ=7458	59664	0
On-chip storage Allocation 2					74710	
Total off-chip storage					13689600	

Tab. 1: Timing and memory requirements.

for each of the nodes *B* to *L* are denoted as ξ, α, β, φ, λ, σ and ρ. The latency for the buffer at node *M* equals the latency of node *I* in order for the frames of the two input streams to appear simultaneously without any phase shift at node *J*. Similarly, the latency for the buffer at node *N* equals the sum of the latencies of node *B* and *C* in order for the frames of the two input streams to appear simultaneously without any phase shift at node *D*. As for the buffer at node *O*, the first conclusion to be drawn is that $2\beta+\varphi+2\lambda+2\sigma \geq \tau$ and thus it is necessary for the buffer to be allocated to the left-side path in the dataflow depicted in Fig. 2A. The timing constraint for the two input streams at node *K* can then be expressed as $\chi + \tau = 2\beta + \varphi + 2\lambda + 2\sigma$. Two different allocations are then possible for node *O*.

The on-chip memory reduction of 61 percent clearly demonstrates the significance of this optimization problem, that we as future work intend to automate.

References

[1] Catthoor, F. et al.: *Custom Memory Management Methodology*. Kluwer Academic Publishers (1998).

[2] Bovik, A.: *Handbook of Image&Video Processing*. Academic Press (2000).